

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Reihaneh Safavi-Naini Ran Canetti (Eds.)

Advances in Cryptology – CRYPTO 2012

32nd Annual Cryptology Conference
Santa Barbara, CA, USA, August 19-23, 2012
Proceedings

Volume Editors

Reihaneh Safavi-Naini

University of Calgary, Department of Computer Science
2500 University Drive NW, Calgary, AB T2N 1N4, Canada
E-mail: rei@ucalgary.ca

Ran Canetti

Boston University, Department of Computer Science
111 Cummington Street, Boston, MA 02215, USA
E-mail: canetti@bu.edu

and

Tel Aviv University, Blavatnik School of Computer Science
Tel Aviv, Israel
E-mail: canetti@tau.ac.il

ISSN 0302-9743
ISBN 978-3-642-32008-8

DOI 10.1007/978-3-642-32009-5

Springer Heidelberg Dordrecht London New York

e-ISSN 1611-3349
e-ISBN 978-3-642-32009-5

Library of Congress Control Number: 2012942915

CR Subject Classification (1998): E.3, G.2.1, F.2.1-2, D.4.6, K.6.5, C.2, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

© International Association for Cryptologic Research 2012

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

CRYPTO 2012, the 32nd Annual International Cryptology Conference, was held August 19–23 on the campus of the University of California, Santa Barbara. The event was sponsored by the International Association for Cryptologic Research (the IACR) in cooperation with the UCSB Computer Science Department and the IEEE Computer Society’s Technical Committee on Security and Privacy.

We received 225 submissions of which 48 were accepted for publication, a record number for IACR flagship conferences. These proceedings contains the revised versions of all the papers. One pair of papers shared a single presentation slot in the program (marked in the Table of Contents). There were also two invited talks. On Monday, Jonathan Zittrain, Professor of Law and Computer Science at Harvard University, gave a talk entitled “The End of Crypto.” On Wednesday, Ernie Brickell, Chief Security Architect for Intel Corporation, spoke about “Recent Advances and Existing Research Questions in Platform Security.” To accommodate the increase in the number of accepted papers, one paper presentation session was planned on the traditionally free Tuesday afternoon. This session was followed by a tutorial session on differential privacy, entitled “Pinning Down ‘Privacy’ in Statistical Databases: A Tutorial,” delivered by Adam Smith. The rump session was as usual on Tuesday evening, and was chaired by Dan Bernstein and Tanja Lange.

Our goal was to have a technical program that is strong and representative of the diversity and breadth of cryptologic research. Toward this goal we took a number of steps including selecting a large Program Committee (PC) with diverse research interest and experience, and in the Call for Papers, encouraging submissions in all areas of cryptology with emphasis on innovative application and approaches. Papers were reviewed double-blind, with non-PC-member papers assigned to three reviewers, and PC member papers to four reviewers. During the discussion phase, when necessary, extra reviews were solicited. As part of paper discussion phase we held a PC meeting at Cambridge, UK, collocated with Eurocrypt 2012. We ensured that all papers received fair and objective evaluation by experts and also a broader group of PC members, with particular attention paid to highlighting strengths and weaknesses of papers. The final decisions were made based on the reviews and discussion, and in the case of two “equal” papers, taking other factors such as balance of the program into account. In the case of ambiguity we relayed questions of reviewers to authors, and delivered back the responses, with all communications anonymized. The task of paper selection was especially challenging given the high number of strong submissions. In the end, a sizable number of strong papers could not be included in the program for lack of space.

For the Best Paper Award, the PC overwhelmingly selected “Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks,

and Combinatorial Search Problems,” by Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. The PC praised the work for important cryptographic applications in symmetric cryptography and public-key cryptography, as well as applications to other combinatorial problems.

We would like to sincerely thank authors of all submissions – those whose paper made it into the program and those whose papers did not. We and the PC as a whole were impressed by the quality of submissions contributed from all around the world. Although this made the task of selecting the final list very challenging, it gave us the opportunity to have a very strong and diverse program.

Our sincere gratitude also goes out to the PC. We were extremely fortunate that so many brilliant people put such an inordinate amount of time not only in writing reviews, but also actively participating in discussions for nearly six weeks. They responded promptly to our requests for extra reviews, opinions, comments, comparisons and inputs. We were extremely impressed by the knowledge, dedication, and integrity of our PC. We are also indebted to the many external reviewers who significantly contributed to the comprehensive evaluation of papers. A list of PC members and external reviewers appears after this note. Despite all our efforts, the list of external reviewers may have errors or omissions and we apologize for that.

We would like to thank Yiqun Lisa Yin, the General Chair, for working closely with us throughout the whole process, providing the much-needed support in every step, including creating and maintaining the website, and planning and organizing the logistics of the PC meeting and the conference.

We benefited enormously from advice and feedback of the past CRYPTO Program Chairs, Phil Rogaway, Tal Rabin, and Shai Halevi. They generously shared with us their experiences that enabled us to take more informed decisions. Shai Halevi also provided us with unlimited support of his software *websubrev* that we used for the whole conference planning, paper evaluation, and interaction with PC members and authors. Josh Benaloh was our IACR point of contact, always providing timely and informative answers to our questions. Alfred Hofsäss and his colleagues at Springer provided a meticulous service for the timely production of this volume.

We would like to thank Google, Microsoft Research, Qualcomm, RIM, and Voltage Security for their generous support.

Serving as CRYPTO Program Co-chairs was a privilege and also a great challenge. This was the first year that CRYPTO was to implement Co-chairing, with Rei serving as Senior Co-chair, having tie-breaker decision role. Despite many unknowns and the need for extra effort to define the processes and order of things in Co-chairing, in the end it was a great opportunity to work together and build on our strengths. We are happy that we took the challenge and along the way found new friendships in addition to the running of the conference.

CRYPTO 2012

The 32nd Annual International Cryptology Conference
Santa Barbara, California, USA
August 19–23, 2012

Sponsored by the
International Association of Cryptologic Research (IACR)
in cooperation with the
Computer Science Department of the University of California, Santa Barbara
and the
IEEE Computer Society's Technical Committee on Security and Privacy

General Chair

Yiqun Lisa Yin Independent Security Consultant, USA

Program Co-chairs

Rei Safavi-Naini
Ren Canetti
University of Calgary, Canada
Boston University (USA) and Tel Aviv
University (Israel)

Program Committee

Benny Applebaum	Tel Aviv University, Israel
Dan Boneh	Stanford, USA
Colin Boyd	QUT, Australia
Ivan Damgård	Aarhus University, Denmark
Yevgeniy Dodis	New York University, USA
Serge Fehr	CWI Amsterdam, The Netherlands
Cédric Fournet	Microsoft Research, UK
Marc Fischlin	Darmstadt University of Technology, Germany
Pierre-Alain Fouque	École Normale Supérieure, France
Juan Garay	AT&T Labs - Research, USA
Steven Galbraith	The University of Auckland, New Zealand
Jens Groth	University College London, UK
Susan Hohenberger	Johns Hopkins University, USA
Yuval Ishai	Technion, Israel
Ari Juels	RSA Laboratories, USA

Yael Kalai	Microsoft Research, USA
Hugo Krawczyk	IBM Research, USA
Ralf Küsters	University of Trier, Germany
Aggelos Kiayias	University of Connecticut, USA
Kaoru Kurosawa	Ibaraki University, Japan
Stefan Lucks	Bauhaus-Universität Weimar, Germany
Tal Malkin	Columbia University, USA
Alexander May	Ruhr University Bochum, Germany
Daniele Micciancio	University of California at San Diego, USA
Kaisa Nyberg	Aalto University and Nokia, Finland
Tatsuaki Okamoto	NTT, Japan
Kenny Paterson	Royal Holloway, University of London, UK
Chris Peikert	Georgia Tech, USA
Thomas Peyrin	Nanyang Technological University, Singapore
Renato Renner	ETH Zurich, Switzerland
Palash Sarkar	Indian Statistical Institute, Kolkata, India
François-Xavier Standaert	UCL, Belgium
Damien Stehlé	CNRS and ENS de Lyon, France
Thomas Shrimpton	Portland State University, USA
Tsuyoshi Takagi	Kyushu University, Japan
Eran Tromer	Tel Aviv University, Israel
Dominique Unruh	University of Tartu, Estonia
Vinod Vaikuntanathan	University of Toronto, Canada

Advisory Member

Phil Rogaway (CRYPTO 2011 Program Chair)	University of California, Davis, USA
---	--------------------------------------

External Reviewers

Masayuki Abe	Santiago Zanella Beguelin	Zvika Brakerski
Hadi Ahmadi	Amos Beimel	Christina Brzuska
Mohsen Alimomeni	Mihir Bellare	Ignacio Cascudo
Jacob Alperin-Sheriff	Rikke Bendlin	David Cash
Elena Andreeva	Mario Berta	Nathan Chenette
Kazumaro Aoki	Rishiraj Bhattacharyya	Jung Hee Cheon
Yoshinori Aono	Nir Bitansky	Alessandro Chiesa
Gilad Asharov	Céline Blondeau	Ashish Choudhury
Jean-Philippe Aumasson	Andrey Bogdanov	Baudoin Collard
Paul Baecher	Joppe W. Bos	Jason Crampton
Thomas Baignères	Niek Bouman	Cas Cremers
Josep Balasch	Xavier Boyen	Dana Dachman-Soled
Stephanie Bayer	Elette Boyle	George Danezis
Normand Beaudry		Anindya De

Jean Paul Degabriele	Brett Hemenway	Marine Minier
Yi Deng	Gottfried Herold	Arno Mittelbach
Claus Diem	Alejandro Hevia	Payman Mohassel
Jintai Ding	Martin Hirt	Nicky Mouha
Dejan Dukaric	Viet Tung Hoang	Sean Murphy
Frederic Dupuis	Dennis Hofheinz	Steve Myers
Junfeng Fan	Pavel Hubacek	David Naccache
Pooya Farshim	Yuval Ishai	Jesper Buus Nielsen
Sebastian Faust	Mitsugu Iwamoto	Ryo Nishimaki
Dario Fiore	Abhishek Jain	Kobbi Nissim
Ewan Fleischmann	Stanislaw Jarecki	Peter Nordholdt
Christian Forler	Pascal Junod	Adam O'Neill
David Freeman	Charanjit Jutla	Wakaha Ogata
Eduarda Freire	Kimmo Järvinen	Cristina Onete
Georg Fuchsbauer	Nikos Karvelas	Claudio Orlandi
Eiichiro Fujisaki	Jonathan Katz	Carles Padro
Jakob Funder	Yutaka Kawai	Bryan Parno
Jun Furukawa	Eike Kiltz	Rafael Pass
Ariel Gabizon	Susumu Kiyoshima	Valerio Pastro
Tommaso Gagliardoni	François Koeune	Arpita Patra
David Galindo	Markulf Kohlweiss	Olivier Pereira
Sanjam Garg	Vladimir Kolesnikov	Paolo Palmieri
Pierrick Gaudry	Sara Krehbiel	Olivier Pereira
Peter Gazi	Virendra Kumar	Ludovic Perret
Rosario Gennaro	Noboru Kunihiro	Thomas Peters
Wesley George	Fabien Laguillaumie	Christophe Petit
Benoit Gerard	Adeline Langlois	Le Trieu Phong
Benedikt Gierlich	Gregor Leander	Krzysztof Pietrzak
Ian Goldberg	Anja Lehmann	Benny Pinkas
Sharon Goldberg	Allison Lewko	Christopher Portmann
Alonso Gonzalez	Benoit Libert	Manoj Prabhakaran
Serge Gorbunov	Rachel Lin	Gordon Proctor
Dov Gordon	Yehuda Lindell	Xavier Pujol
Vipul Goyal	Adriana Lopez-Alt	Tal Rabin
Jian Guo	Edward Lui	Charles Rackoff
Robbert de Haan	Vadim Lyubashevsky	Ananth Raghunathan
Iftach Haitner	Bernardo Machado	Tamara Rezk
Shai Halevi	David	Tom Ristenpart
Risto Hakala	Mark Manulis	Matt Robshaw
Keisuke Hakuta	Giorgia Azzurra Marson	Yannis Rouselakis
Goichiro Hanaoka	Ryutaro Mastumoto	Amit Sahai
Guillaume Hanrot	Takahiro Matsuda	Yusuke Sakai
Yasufumi Hashimoto	Florian Mendel	Kazuo Sakiyama
Taku Hayashi	Bart Mennink	Katerina Samari
Carmit Hazay	Alexander Meurer	Yu Sasaki

Christian Schaffner	Stefano Tessaro	Hoeteck Wee
Thomas Schneider	Enrico Thomae	Lei Wei
Dominique Schroeder	Emmanuel Thomé	Jakob Wenzel
Jacob Schuldt	Susan Thomson	Daniel Wichs
Sven Schäge	Marco Tomamichel	Douglas Wikstrom
Gil Segev	Nikos Triandopoulos	Christopher Wolf
Abhi Shelat	Yiannis Tselekounis	David Woodruff
Chih-hao Shen	Max Tuengerthal	Shota Yamada
Dave Singelée	Jon Ullman	Go Yamamoto
Adam Smith	Yevgeniy Vahlis	Takanori Yasuda
Ben Smith	Margarita Vald	Tomoko Yonemura
Douglas Stebila	Serge Vaudenay	Kazuki Yoneyama
Emil Stefanov	Muthu Venkitasubramaniam	Yu Yu
John Steinberger	Daniele Venturi	Hila Zarosim
Ron Steinfeld	Ivan Visconti	Mark Zhandry
Uri Stemmer	Andreas Vogt	Mingwu Zhang
Paul Syverson	Felipe Voloch	Hong-Sheng Zhou
Bjoern Tackmann	Michael Walter	Vassilis Zikas
Katsuyuki Takashima	Bogdan Warinschi	Angela Zottarel
Qiang Tang	Gaven Watson	
Aris Tentes		

Table of Contents

Symmetric Cryptosystems

An Enciphering Scheme Based on a Card Shuffle	1
<i>Viet Tung Hoang, Ben Morris, and Phillip Rogaway</i>	
Tweakable Blockciphers with Beyond Birthday-Bound Security	14
<i>Will Landecker, Thomas Shrimpton, and R. Seth Terashima</i>	
Breaking and Repairing GCM Security Proofs	31
<i>Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu</i>	
On the Distribution of Linear Biases: Three Instructive Examples	50
<i>Mohamed Ahmed Abdelraheem, Martin Ågren, Peter Beelen, and Gregor Leander</i>	
Substitution-Permutation Networks, Pseudorandom Functions, and Natural Proofs	68
<i>Eric Miles and Emanuele Viola</i>	

Invited Talk

The End of Crypto	86
<i>Jonathan Zittrain</i>	

Secure Computation I

Must You Know the Code of f to Securely Compute f ?	87
<i>Mike Rosulek</i>	
Adaptively Secure Multi-Party Computation with Dishonest Majority	105
<i>Sanjam Garg and Amit Sahai</i>	
Collusion-Preserving Computation	124
<i>Joël Alwen, Jonathan Katz, Ueli Maurer, and Vassilis Zikas</i>	
Secret Sharing Schemes for Very Dense Graphs	144
<i>Amos Beimel, Oriol Farràs, and Yuval Mintz</i>	

Attribute-Based and Functional Encryption

Functional Encryption with Bounded Collusions via Multi-party Computation	162
<i>Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee</i>	
New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques	180
<i>Allison Lewko and Brent Waters</i>	
Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption	199
<i>Amit Sahai, Hakan Seyalioglu, and Brent Waters</i>	
Functional Encryption for Regular Languages	218
<i>Brent Waters</i>	

Proof Systems

Secure Database Commitments and Universal Arguments of Quasi Knowledge	236
<i>Melissa Chase and Ivan Visconti</i>	
Succinct Arguments from Multi-prover Interactive Proofs and Their Efficiency Benefits	255
<i>Nir Bitansky and Alessandro Chiesa</i>	

Protocols

On the Security of TLS-DHE in the Standard Model	273
<i>Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk</i>	
Semantic Security for the Wiretap Channel	294
<i>Mihir Bellare, Stefano Tessaro, and Alexander Vardy</i>	
Multi-instance Security and Its Application to Password-Based Cryptography	312
<i>Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro</i>	

Hash Functions

Hash Functions Based on Three Permutations: A Generic Security Analysis	330
<i>Bart Mennink and Bart Preneel</i>	

To Hash or Not to Hash Again? (In)Differentiability Results for H^2 and HMAC	348
<i>Yevgeniy Dodis, Thomas Ristenpart, John Steinberger, and Stefano Tessaro</i>	
New Preimage Attacks against Reduced SHA-1	367
<i>Simon Knellwolf and Dmitry Khovratovich</i>	

Stam's Conjecture and Threshold Phenomena in Collision Resistance	384
<i>John Steinberger, Xiaoming Sun, and Zhe Yang</i>	

Composable Security

Universal Composability from Essentially Any Trusted Setup	406
<i>Mike Rosulek</i>	

Impossibility Results for Static Input Secure Computation	424
<i>Sanjam Garg, Abishek Kumarasubramanian, Rafail Ostrovsky, and Ivan Visconti</i>	

New Impossibility Results for Concurrent Composition and a Non-interactive Completeness Theorem for Secure Computation	443
<i>Shweta Agrawal, Vipul Goyal, Abhishek Jain, Manoj Prabhakaran, and Amit Sahai</i>	

Black-Box Constructions of Composable Protocols without Set-Up	461
<i>Huijia Lin and Rafael Pass</i>	

Privacy

Crowd-Blending Privacy	479
<i>Johannes Gehrke, Michael Hay, Edward Lui, and Rafael Pass</i>	

Differential Privacy with Imperfect Randomness	497
<i>Yevgeniy Dodis, Adriana López-Alt, Ilya Mironov, and Salil Vadhan</i>	

Leakage and Side-Channels

Tamper and Leakage Resilience in the Split-State Model	517
<i>Feng-Hao Liu and Anna Lysyanskaya</i>	

Securing Circuits against Constant-Rate Tampering	533
<i>Dana Dachman-Soled and Yael Tauman Kalai</i>	

How to Compute under \mathcal{AC}^0 Leakage without Secure Hardware	552
<i>Guy N. Rothblum</i>	

Invited Talk

Recent Advances and Existing Research Questions in Platform Security	570
<i>Ernie Brickell</i>	

Signatures

Group Signatures with Almost-for-Free Revocation	571
<i>Benoît Libert, Thomas Peters, and Moti Yung</i>	
Tightly Secure Signatures and Public-Key Encryption	590
<i>Dennis Hofheinz and Tibor Jager</i>	

Implementation Analysis

Efficient Padding Oracle Attacks on Cryptographic Hardware	608
<i>Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay</i>	
Public Keys	626
<i>Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter</i>	

Secure Computation II

Multiparty Computation from Somewhat Homomorphic Encryption	643
<i>Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias</i>	
Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority	663
<i>Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky</i>	
A New Approach to Practical Active-Secure Two-Party Computation...	681
<i>Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra</i>	

Black-Box Separation

The Curious Case of Non-Interactive Commitments – On the Power of Black-Box vs. Non-Black-Box Use of Primitives	701
<i>Mohammad Mahmoody and Rafael Pass</i>	

Cryptanalysis

Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems	719
<i>Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir</i>	
Resistance against Iterated Attacks by Decorrelation Revisited	741
<i>Ash Bay, Atefeh Mashatan, and Serge Vaudenay</i>	

Quantum Cryptography

Secure Identity-Based Encryption in the Quantum Random Oracle Model	758
<i>Mark Zhandry</i>	
Quantum to Classical Randomness Extractors	776
<i>Mario Berta, Omar Fawzi, and Stephanie Wehner</i>	
Actively Secure Two-Party Evaluation of Any Quantum Operation	794
<i>Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail</i>	

Key Encapsulation and One-Way functions

On the Impossibility of Constructing Efficient Key Encapsulation and Programmable Hash Functions in Prime Order Groups	812
<i>Goichiro Hanaoka, Takahiro Matsuda, and Jacob C.N. Schuldt</i>	
Hardness of Computing Individual Bits for One-Way Functions on Elliptic Curves	832
<i>Alexandre Duc and Dimitar Jetchev</i>	
Homomorphic Evaluation of the AES Circuit	850
<i>Craig Gentry, Shai Halevi, and Nigel P. Smart</i>	
Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP	868
<i>Zvika Brakerski</i>	
Author Index	887

An Enciphering Scheme Based on a Card Shuffle

Viet Tung Hoang¹, Ben Morris², and Phillip Rogaway¹

¹ Dept. of Computer Science,

² Dept. of Mathematics,

University of California, Davis, USA

Abstract. We introduce the *swap-or-not shuffle* and show that the technique gives rise to a new method to convert a pseudorandom function (PRF) into a pseudorandom permutation (PRP) (or, alternatively, to directly build a confusion/diffusion blockcipher). We then prove that swap-or-not has excellent quantitative security bounds, giving a Luby-Rackoff type result that ensures security (assuming an ideal round function) to a number of adversarial queries that is nearly the size of the construction’s domain. Swap-or-not provides a direct solution for building a small-domain cipher and achieving format-preserving encryption, yielding the best bounds known for a practical scheme for enciphering credit-card numbers. The analysis of swap-or-not is based on the theory of mixing times of Markov chains.

Keywords: Blockciphers, Feistel network, Luby-Rackoff, Markov chain, PRF-to-PRP conversion, pseudorandom permutations, swap-or-not.

1 Introduction

OVERVIEW. Despite the diversity of proposed blockciphers, only two approaches underlie the construction of real-world designs: essentially everything looks like some sort of Feistel network (e.g., DES, FEAL, MARS, RC6) or SP-network (e.g., Rijndael, Safer, Serpent, Square). Analogously, in the literature on constructing pseudorandom permutations (PRPs) from pseudorandom functions (PRFs), we have provable-security analyses for Feistel variants (e.g., [12–14, 18, 21]), as well as modes of operation (e.g., [10, 11, 18, 19]) that can again be construed as SP-networks, now on a large domain. Perhaps there just are not that many fundamentally different ways to make a blockcipher. Or perhaps we might have failed to notice *other* possibilities.

In this short paper we describe a very different way to make a blockcipher. We call it a *swap-or-not* network (or cipher or shuffle). Besides introducing the construction, we evidence its cryptographic utility. We do this by showing that swap-or-not provides the quantitatively best mechanism known, in terms of concrete security bounds, to convert a PRF into a PRP. We also show that swap-or-not provides a practical solution for the problem of format-preserving encryption (FPE) on domains of troublesome size, such as enciphering credit-card numbers.

```

proc  $E_{KF}(X)$  //swap-or-not
for  $i \leftarrow 1$  to  $r$  do
   $X' \leftarrow K_i \oplus X$ 
   $\hat{X} \leftarrow \max(X, X')$ 
  if  $F_i(\hat{X}) = 1$  then  $X \leftarrow X'$ 
return  $X$ 

```

Fig. 1. Cipher $E = \text{SN}[r, n]$ encrypts $X \in \{0, 1\}^n$ using a key KF naming $K_1, \dots, K_r \in \{0, 1\}^n$ and round functions $F_1, \dots, F_r : \{0, 1\}^n \rightarrow \{0, 1\}$

that this works, that one gets a permutation, is simply that $X \mapsto K_i \oplus X$ is an involution, and our round function depends on the set $\{X, K_i \oplus X\}$. The inverse direction for swap-or-not is identical to the forward direction shown above except for having i run from r down to 1.

Restating the algorithm in English, at each round i we pair the current value of $X \in \{0, 1\}^n$ with a “partner” point $X' = K_i \oplus X$. We either replace X by its partner or leave it alone. Which of these two things we do is determined by applying the boolean-valued F_i to the two-element set $\{X, X'\}$. Actually, in order to give F_i a more conventional domain, we select a canonical representative from $\{X, X'\}$, say $\hat{X} = \max(X, X')$, and apply F_i to it. Note that each plaintext maps to a ciphertext by xoring into it some subset of the subkeys $\{K_1, \dots, K_r\}$. This might sound linear, but it most definitely is not.

CARD SHUFFLING VIEW. The swap-or-not construction was invented, and will be analyzed, by regarding it as a way to shuffle a deck of cards. Seeing a blockcipher as a card shuffle enables one to exploit a large body of mathematical techniques, these dating back to the first half of the twentieth century. In addition, some ways to shuffle cards give rise to enciphering schemes that cryptographers did not consider. Swap-or-not is such a case.

One can always see a card shuffle as an enciphering scheme, and vice versa. If you have some method to shuffle N cards, this determines a corresponding way to encipher N points: place a card at each position $X \in [N]$, where $[N] = \{0, 1, \dots, N-1\}$; shuffle the deck; then look to see the position where the card initially at position X ended up. Call that position the ciphertext Y for X . The randomness used in the shuffle corresponds the cipher’s key.

The first thing needed for a card shuffle to give rise to a computationally feasible blockcipher is that the shuffle be *oblivious*, an idea suggested by Moni Naor [18, p. 62], [23, p. 17]. In an oblivious shuffle one can trace the trajectory of a card without attending to lots of *other* cards in the deck. Most conventional shuffles, such as the riffle shuffle, are not oblivious. The Thorp shuffle [26] is oblivious—and so is swap-or-not. As a shuffle, here’s how it looks.

CONSTRUCTION. Suppose we aim to encipher n -bit strings; our message space is the set $\mathcal{X} = \{0, 1\}^n$. Assume we will use r rounds, and that the blockcipher’s key KF names subkeys $K_1, \dots, K_r \in \{0, 1\}^n$ as well as round functions F_1, \dots, F_r , each of which maps n -bits to a single bit, so $F_i : \{0, 1\}^n \rightarrow \{0, 1\}$. Then we encipher $X \in \{0, 1\}^n$ as shown in Fig. 1. The reason

Recasting swap-or-not as a way to shuffle cards, suppose we have N cards, one at each position $X \in [N]$, where $N = 2^n$. To shuffle the deck, choose a random $K \in \{0, 1\}^n$ and then, for each pair of card positions X and $K \oplus X$, flip a fair coin. If it lands heads, swap the cards at the indicated positions; if it lands tails, leave them alone. See Fig. 2. The process can be repeated any number r times, using independent coins (both the K -values and the b -values) for each shuffle.

When the swap-or-not shuffle of Fig. 2 is translated back into the language of encryption, one recovers the swap-or-not cipher of Fig. 1; these are different views of precisely the same process. The random pairing-up of cards specified by K for the i th shuffle corresponds to the subkey K_i . The random bit b flipped at the shuffle's round i for the pair $\{X, K \oplus X\}$ corresponds $F_i(\hat{X})$.

```
proc  $E_{KF}(X)$           //Generalized domain
for  $i \leftarrow 1$  to  $r$  do
     $X' \leftarrow K_i - X$ 
     $\hat{X} \leftarrow \max(X, X')$ 
    if  $F_i(\hat{X}) = 1$  then  $X \leftarrow X'$ 
return  $X$ 
```

Fig. 3. Cipher $E = \text{SN}[r, N, +]$ encrypts $X \in [N]$ using a key KF naming $K_1, \dots, K_r \in [N]$ and round functions $F_1, \dots, F_r: [N] \rightarrow \{0, 1\}$

a power of two—we'll be able to encipher points on any set $\mathcal{X} = [N]$, just by naming a group operator, say addition modulo N . For generalizing the shuffle of Fig. 2, the value K is uniformly drawn from $[N]$ rather than from $\{0, 1\}^n$, and we consider the pair of positions $\{X, K - X\}$ rather than $\{X, K \oplus X\}$. For the generalized cipher—see Fig. 3—the key KF will name subkeys $K_1, \dots, K_r \in [N]$ and round functions $F_1, \dots, F_r: [N] \rightarrow \{0, 1\}$. We set $X' \leftarrow K_i - X$ rather than $X' \leftarrow K_i \oplus X$. The inverse remains what one gets by iterating from r down to 1.

RESULTS. As with Luby and Rackoff's seminal paper [14], we can analyze the swap-or-not construction by regarding its constituent parts as uniformly random. Formally, let us write $\text{SN}[r, N, +]: \mathcal{K} \times [N] \rightarrow [N]$ for the blockcipher E specified in Fig. 3 that is swap-or-not with r rounds, a message space of $[N]$, the indicated group operator, and where the key space names all possible subkeys $K_1, \dots, K_r \in [N]$ and all possible round functions $F_1, \dots, F_r: [N] \rightarrow \{0, 1\}$.

```
K  $\xleftarrow{\$} \{0, 1\}^n$       //swap-or-not as a shuffle
for each pair of positions  $\{X, K \oplus X\}$ 
    b  $\xleftarrow{\$} \{0, 1\}$ 
    if  $b = 1$  then swap the cards
        at positions  $X$  and  $K \oplus X$ 
```

Fig. 2. Mixing a deck of $N = 2^n$ cards, each at a position $X \in \{0, 1\}^n$. The code shows one shuffle. For better mixing, the shuffle is repeated r times.

GENERALIZING. It is useful to be a bit more general here, working in a finite abelian group $G = ([N], +)$ instead of the group $(\{0, 1\}^n, \oplus)$ of bit strings under xor. (For convenience, we have assumed that the group elements are named $[N] = \{0, \dots, N-1\}$.) In this way we won't need the number of points N in the message space $\mathcal{X} = [N]$ to be

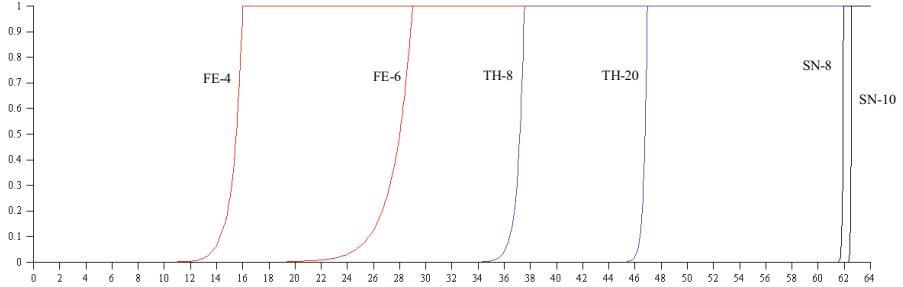


Fig. 4. Illustration of results. The message space has $N = 2^{64}$ points. The graphs show established upper bounds on CCA advantage when the adversary asks q queries, where $\log_2(q)$ labels the x -axis. **Rightmost two graphs:** the new results—the swap-or-not cipher for either eight passes (512 rounds) (SN-8) or 10 (SN-10), as given by Theorem 4. (One pass is defined as $\lceil \lg N \rceil$ rounds.) For comparison, the **leftmost two graphs** are for balanced Feistel, both the classical 4-round result of Luby and Rackoff [14, 20] (LR-4) and then a six-round result of Patarin (LR-6) [22, Th. 7]. The **middle two graphs** are for the Thorp shuffle, either with eight passes (TH-8) or 20 (TH-20), as given by [17, Th. 5].

Thus a random key KF for this cipher has the K_i and F_i values uniformly chosen. We define the CCA (also called the “strong-PRP”) advantage of an adversary A attacking E by dropping it into one of two worlds. In the first, the adversary gets an oracle for $E_{KF}(\cdot)$, for a random KF , and also an oracle for its inverse, $E_{KF}^{-1}(\cdot)$. Alternatively, the adversary is given a uniformly random permutation $\pi: [N] \rightarrow [N]$, along with its inverse, $\pi^{-1}(\cdot)$. Define

$$\mathbf{Adv}_{\text{SN}[r,N,+]}^{\text{cca}}(q) = \max_A \left\{ \Pr[A^{E_{KF}(\cdot), E_{KF}^{-1}(\cdot)} \Rightarrow 1] - \Pr[A^{\pi(\cdot), \pi^{-1}(\cdot)} \Rightarrow 1] \right\},$$

the maximum over all adversaries that ask at most q total queries. Our main result is that

$$\mathbf{Adv}_{\text{SN}[r,N,+]}^{\text{cca}}(q) \leq \frac{4N^{3/2}}{r+4} \left(\frac{q+N}{2N} \right)^{r/4+1}. \quad (1)$$

Roughly said, you need $r = 6 \lg N$ rounds of swap-or-not to start to see a good bound on CCA-security. After that, the adversary’s advantage drops off inverse exponentially in r . The summary explanation of formula (1) just given assumes that the number of adversarial queries is capped at $q = (1 - \epsilon)N$ for some fixed $\epsilon > 0$.

The quantitative guarantee above is far stronger than anything a balanced Feistel network can deliver. The only remotely comparable bound we know, retaining security to $N^{1-\epsilon}$ queries instead of $(1 - \epsilon)N$ queries, is the Thorp shuffle [26] (or, equivalently, a maximally-unbalanced Feistel network [17]). But the known result, establishing $\mathbf{Adv}_{E'}^{\text{cca}}(q) \leq (2q/r + 1)(4nq/N)^r$ if one shuffles

$N = 2^n$ points for $r(4n - 2)$ rounds [17], vanishes by the time that $q \geq \frac{N}{4\lg N}$. Numerically, the Thorp-shuffle bounds come out much weaker for most r , q , and N . See Fig. 4 for sample graphs comparing known bounds on balanced Feistel, the Thorp shuffle, and swap-or-not.

As a simple numerical example, swap-or-not enciphering 64-bit strings for 1200 rounds using a random round function will yield a maximal CCA advantage of less than 10^{-10} , even if the adversary can ask $q = 2^{63}$ queries. While the number of rounds is obviously large, no other construction can deliver a comparable guarantee, achieving security even when q is close to N .

For a more complexity-theoretic discussion of swap-or-not, see Section 4.

FORMAT-PRESERVING ENCRYPTION. Swap-or-not was originally invented as a solution for *format-preserving encryption* (FPE) [1, 3, 5], where it provides the best known solution, in terms of proven-security bounds, when N is too big to spend linear time computing, yet too small for conventional constructions to deliver desirable bounds. This landscape has not much changed with the recent work of Stefanov and Shi [24], who, following Granboulan and Pornin [9], show how to speed up (e.g., to $\tilde{\Theta}(N^{0.5})$ time) determining where a card goes in a particular N -card shuffle after spending $\tilde{\Theta}(N)$ time at key-setup. For more discussion of swap-or-not and its use in FPE, see Section 5.

2 Preliminaries

TOTAL VARIATION DISTANCE. Let μ and ν be probability distributions on Ω . The *total variation distance* between distributions μ and ν is defined as

$$\|\mu - \nu\| = \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)| = \max_{S \subseteq \Omega} \{\mu(S) - \nu(S)\} .$$

BLOCKCIPHERS. Let $E: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{M}$ be a blockcipher, meaning that \mathcal{K} and \mathcal{M} are finite and each $E_K(\cdot) = E(K, \cdot)$ is a permutation on \mathcal{M} . We emphasize that \mathcal{K} and \mathcal{M} need not consist of binary strings of some particular length, as is often assumed to be the case. For any blockcipher E , we let E^{-1} be its inverse blockcipher.

For blockcipher $E: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{M}$ and adversary A the *advantage* of A in carrying out an (adaptive) chosen-ciphertext attack (CCA) on E is

$$\mathbf{Adv}_E^{\text{cca}}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}: A^{E_K(\cdot), E_K^{-1}(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \text{Perm}(\mathcal{M}): A^{\pi(\cdot), \pi^{-1}(\cdot)} \Rightarrow 1].$$

Here $\text{Perm}(\mathcal{M})$ is the set of all permutations on \mathcal{M} . We say that A carries out an (adaptive) chosen-plaintext attack (CPA) if it asks no queries to its second oracle. Adversary A is *non-adaptive* if it asks the same queries on every run. Let $\mathbf{Adv}_E^{\text{cca}}(q)$ be the maximum advantage of any (adaptive) CCA adversary against E subject to the adversary asking at most q total oracle queries. Similarly define $\mathbf{Adv}_E^{\text{ncpa}}(q)$ for nonadaptive CPA attacks (NCPA).

For blockciphers $F, G: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{M}$ let $F \circ G$ denote their cascade, with F 's output fed into G 's input; formally, $F \circ G: \mathcal{K}^2 \times \mathcal{M} \rightarrow \mathcal{M}$ is defined by $(F \circ G)_{(K,K')} = G_{K'}(F_K(X))$.

LIFTING NCPA TO CCA SECURITY. We bound the CCA-security of a Feistel network from its NCPA-security by using the following result of Maurer, Pietrzak, and Renner [15, Corollary 5]. It is key to our approach, effectively letting us assume that our adversaries are of the simple, NCPA breed. Recall that in writing $F \circ G$, the blockciphers are, in effect, independently keyed.

Lemma 1 (Maurer-Pietrzak-Renner). *If F and G are blockciphers on the same message space then, for any q , $\mathbf{Adv}_{F \circ G^{-1}}^{\text{cca}}(q) \leq \mathbf{Adv}_F^{\text{ncpa}}(q) + \mathbf{Adv}_G^{\text{ncpa}}(q)$.*

3 Security of Swap-or-Not

Fix a finite abelian group $G = ([N], +)$ where $[N] = \{0, 1, \dots, N - 1\}$. We define the swap-or-not shuffle $\text{SN}[r, N, +]$ of r rounds over the elements of G . The shuffling at round t is as follows. Initially, each of N distinct cards is at a position in the set $[N]$. To shuffle during this round, choose $K_t \xleftarrow{\$} [N]$, the *subkey* at round t . Then, for each set $\{X, K_t - X\}$ with $X \in G$, choose $b \xleftarrow{\$} \{0, 1\}$ and then swap the cards at positions X and $K_t - X$ if $b = 1$.

Let $\{W_t : t \geq 0\}$ be the Markov chain representing the swap-or-not shuffle with N cards. More formally, let \mathcal{C} be a set of cardinality N , whose elements we call *cards*. The state space of $\{W_t\}$ is the set of bijections from \mathcal{C} to $\{0, \dots, N-1\}$. For a card $z \in \mathcal{C}$, we interpret $W_t(z)$ as the position of card z at time t .

Let A be a deterministic adversary that makes exactly q queries. Our proof is based on an analysis of the mixing rate of the swap-or-not shuffle. However, since A makes only $q \leq N$ queries, we need only bound the rate at which some q -element subset of the cards mixes. So let z_1, \dots, z_q be distinct cards in \mathcal{C} , and let X_t be the vector of positions of cards z_1, \dots, z_q at time t . For j in $\{1, \dots, q\}$ we write $X_t(j)$ for the position of card z_j at time t , and define $X_t(1, \dots, j) = (X_t(1), \dots, X_t(j))$. We shall call X_t the *projected swap-or-not shuffle*. Note that the stationary distribution of X_t , which we denote by π , is uniform over the set of distinct q -tuples of elements from G . Equivalently, π is the distribution of q samples without replacement from G . Let τ_t denote the distribution of X_t .

Theorem 2 (Rapid mixing). *Consider the swap-or-not shuffle $\text{SN}[r, N, +]$ for $r, N \geq 1$, and let $q \in \{1, \dots, N\}$. Fix z_1, \dots, z_q and let $\{X_t : t \geq 0\}$ be the corresponding projected swap-or-not shuffle, let π be its stationary distribution, and let τ_t be the distribution of X_t . Then*

$$\|\tau_r - \pi\| \leq \frac{2N^{3/2}}{r+2} \left(\frac{q+N}{2N} \right)^{r/2+1}.$$

Proof. Let τ_t^k be the conditional distribution of X_t given the subkeys K_1, \dots, K_r . (Here we consider K_1, \dots, K_r random variables, and we condition on the σ -algebra of these random variables.) We will actually show that $\mathbf{E}(\|\tau_r^k - \pi\|)$ satisfies the claimed inequality. Note that since K_1, \dots, K_r are random variables, so is τ_r^k , and hence so is $\|\tau_r^k - \pi\|$. This implies the theorem since $\tau_r = \mathbf{E}(\tau_r^k)$ and hence

$$\|\tau_r - \pi\| = \|\mathbf{E}(\tau_r^k - \pi)\| \leq \mathbf{E}(\|\tau_r^k - \pi\|),$$

by Jensen's inequality, since for distributions μ and τ , the total variation distance $\|\mu - \tau\|$ is half the L^1 -norm of $\mu - \tau$, and the L^1 -norm is convex. For a distribution ν on q -tuples of Ω , define

$$\begin{aligned}\nu(u_1, \dots, u_j) &= \Pr[Z_1 = u_1, \dots, Z_j = u_j] \text{ and} \\ \nu(u_j \mid u_1, \dots, u_{j-1}) &= \Pr[Z_j = u_j \mid Z_1 = u_1, \dots, Z_{j-1} = u_{j-1}]\end{aligned}$$

where $(Z_1, \dots, Z_q) \sim \nu$. For example, $\tau_t(u_1, \dots, u_j)$ is the probability that, in the swap-or-not shuffle, cards z_1, \dots, z_j land in positions u_1, \dots, u_j at time t , while $\tau_t(u_j \mid u_1, \dots, u_{j-1})$ is the probability that at time t card z_j is in position u_j given that cards z_1, \dots, z_{j-1} are in positions u_1, \dots, u_{j-1} . On the other hand, $\pi(u_j \mid u_1, \dots, u_{j-1})$ is the probability that, in a uniform random ordering, card z_j is in position u_j given that cards z_1, \dots, z_{j-1} land in positions u_1, \dots, u_{j-1} .

Each of the conditional distributions $\tau_t^k(\cdot \mid u_1, \dots, u_{j-1})$ converges to uniform as $t \rightarrow \infty$. When all of these distributions are “close” to uniform, then τ_t^k will be close to π . In fact, we only need the conditional distributions to be close “on average,” as is formalized in the following lemma, which is easily established using coupling. For a proof, see [17, Appendix A].

Lemma 3. *Fix a finite nonempty set Ω and let μ and ν be probability distributions supported on q -tuples of elements of Ω , and suppose that $(Z_1, \dots, Z_q) \sim \mu$. Then*

$$\|\mu - \nu\| \leq \sum_{\ell=0}^{q-1} \mathbf{E}(\|\mu(\cdot \mid Z_1, \dots, Z_\ell) - \nu(\cdot \mid Z_1, \dots, Z_\ell)\|). \quad (2)$$

Note that in the above lemma, since Z_1, \dots, Z_q are random variables (whose joint distribution is given by μ), so is $\|\mu(\cdot \mid Z_1, \dots, Z_\ell) - \nu(\cdot \mid Z_1, \dots, Z_\ell)\|$ for every $\ell < q$; each summand in the right-hand side of (2) is the expectation of one of these random variables.

Recall that τ_t^k is the conditional distribution of X_t given K_1, \dots, K_r . Fix $\ell \in \{0, \dots, q-1\}$. We wish to bound the expected distance between the distribution $\tau_t^k(\cdot \mid X_t(1), \dots, X_t(\ell))$ and $\pi(\cdot \mid X_t(1), \dots, X_t(\ell))$ (i.e., the uniform distribution on $G \setminus \{X_t(1), \dots, X_t(\ell)\}$).

For $t \geq 0$, let $S_t = G \setminus \{X_t(1), \dots, X_t(\ell)\}$. Thus S_t is the set of positions that card $z_{\ell+1}$ could be located in at time t , given the positions of cards z_1, \dots, z_ℓ . For $a \in S_t$, let $p_t(a) = \tau_t^k(a \mid X_t(1), \dots, X_t(\ell))$. Then we have

$$\|\tau_t^k(\cdot \mid X_t(1), \dots, \ell) - \pi(\cdot \mid X_t(1), \dots, \ell)\| = \frac{1}{2} \sum_{a \in S_t} |p_t(a) - 1/m|, \quad (3)$$

where $m = |S_t| = N - \ell$. Using the Cauchy-Schwarz inequality twice gives

$$\begin{aligned} \left(\mathbf{E} \left[\sum_{a \in S_t} |p_t(a) - 1/m| \right] \right)^2 &\leq \mathbf{E} \left[\left(\sum_{a \in S_t} |p_t(a) - 1/m| \right)^2 \right] \\ &\leq m \cdot \mathbf{E} \left[\sum_{a \in S_t} (p_t(a) - 1/m)^2 \right] \\ &\leq N \cdot \mathbf{E} \left[\sum_{a \in S_t} (p_t(a) - 1/m)^2 \right]. \end{aligned} \quad (4)$$

We shall prove, by induction on t , that

$$\mathbf{E} \left[\sum_{a \in S_t} (p_t(a) - 1/m)^2 \right] \leq \left(\frac{\ell + N}{2N} \right)^t \quad (5)$$

for every $t \leq r$. Then, substituting $t = r$ to (3), (4), and (5), we have

$$\begin{aligned} &\mathbf{E} \left(\|\tau_r^k(\cdot | X_r(1, \dots, \ell)) - \pi(\cdot | X_r(1, \dots, \ell))\| \right) \\ &\leq \frac{1}{2} \left(N \cdot \mathbf{E} \left[\sum_{a \in S_r} (p_r(a) - 1/m)^2 \right] \right)^{1/2} \leq \frac{\sqrt{N}}{2} \left(\frac{\ell + N}{2N} \right)^{r/2}. \end{aligned}$$

Substituting this into Lemma 3 gives

$$\begin{aligned} \mathbf{E} \left(\|\tau_r^k - \pi\| \right) &\leq \sum_{\ell=0}^{q-1} \mathbf{E} \left(\|\tau_r^k(\cdot | X_r(1, \dots, \ell)) - \pi(\cdot | X_r(1, \dots, \ell))\| \right) \\ &\leq \sum_{\ell=0}^{q-1} \frac{\sqrt{N}}{2} \left(\frac{\ell + N}{2N} \right)^{r/2} \\ &\leq N^{3/2} \int_0^{q/2N} (1/2 + x)^{r/2} dx \leq \frac{2N^{3/2}}{r+2} \left(\frac{q+N}{2N} \right)^{r/2+1}. \end{aligned}$$

We now verify equation (5). First, consider the base case $t = 0$. Since the initial positions of the cards are deterministic,

$$\mathbf{E} \left[\sum_{a \in S_0} (p_0(a) - 1/m)^2 \right] = (1 - 1/m)^2 + (m - 1) \cdot (0 - 1/m)^2 = 1 - 1/m < 1.$$

Now suppose that equation (5) holds for t . We prove that it also holds for $t + 1$. Define $s_t = \sum_{a \in S_t} (p_t(a) - 1/m)^2$. It is sufficient to show that

$$\mathbf{E}(s_{t+1} | s_t) = \left(\frac{\ell + N}{2N} \right) s_t. \quad (6)$$

Define $f : S_t \rightarrow S_{t+1}$ by

$$f(a) = \begin{cases} a & \text{if } a \in S_{t+1}; \\ K_{t+1} - a & \text{otherwise.} \end{cases}$$

Note that f is a bijection from S_t to S_{t+1} : it sends S_t to S_{t+1} because if $a \in S_t$ then either a or $K_{t+1} - a$ must be in S_{t+1} , and it has an inverse $f^{-1} : S_{t+1} \rightarrow S_t$ defined by

$$f^{-1}(b) = \begin{cases} b & \text{if } b \in S_t; \\ K_{t+1} - b & \text{otherwise.} \end{cases}$$

Furthermore, note that

$$p_{t+1}(f(a)) = \begin{cases} p_t(a) & \text{if } K_{t+1} - a \notin S_t; \\ \frac{1}{2}p_t(a) + \frac{1}{2}p_t(K_{t+1} - a) & \text{otherwise.} \end{cases}$$

Since K_{t+1} is independent of the process up to time t , for every $y \in G$, we have $\Pr[K_{t+1} - a = y | s_t] = 1/N$. Hence, since $|S_t| = m$, conditioning on the value of $K_{t+1} - a$ gives

$$\mathbf{E}\left(\left[p_{t+1}(f(a)) - \frac{1}{m}\right]^2 | s_t\right) = \frac{\ell}{N} \left(p_t(a) - \frac{1}{m}\right)^2 + \frac{1}{N} \sum_{y \in S_t} \left[\frac{p_t(a) + p_t(y)}{2} - \frac{1}{m}\right]^2. \quad (7)$$

The sum can be rewritten as

$$\begin{aligned} & \sum_{y \in S_t} \frac{1}{4} \left[(p_t(y) - 1/m) + (p_t(a) - 1/m) \right]^2 \\ &= \frac{1}{4} \sum_{y \in S_t} (p_t(y) - 1/m)^2 + \frac{1}{2} (p_t(a) - 1/m) \sum_{y \in S_t} (p_t(y) - 1/m) + \frac{1}{4} \sum_{y \in S_t} (p_t(a) - 1/m)^2 \\ &= \frac{1}{4} s_t + \frac{m}{4} (p_t(a) - 1/m)^2, \end{aligned}$$

since $\sum_{y \in S_t} (p_t(y) - 1/m) = 0$. Combining this with (7) gives

$$\mathbf{E}\left(\left[p_{t+1}(f(a)) - 1/m\right]^2 | s_t\right) = \frac{s_t}{4N} + \frac{4\ell + m}{4N} (p_t(a) - 1/m)^2. \quad (8)$$

Note that

$$\begin{aligned} \mathbf{E}(s_{t+1} | s_t) &= \sum_{b \in S_{t+1}} \mathbf{E}\left(\left[p_{t+1}(b) - 1/m\right]^2 | s_t\right) \\ &= \sum_{a \in S_t} \mathbf{E}\left(\left[p_{t+1}(f(a)) - 1/m\right]^2 | s_t\right). \end{aligned}$$

Evaluating each term in the sum using (8) gives

$$\begin{aligned} \mathbf{E}(s_{t+1} | s_t) &= \frac{ms_t}{4N} + \frac{4\ell + m}{4N} \sum_{a \in S_t} (p_t(a) - 1/m)^2 \\ &= \frac{ms_t}{4N} + \frac{(4\ell + m)s_t}{4N} \\ &= \frac{\ell + N}{2N} s_t, \end{aligned}$$

where the last line holds because $m + \ell = N$. It follows that $\mathbf{E}(s_{t+1} \mid s_t) = \left(\frac{\ell+N}{2N}\right)s_t$, which verifies (6) and hence (5). This completes the proof. \square

CCA-SECURITY. Observe that if $E = \text{SN}[r, N, +]$ for some abelian group $G = ([N], +)$ then E^{-1} is also $\text{SN}[r, N, +]$. Employing Lemma 1 we conclude our main theorem.

Theorem 4. *Let $E = \text{SN}[2r, N, +]$. Then $\mathbf{Adv}_E^{\text{cca}}(q) \leq \frac{4N^{3/2}}{r+2} \left(\frac{q+N}{2N}\right)^{r/2+1}$.*

4 Complexity-Theoretic Interpretation

While Theorem 4 is information-theoretic, it should be clear that the result applies to the complexity-theoretic setting too, in exactly the same manner as Luby-Rackoff [14] and its successors. Namely, from a PRF $F: \mathcal{K} \times \{0,1\}^* \rightarrow \{0,1\}$ and a number n , define n -bit round functions $F_i(X)$ whose j th bit is $F(\langle i, j, n, X \rangle)$. Also define n -bit round keys K_i whose j th bit is $F(\langle i, j, n \rangle)$. Using these components, apply the swap-or-not construction for, say, $r = 7n$ rounds, yielding a PRP E on n bits. Translating the information-theoretic result into this setting, the PRP-security of E is the PRF-security of F minus a term that remains negligible until $q = (1 - \epsilon)2^n$ adversarial queries, for any $\epsilon > 0$. That is, from the asymptotic point of view, the swap-or-not construction preserves essentially all of a PRF's security in the constructed PRP.

We emphasize that our security results only cover the (strong) PRP notion of security. An interesting question we leave open is whether the swap-or-not cipher is indifferentiable from a random permutation [16]. Following Coron, Patarin, and Seurin [6], Holenstein, Künzler, and Tessaro show that the 14-round Feistel construction is indifferentiable from a random permutation [12]. But their proof is complex and delivers very poor concrete-security bounds. It would be desirable to have a construction supporting a simpler proof with better bounds.

5 Format-Preserving Encryption

In the *format-preserving encryption* (FPE) problem, one wants to encipher on an arbitrary set \mathcal{X} , often $\mathcal{X} = [N]$ for some number N . Usually constructions are sought that start from a conventional blockcipher, like AES. The problem has attracted increasing interest [1–5, 8, 9, 17, 24, 25, 27], and is the subject of ongoing standardization work by NIST and the IEEE.

When N is sufficiently small that one can afford $\tilde{\Omega}(N)$ -time to encrypt, provably good solutions are easy, by directly realizing a random shuffle [3]. And when N is sufficiently large that no adversary could ask anything near $N^{1/2}$ queries, nice solutions are again easy, using standard cryptographic constructions

like multi-round Feistel. But for intermediate-size domains, like those with 2^{30} – 2^{60} points, the bounds associated to well-known construction are disappointing, even if known attacks are not remotely feasible, and spending time proportional to the domain size, even in key-setup phase, is not attractive.

With these problematic-size domains in mind, suppose we use swap-or-not to encipher 9-digit social security numbers ($N \approx 2^{30}$). Employing Theorem 4, if we use 340 rounds we are guaranteed a maximal CCA advantage of less than 10^{-10} even if the adversary can ask $q = 10^8$ queries. Similarly, suppose we use swap-or-not to encipher 16-digit credit cards ($N \approx 2^{53}$). If we use 500 rounds we are guaranteed a maximal CCA advantage of less than 10^{-10} even if the adversary can ask $q = 10^{15}$ queries. (Of course these numbers assume random round functions; if one bases the construction on AES, say, one will have to add in a term for its insecurity.) The round counts are obviously high, yet the rounds are fast and the guarantees are strong. (We note too that, at least for the binary-string setting and AES as a starting point, there are tricks to reduce the number of blockcipher calls by a factor of five, as shown in prior work [17]. But this is probably not helpful in the presence of good AES support, as with recent Intel processors.)

A very different approach to small-domain FPE is taken by Granboulan and Pornin [9], who show how to realize a particular shuffle on N cards in $O(\lg^3 N)$ encryption time and $O(\lg N)$ space. But the method seems to be impractical, requiring extended-precision arithmetic to sample from a hypergeometric distribution. Stefanov and Shi go on to show how to exploit preprocessing to realize a different N -card shuffle [24]. Their method is applicable when the key-setup cost of $\tilde{\Theta}(N)$ is feasible, as is key storage and per-message encryption cost of $\tilde{\Theta}(N^{1/2})$. Near or beyond $N \approx 2^{30}$, these assumptions seem unlikely to hold in most settings. That said, the approach allows an adversary to query all N points, whereas the shuffle of this paper has only been proven to withstand $(1 - \epsilon)N$ queries. (We conjecture that swap-or-not works well for N queries and reasonable r —that its mixing time is fast—but no such result is proven here.)

6 Confusion/Diffusion Ciphers

Swap-or-not can also be construed as an approach for making a confusion/diffusion blockcipher. In doing this one would instantiate round functions $F_i: \{0, 1\}^n \rightarrow \{0, 1\}$ by a fast, concrete construction. Perhaps the simplest plausible instantiation is have F_i be specified by an n -bit

```
proc  $E_{KL}(X)$  //inner-product realization
for  $i \leftarrow 1$  to  $r$  do
     $X' \leftarrow K_i \oplus X$ 
     $\hat{X} \leftarrow \max(X, X')$ 
    if  $L_i \odot \hat{X} = 1$  then  $X \leftarrow X'$ 
return  $X$ 
```

Fig. 5. Cipher $E = \text{SN}[r, n, \odot]$ encrypts a string $X \in \{0, 1\}^n$ using a key KL that specifies subkeys $K_1, \dots, K_r, L_1, \dots, L_r \in \{0, 1\}^n$

string L_i , letting $F_i(\hat{X}) = L_i \odot \hat{X} = L_i[1]\hat{X}[1] \oplus \cdots \oplus L_i[n]\hat{X}[n]$ be the inner-product of L_i and \hat{X} . This concrete realization of swap-or-not is shown in Fig. 5. (We comment that for this instantiation it is necessary to use “max” instead of “min” in selecting a canonical one of $\{X, X'\}$; otherwise, we’d have $X = 0^n$ always encrypting to 0^n .)

We do not know how many rounds to suggest such that the construction of Fig. 5 should be a good blockcipher. It is incorrect to think that the theoretical analysis suggests a value like $r = 6n$; for one thing, there is an enormous gap between computing a random round function $F_i(\hat{X})$ and an inner product $L_i \odot \hat{X}$. We leave it as a problem for cryptanalysts to investigate how large r needs to be, to ascertain if inner product with L_i is actually a good choice for F_i , and to understand what other choices might work well.

Acknowledgments. The authors gratefully acknowledge comments from Mihir Bellare and Terence Spies. This work was supported under NSF grants DMS-1007739 and CNS-0904380.

References

1. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-Preserving Encryption. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 295–312. Springer, Heidelberg (2009)
2. Bellare, M., Rogaway, P., Spies, T.: The FFX mode of operation for format-preserving encryption (February 2010) (submission to NIST, available from their website)
3. Black, J., Rogaway, P.: Ciphers with Arbitrary Finite Domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (2002)
4. Brier, E., Peyrin, T., Stern, J.: BPS: a format-preserving encryption proposal (submission to NIST, available from their website)
5. Brightwell, M., Smith, H.: Using datatype-preserving encryption to enhance data warehouse security. In: 20th National Information Systems Security Conference Proceedings (NISSC), pp. 141–149 (1997)
6. Coron, J.-S., Patarin, J., Seurin, Y.: The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (2008)
7. Diaconis, P., Fill, J.: Strong stationary times via a new form of duality. Annals of Probability 18(4), 1483–1522 (1990)
8. FIPS 74. U.S. National Bureau of Standards (U.S.). Guidelines for implementing and using the NBS Data Encryption Standard. U.S. Dept. of Commerce (1981)
9. Granboulan, L., Pornin, T.: Perfect Block Ciphers with Small Blocks. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 452–465. Springer, Heidelberg (2007)
10. Halevi, S.: EME*: Extending EME to Handle Arbitrary-Length Messages with Associated Data. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 315–327. Springer, Heidelberg (2004)
11. Halevi, S., Rogaway, P.: A Tweakable Enciphering Mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
12. Holenstein, T., Künzler, R., Tessaro, S.: The equivalence of the random oracle model and the ideal cipher model, revisited. In: STOC 2011, pp. 89–98 (2011); Full version at arXiv:1011.1264

13. Hoang, V.T., Rogaway, P.: On Generalized Feistel Networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 613–630. Springer, Heidelberg (2010)
14. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. on Computing 17(2), 373–386 (1988)
15. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability Amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
16. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
17. Morris, B., Rogaway, P., Stegers, T.: How to Encipher Messages on a Small Domain: Deterministic Encryption and the Thorp Shuffle. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009)
18. Naor, M., Reingold, O.: On the construction of pseudo-random permutations: Luby-Rackoff revisited. J. of Cryptology 12(1), 29–66 (1999)
19. Naor, M., Reingold, O.: A pseudo-random encryption mode (1997) (manuscript)
20. Patarin, J.: Pseudorandom Permutations Based on the DES Scheme. In: Charpin, P., Cohen, G. (eds.) EUROCODE 1990. LNCS, vol. 514, pp. 193–204. Springer, Heidelberg (1991)
21. Patarin, J.: Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\varepsilon)}$ Security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 513–529. Springer, Heidelberg (2003)
22. Patarin, J.: Security of balanced and unbalanced Feistel schemes with linear non equalities. Cryptology ePrint report 2010/293 (2010)
23. Rudich, S.: Limits on the provable consequences of one-way functions. Ph.D. Thesis, UC Berkeley (1989)
24. Stefanov, E., Shi, E.: FastPRP: Fast pseudo-random permutations for small domains. Cryptology ePrint Report 2012/254 (2012)
25. Stütz, T., Uhl, A.: Efficient Format-Compliant Encryption of Regular Languages: Block-Based Cycle-Walking. In: De Decker, B., Schaumüller-Bichl, I. (eds.) CMS 2010. LNCS, vol. 6109, pp. 81–92. Springer, Heidelberg (2010)
26. Thorp, E.: Nonrandom shuffling with applications to the game of Faro. Journal of the American Statistical Association 68, 842–847 (1973)
27. Wen, J., Severa, M., Zeng, W., Luttrell, M., Jin, W.: Circuits and systems for video technology. IEEE Transactions on Circuits & Systems for Video Technology 12(6), 545–557 (2002)

Tweakable Blockciphers with Beyond Birthday-Bound Security

Will Landecker, Thomas Shrimpton, and R. Seth Terashima

Dept. of Computer Science, Portland State University
`{landeckw,teshrim,robert22}@cs.pdx.edu`

Abstract. Liskov, Rivest and Wagner formalized the tweakable blockcipher (TBC) primitive at CRYPTO’02. The typical recipe for instantiating a TBC is to start with a blockcipher, and then build up a construction that admits a tweak. Almost all such constructions enjoy provable security only to the birthday bound, and the one that does achieve security beyond the birthday bound (due to Minematsu) severely restricts the tweak size and requires per-invocation blockcipher rekeying.

This paper gives the first TBC construction that simultaneously allows for arbitrarily “wide” tweaks, does not rekey, and delivers provable security beyond the birthday bound. Our construction is built from a blockcipher and an ϵ -AXU₂ hash function.

As an application of the TBC primitive, LRW suggest the TBC-MAC construction (similar to CBC-MAC but chaining through the tweak), but leave open the question of its security. We close this question, both for TBC-MAC as a PRF and a MAC. Along the way, we find a nonce-based variant of TBC-MAC that has a *tight* reduction to the security of the underlying TBC, and also displays graceful security degradation when nonces are misused. This result is interesting on its own, but it also serves as an application of our new TBC construction, ultimately giving a variable input-length PRF with beyond birthday-bound security.

1 Introduction

A blockcipher $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is typically viewed as a family of permutations E_K over $\{0, 1\}^n$, where the index into the family is the key $K \in \{0, 1\}^k$. A *tweakable blockcipher* (TBC) extends this viewpoint by adding a second “dimension” to the function family, called a *tweak*. In particular, a TBC $\tilde{E}: \{0, 1\}^k \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a family of permutations indexed by a pair $(K, T) \in \{0, 1\}^k \times \mathcal{T}$. There is, however, a semantic asymmetry between the key and the tweak: the key is secret and gives rise to security, while the tweak may be public and gives rise to variability.

Liskov, Rivest and Wagner [21] formalized the TBC primitive. Their thesis was that primitives with inherent variability are a more natural starting point for building modes of operation, whereas classical constructions would use a blockcipher (deterministic once the key is fixed) and induce variability by using a per-message IV or nonce. Subsequent papers have delivered tweakable

enciphering schemes (e.g. [14–16, 32, 8] and others), message authentication codes (e.g. [28]), and authenticated encryption (e.g. [27, 28, 20]) modes of operation. The Skein [30] hash function has a TBC at its core. TBC-based constructions have found widespread practical application for full-disk encryption.

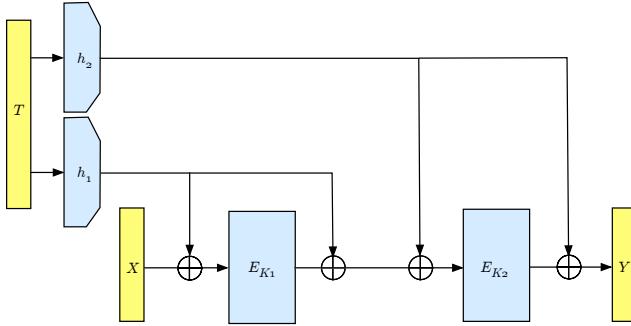
BUILDING TBCs. There are few dedicated TBC designs: the Hasty Pudding [29] and Mercy [10] ciphers natively admit tweaks. The more common approach is to start from a blockcipher and build up a TBC, incorporating support for a tweak without (one hopes) sacrificing whatever security the original blockcipher offered. The original LRW paper itself gave two constructions, which we call LRW1 and LRW2. The former construction is $\text{LRW1}[E]_K(T, X) = E_K(T \oplus E_K(X))$ and it is secure tweakable-PRP¹ if the underlying n -bit blockcipher E is a secure PRP, although there is a birthday-type loss in the reduction. (That is, the security bound becomes vacuous around $2^{n/2}$ queries.) In addition to birthday-bound security, the tweakspace is limited to $\mathcal{T} \subseteq \{0, 1\}^n$. The second LRW construction $\text{LRW2}[H, E]_{h, K}(T, X) = h(T) \oplus E_K(X \oplus h(T))$ avoids this length restriction by hashing the tweak. LRW prove that this is a tweakable strong-PRP when E is a secure strong-PRP and h is a random element of an ϵ -almost 2-xor-universal (ϵ -AXU₂) hash function family H . But here, too, one finds only birthday-bound security. Variations on the LRW constructions, for example Rogaway’s XE and XEX constructions [28], similarly offer provable security only to the birthday bound.

Tweakable blockciphers with beyond birthday-bound (BBB) security may be of particular interest for applications such as large-scale data-at-rest protection, where key management and negotiation issues seem likely to drive up the amount of data protected by a single key. Also, when legacy restrictions require the use of Triple-DES (where $n = 64$), delivering BBB security has obvious benefits. We also note that OCB mode [28] would deliver BBB authenticated-encryption security if constructed over a BBB tweakable blockcipher; other TBC-based constructions with (tight) security reductions to the security of the underlying TBC would similarly benefit.

Nonetheless, constructions of TBCs with BBB security are rare. One due to Minematsu [24] achieves BBB security, but only admits short tweaks (e.g. $\mathcal{T} = \{0, 1\}^{n-m}$ for $m \geq n/2$). It requires two blockcipher calls per TBC invocation, and suffers an additional performance penalty by rescheduling one blockcipher key whenever the tweak changes. This last point also violates a TBC design goal, that changing a tweak should more efficient than changing a key.

A NEW CONSTRUCTION WITH BBB SECURITY: CLRW2. Our main technical result is the first TBC construction that has strong tweakable-PRP security beyond the birthday bound, admits essentially arbitrary tweaks, and does not require per-invocation rekeying of any of the underlying objects.

¹ This notion is formally defined in Section 2. Informally, a TBC \tilde{E} is a secure tweakable-PRP if, for a random and secret key K , the family of mappings $\tilde{E}_K(\cdot, \cdot)$ is computationally indistinguishable from a family of random permutations. The tweakable strong-PRP notion allows for inverse queries, too.

**Fig. 1.** The CLRW2 Construction

We call this the *Chained LRW2* (CLRW2) construction, since it can be written as $\text{LRW2}[H, E]_{h_2, K_2}(T, \text{LRW2}[H, E]_{h_1, K_1}(T, X))$. The bulk of the paper is dedicated to showing that when E is a secure strong-PRP and H is an ϵ -AXU₂ hash function family with $\epsilon = 2^{-n}$, the CLRW2 TBC is a strong tweakable-PRP with security against adaptive attackers making $\mathcal{O}(2^{2n/3})$ queries. Figure 2 gives a graphical comparison of our security bound and the birthday bound.

We also consider some variations of CLRW2, for example omitting internal xors, or keying the two blockciphers with the same key.

Note that there are many efficient constructions of ϵ -AXU₂ families with $\epsilon \approx 2^{-n}$ and, except perhaps for very long tweaks, the running time of CLRW2 is likely to be dominated by the two blockcipher calls.

ANALYZING THE TBC-MAC CONSTRUCTION AND VARIANTS. In addition to formalizing the TBC primitive, LRW suggested TBC-based constructions for (authenticated) encryption, hashing and message authentication. The last of these has yet to receive formal analysis, so we consider it. The basic TBC-MAC construction operates as follows. Fix $k, n > 0$ and let $\tilde{E}: \{0, 1\}^k \times \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a tweakable blockcipher. Fix $T_0 \in \{0, 1\}^n$. Then for any key $K \in \{0, 1\}^k$ and a plaintext $M = M_1, \dots, M_b$ consisting of n -bit blocks, define $\text{TBCMAC}[\tilde{E}]_K(M) = T_b$ where $T_i \leftarrow \tilde{E}_K(T_{i-1}, M_i)$ for all $i \in [1..b]$. This is the TBC-MAC (over \tilde{E}) of the input M . It is intuitive to think of TBC-MAC as analogous to CBC-MAC. Indeed, if $\tilde{E}_K(T, X) = E_K(T \oplus X)$ then we have the CBC-MAC construction. But perhaps by abstracting away the details of \tilde{E} one can achieve better security than that offered by CBC-MAC? This seems a reasonable expectation, since an attacker can directly influence the input to the blockcipher E in CBC-MAC via the exclusive-or operation, but no such influence is guaranteed when the chaining value (the tweak) is separated from the plaintext input block. Moreover, it is easy to build TBCs with tweak inputs that are much larger than n bits (LRW already gave one way), and exploiting this may allow for simple twists on the basic TBC-MAC that give better security.

We first consider TBC-MAC as a variable-input-length pseudorandom function (VIL-PRF). We show that it is secure if the underlying TBC is secure

tweakable-PRP. Like CBC-MAC, however, TBC-MAC has only birthday-bound security. A small benefit is that this result is not restricted to prefix-free encoded inputs as it is for CBC-MAC. Actually, one can view TBC-MAC as an instance of the Merkle-Damgård iteration [23, 11] over a compression function with a dedicated key input. In this setting Bellare and Ristenpart [3] have already shown that various versions of Merkle-Damgård (plain, suffix-free encoded inputs, prefix-free encoded inputs) are PRF-preserving.

A more interesting result is found if the underlying TBC allows “wide” tweaks, i.e. tweaks that are wider than the blocksize. In this case, a simple nonce-based version of TBC-MAC (**TBCMAC2**) achieves much better PRF security bounds. In fact, if nonces are properly respected, the mode of operation imparts *no* loss over the security of the underlying TBC. Thus, **TBCMAC2** instantiated with a beyond-birthday secure TBC yields a variable-input-length PRF with beyond-birthday security. What’s more, the security bound degrades quadratically in the maximum number of times any nonce is repeated, providing more graceful behavior than most nonce-based constructions, which fail catastrophically when a nonce-repeat occurs. Such nonce misuse-resistance can be quite useful in practice.

Lastly, we show that TBC-MAC is unforgeable assuming only that the underlying TBC is likewise unforgeable. This holds only for prefix-free encoded inputs. In fact, this follows from the work of Maurer and Sjödin [22], who give general results for the Merkle-Damgård iteration. When the prefix-free encoding restriction is lifted, we exhibiting a TBC \tilde{E} that is unforgeable, yet TBC-MAC over \tilde{E} is easily forged.

UNFORGEABILITY PRESERVATION OF TBC CONSTRUCTIONS. In the full version of this work, we make another contribution to the theory of TBC constructions. We begin to explore the provable security of TBCs built from blockciphers that are assumed only to be unpredictable, rather than pseudorandom. In particular, we show that **LRW1** is *not* unforgeability preserving. That is, we build a blockcipher E that is unforgeable but for which is it easy to forge **LRW1**[E]. (In fact, we use **LRW1** against itself in this result!) Likewise for **LRW2**, we show that there is an ϵ -AXU₂ hash function family and an unforgeable blockcipher E such that **LRW2**[H, E] is easily forged. (Again, we use **LRW1** again to construct the E we need.) At this time, we do not know if **CLRW2** remains unforgeable given only unforgeable underlying blockciphers.

ADDITIONAL RELATED WORK. We have already mentioned the paper of Liskov et al. [21] as the starting point for our work. Goldenberg et al. [17] show how to build a TBC by directly tweaking the Luby-Rackoff construction. Using n -bit random functions, the resulting $2n$ -bit TBC has strong tweakable PRP security to roughly 2^n queries, and can accommodate a tweak of length ℓn using $\ell + 6$ rounds.

Coron et al. [9] show that a three-round Feistel construction over an n -bit TBC with a wide tweak yields a $2n$ -bit TBC that has beyond birthday-bound security if the underlying TBC does. Our **CLRW2** construction meets this requirement.

The PMAC1 construction by Rogaway [28] builds a (parallelizable) VIL-PRF from a TBC, achieving birthday-bound security. Recently, Yasuda [34] introduced the **PMAC_plus** construction, which has $\mathcal{O}(2^{2n/3})$ security like TBCMAC2 but is more efficient and parallelizable. **PMAC_plus** could be viewed as a construction over a tweakable blockcipher (which might be called the “XXE” construction, following Rogaway’s naming convention), but neither the construction nor the proof is cast this way. Separately, Yasuda [33] proves that Algorithm 6 from ISO 9797-1 and SUM-ECBC both have security against $\mathcal{O}(2^{2n/3})$ queries.

The WMAC construction of Black and Cochran [6] is a stateful hash-then-MAC construction that, like our TBCMAC2 construction, allows for graceful (quadratic) security degradation when nonces are repeated. There are various methods for using randomness to build VIL-PRFs with beyond birthday-bound security; for example MACRX [2], RMAC [19], randomized WMAC and enhanced hash-then-MAC [25].

We note that real-world protocols such as TLS [31] employ nonce-based PRFs by using per-message sequence numbers. Nonce-based PRFs also have applications in secure memory; see Garay et al. [18] and references therein.

Bellare and Ristenpart [3] study unforgeability preservation of iterated Merkle-Damgård constructions in the dedicated-key compression-function setting. They show that, in general, these iterations do not preserve unforgeability; however, their counterexample does not apply to TBC-MAC because the compression function they construct is not a TBC.

Zhang et al. [35] study so-called rate-1 MACs constructed from variations of the PGV [26, 7] blockcipher-based compression functions. They show that certain of these compression functions, for example $f(T, X) = E_K \oplus_T(X)$, iterate (through T) to unforgeable MACs under the assumption that the underlying blockcipher is related-key unpredictable for specific related-key functions. In the case of our example, the related-key functions are $\{K \mapsto K \oplus T \mid T \in \{0, 1\}^{|K|}\}$. But in this example and others, assuming that the blockcipher is related-key unforgeable is equivalent to assuming that the compression function is an unforgeable TBC, chaining through the tweak leads to TBC-MAC. Hence our results generalize some of those given by Zhang et al. [35]. We note that TBCs like $E_K \oplus_T(X)$ are inefficient choices for iteration through the tweak, since they require rescheduling the blockcipher key each round.

We mention in passing that the basic three-key enciphered CBC construction due to Dodis et al. [12] can, in large, part be viewed as an instance of TBC-MAC over the LWRW1 TBC. (The *IV* is no longer a fixed value, but depends on the first input block.)

2 Preliminaries

NOTATION. When \mathcal{X} is a set, we write $x \leftarrow^{\$} \mathcal{X}$ to mean that an element (named x) is uniformly sampled from \mathcal{X} . We overload the notation for probabilistic algorithms, writing $x \leftarrow^{\$} M$ to mean that algorithm M runs and outputs a value

named x . When X and Y are strings, we write $X \parallel Y$ for their concatenation. When $X \in \{0,1\}^*$ we write $|X|$ for its length. For a tuple of strings (X_1, X_2, \dots, X_r) we define $|(X_1, X_2, \dots, X_r)| = |X_1 \parallel X_2 \parallel \dots \parallel X_r|$. The set $\{0,1\}^n$ is the set of all n -bit strings, $(\{0,1\}^n)^r$ is the set of all nr -bit strings understood as r blocks of n -bits each, and $(\{0,1\}^n)^+$ is the set of all strings that are a positive number of n -bit blocks in length. When $X \in (\{0,1\}^n)^+$, we write $X_1, \dots, X_b \leftarrow X$ to mean that X is parsed into b blocks of n -bits each. For a string X of even length n , we define X_L and X_R to be $X[1..\frac{n}{2}]$ and $X[(\frac{n}{2}+1)..n]$, respectively. An *adversary* A is a probabilistic algorithm that takes zero or more oracles. We often use the notation $A \Rightarrow x$ to denote the event (defined over some specified probability space) that some algorithm A outputs value x .

We make use of the code-based game-playing framework of Bellare and Rogaway [5]. When G is a game and A an adversary, we write $\Pr[G^A \Rightarrow y]$ for the probability that the **Finalize** procedure of game G outputs y when executed with adversary A . The probability is over the coins of G and A . When the **Finalize** procedure is trivial, returning whatever A does, we omit the procedure from the game and write $\Pr[A^G \Rightarrow y]$ for the probability that A outputs y when executed with game G . In games, all boolean flags are initialized to false and all arrays are initially undefined at every point.

FUNCTION FAMILIES AND (TWEAKABLE) BLOCKCIPHERS. Let \mathcal{K} , \mathcal{D} and \mathcal{R} be sets, where at least \mathcal{K} is non-empty. A mapping $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$ can be thought of as a function family $F = \{F_K\}$ where for each $K \in \mathcal{K}$ we assign $F_K(\cdot) = F(K, \cdot)$. We will use both representations of the family, as a two-argument mapping and as a set indexed by the first argument, choosing whichever is most convenient. We write $\text{Func}(\mathcal{D}, \mathcal{R})$ for the set of all mappings from \mathcal{D} to \mathcal{R} . We write $\text{Perm}(n)$ to denote the set of all permutations (bijections) over $\{0,1\}^n$. We can view each of these as function families with some understood ordering.

A *blockcipher* is a function family $E: \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ such that for all $K \in \mathcal{K}$ the mapping $E_K(\cdot) \in \text{Perm}(n)$. We write $\text{BC}(\mathcal{K}, n)$ to mean the set of all such blockciphers, shortening to $\text{BC}(k, n)$ when $\mathcal{K} = \{0,1\}^k$. A *tweakable blockcipher* (TBC) is a function family $\tilde{E}: \mathcal{K} \times (\mathcal{T} \times \{0,1\}^n) \rightarrow \{0,1\}^n$ such that for every $K \in \mathcal{K}$ and $T \in \mathcal{T} \subseteq \{0,1\}^*$ the mapping $\tilde{E}_K(T, \cdot)$ is a permutation over $\{0,1\}^n$. The set \mathcal{T} is called the *tweakspace* of the TBC, and the element $T \in \mathcal{T}$ is the *tweak*.

SECURITY NOTIONS. Let $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$ be a function family, and let A be an adversary taking one oracle. Then we define

$$\mathbf{Adv}_F^{\text{prf}}(A) = \Pr \left[K \xleftarrow{\$} \mathcal{K} : A^{F_K(\cdot)} \Rightarrow 1 \right] - \Pr \left[\rho \xleftarrow{\$} \text{Func}(\mathcal{D}, \mathcal{R}) : A^{\rho(\cdot)} \Rightarrow 1 \right]$$

to be the PRF advantage of A attacking F . Here, and throughout, the probability is over the random choices of the described experiment and those of the adversary. We define

$$\mathbf{Adv}_F^{\text{uf-cma}}(A) = \Pr \left[K \xleftarrow{\$} \mathcal{K}; (M, \tau) \xleftarrow{\$} A^{F_K(\cdot)} : F_K(M) = \tau \wedge \text{new-msg} \right]$$

to be the UF-CMA advantage (or “forging” advantage) of A . Here the event new-msg holds iff the string M was never asked by A to its oracle.

Let $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a blockcipher, and let $\tilde{E}: \{0, 1\}^k \times (\mathcal{T} \times \{0, 1\}^n) \rightarrow \{0, 1\}^n$ be a tweakable blockcipher. Let $K \xleftarrow{\$} \{0, 1\}^k$, $\pi \xleftarrow{\$} \text{Perm}(n)$, and $\Pi \xleftarrow{\$} \text{BC}(\mathcal{T}, n)$. Then we define

$$\begin{aligned}\mathbf{Adv}_E^{\text{prp}}(A) &= \Pr \left[A^{E_K(\cdot)} \Rightarrow 1 \right] - \Pr \left[A^{\pi(\cdot)} \Rightarrow 1 \right] \\ \mathbf{Adv}_E^{\text{sprp}}(A) &= \Pr \left[A^{E_K(\cdot), E_K^{-1}(\cdot)} \Rightarrow 1 \right] - \Pr \left[A^{\pi(\cdot), \pi^{-1}(\cdot)} \Rightarrow 1 \right] \\ \mathbf{Adv}_{\tilde{E}}^{\text{prp}}(A) &= \Pr \left[A^{\tilde{E}_K(\cdot, \cdot)} \Rightarrow 1 \right] - \Pr \left[A^{\Pi(\cdot, \cdot)} \Rightarrow 1 \right] \\ \mathbf{Adv}_{\tilde{E}}^{\text{sprp}}(A) &= \Pr \left[A^{\tilde{E}_K(\cdot, \cdot), \tilde{E}_K^{-1}(\cdot, \cdot)} \Rightarrow 1 \right] - \Pr \left[A^{\Pi(\cdot, \cdot), \Pi^{-1}(\cdot, \cdot)} \Rightarrow 1 \right]\end{aligned}$$

to be (respectively) the PRP, strong PRP, tweakable-PRP, and strong tweakable-PRP advantages of A , an adversary taking the indicated number of oracles. These probabilities are over the random coins of A and the random choices of K , π , and Π , as appropriate.

A function family $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$ is ϵ -almost-XOR-universal (ϵ -AXU₂) if for all distinct $X, X' \in \mathcal{D}$ and $Y \in \mathcal{R}$, $\Pr \left[K \xleftarrow{\$} \mathcal{K} : F_K(X) \oplus F_K(X') = Y \right] \leq \epsilon$.

RESOURCES AND CONVENTIONS. We consider the following adversarial resources: the running time t , the number of oracle queries asked q , and the total length of these queries μ . For the PRP and strong PRP notions, we suppress μ since it is implicitly computable from q and the blocksize. In the UF-CMA advantage, μ includes the length of the output forgery attempt (M, τ) . It will often be the case that queries (and forgery attempts) are strings in $(\{0, 1\}^n)^+$ for some blocksize $n > 0$, and here it will be convenient to speak of the total number of blocks $\sigma = \mu/n$. The running time of an adversary is relative to some (implicit) fixed underlying model of computation. Running times will always be given with respect to some security experiment, and we define the running time to include the time to execute the entire experiment. We assume that adversaries do not make pointless queries: they do not repeat queries, nor do they ask queries that are outside of the domain of oracles they may access.

3 Tweakable SPRP-Security of CLRW2

The centerpiece of this work is a TBC construction that provides BBB security, admits a large tweakspace, and does not require rekeying of any underlying object. Given a blockcipher $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and a hash function family $H: \mathcal{K}_H \times \mathcal{D} \rightarrow \{0, 1\}^n$, the CLRW2 construction $\tilde{E}[H, E]: (\mathcal{K}_H)^2 \times (\{0, 1\}^k)^2 \times \mathcal{D} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is given by

$$\begin{aligned}\tilde{E}[H, E]_{h_1, h_2, K_1, K_2}(T, X) &= \\ E_{K_2}(E_{K_1}(X \oplus H_{h_1}(T)) \oplus H_{h_1}(T) \oplus H_{h_2}(T)) \oplus H_{h_2}(T).\end{aligned}$$

The following theorem is our main technical result.

Theorem 1. Fix $k, n > 0$ and let $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a blockcipher. Fix a non-empty set \mathcal{K}_H , and let $\mathcal{D} \subseteq \{0, 1\}^*$. Let $H: \mathcal{K}_H \times \mathcal{D} \rightarrow \{0, 1\}^n$ be an ϵ -AXU₂ function family. Let $\tilde{E} = \tilde{E}[H, E]$ be the CLRW2 construction, defined above. Let A be an adversary asking a total of q queries to its oracles, these of total length μ , and running in time t . Let $\hat{\epsilon} = \max\{\epsilon, 1/(2^n - 2q)\}$. Then there exists an adversary B using the same resources, such that.

$$\mathbf{Adv}_{\tilde{E}}^{\text{sprp}}(A) \leq 2\mathbf{Adv}_E^{\text{sprp}}(B) + \frac{6q^3\hat{\epsilon}^2}{1 - q^3\hat{\epsilon}^2}$$
■

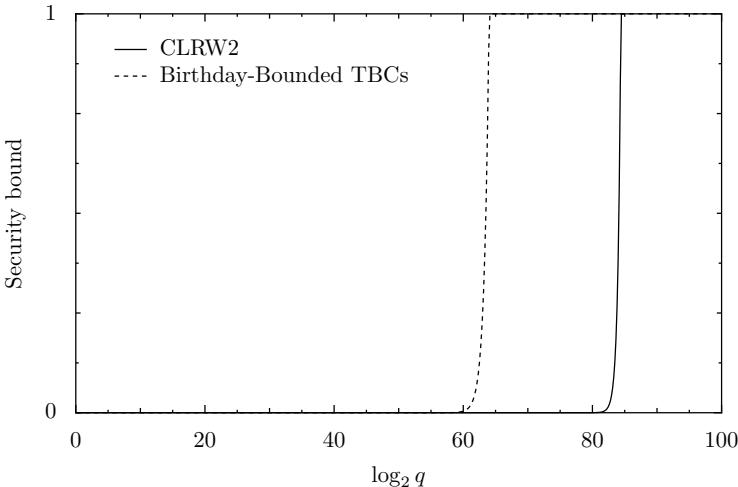


Fig. 2. The maximum advantage of an adversary making q queries against CLRW2 (solid line) and constructions limited by the birthday bound, $q^2/2^n$ (dashed line). Here, $n = 128$, $\epsilon = 2^{-n}$, and we have assumed the $\mathbf{Adv}_E^{\text{sprp}}(B)$ term is negligible.

This bound deserves some interpretation. Consider $\epsilon = 2^{-n}$ (since there are efficient constructions meeting this), and assume $q \leq 2^{n-2}$. Then $\hat{\epsilon} \leq 1/2^{n-1} \approx 2^{-n}$ for interesting values of n . The second term in the bound is at most p when $q \leq (p/(p+6))^{1/3}\hat{\epsilon}^{-2/3}$, so for any small constant p we have $q = \mathcal{O}(2^{2n/3})$. Thus when $\mathbf{Adv}_E^{\text{sprp}}(B)$ is sufficiently small, CLRW2 is secure as a tweakable-SPRP up to about $2^{2n/3}$ queries.² Figure 2 gives a graphical comparison of our bound and the standard birthday bound.

PROOF OVERVIEW. The proof of Theorem 1 is quite long and involved, so we'll start by giving a high-level overview of it. Proofs demonstrating birthday-bound

² We note that $\mathbf{Adv}_E^{\text{sprp}}(B)$ will be at least $t/2^k \approx q/2^k$ by exhaustive key search so, $q = 2^{2n/3}$ requires $k > 2n/3$, which is met by AES ($k = n = 128$) and DES ($k = 56, n = 64$).

security for TBC constructions typically “give up” if the adversary can cause a collision at a blockcipher input. In constructions like LRW1 and LRW2, the TBC output is “no longer random”, even when the blockcipher has been replaced by a random permutation. We overcome this problem by using two rounds of LRW2, and showing that it takes two independent collisions *on the same query* to force non-random CLRW2 outputs.

The chief difficulty is ensuring that the second LRW2 round can withstand a collision so long as there was not also one on the first round. To this end, we argue that given a collision-free first round, the resulting distribution of CLRW2 output values—including those which *require* a second-round collision to obtain—is extremely close to that of an ideal TBC.

The bulk of the proof is a sequence of games bounding the success probability of an adversary in the information-theoretic setting, where the blockciphers have been replaced by random permutations. The first three games address first-round collisions, and show that the distribution of CLRW2 outputs is consistent with that of an ideal cipher unless there is simultaneous a second-round collision. Our next three games address the case in which there is no first-round collision. By swapping the order in which dependent random variables are assigned values, we can choose the output early on in the game, and gain insight into the distribution by which it is governed. This distribution is shown to be very close to the ideal one. The final two games are used to derive an upper bound for the probability that the adversary can set a “bad flag”, which would force the game to exhibit non-ideal behavior. In the end, we are able to assume that the adversary is non-adaptive by giving it explicit control over oracle return values. At that point, the ϵ -AXU₂ property can be applied.

Proof. For notational simplicity, we write h_1 for H_{h_1} , and h_2 for H_{h_2} ; this should cause no confusion. The majority of the proof will consider the construction \tilde{E} with E_{K_1} and E_{K_2} replaced with random permutations π_1 and π_2 , which we write as $\tilde{E}_{h_1, h_2, \pi_1, \pi_2}$. At the end we can make a standard move to lift to the fully complexity theoretic setting.

Let A be an adversary making q queries. If the i^{th} query is to the left (encryption) oracle, we denote the query with (T_i, X_i) and the response with Y_i ; if the query is to the right (decryption) oracle, the roles of X_i and Y_i are reversed. We make the standard simplifying assumption that A makes no redundant queries by, for example, giving its encryption oracle the result of a decryption oracle query with the same tweak. We denote by \mathcal{Y}_i the set of permissible (tweak-respecting) return values for an encryption oracle query, and similarly, \mathcal{X}_i is the set of permissible return values for a decryption oracle query. That is,

$$\begin{aligned}\mathcal{Y}_i &= \{0, 1\}^n \setminus \{Y_j : j < i, T_j = T_i\} \\ \mathcal{X}_i &= \{0, 1\}^n \setminus \{X_j : j < i, T_j = T_i\}.\end{aligned}$$

An encryption oracle simulating an ideal cipher $\Pi \xleftarrow{\$} \text{BC}(\mathcal{T}, n)$ would sample its return value Y_i from \mathcal{Y}_i .

The bulk of this proof concerns showing that a sequence of games are identical, or are identical until a specified event occurs (a boolean variable is set to `true`). Due to space limitations, we will simply describe game transitions on a high level and refer readers interested in a more rigorous argument to the full version of this paper. The permutations π_1 and π_2 are constructed lazily, while h_1 and h_2 are already defined. Initially, boolean variables have the value `false`.

Note that \tilde{E} is the dual of E^{-1} , in the sense that $\tilde{E}_{h_1, h_2, \pi_1, \pi_2}^{-1}(Y, T) = \tilde{E}_{h_2, h_1, \pi_2^{-1}, \pi_1^{-1}}(Y, T)$. When arguing that transitions between games are correct, we will exploit this duality by limiting our discussion to changes in the encryption oracle, and hence to queries made to that oracle; the arguments used to justify the corresponding changes in the decryption oracle are practically identical. Therefore fix some value $i \in [1..q]$, and assume the i^{th} query is to the encryption oracle.

Game $G1$ simulates \tilde{E} by defining π_1 and π_2 through lazy sampling, so

$$\Pr[A^{\tilde{E}, \tilde{E}^{-1}} \Rightarrow 1] = \Pr[A^{G1} \Rightarrow 1].$$

In Game $G2$, we change what happens when there is a collision at the first blockcipher: we sample $Y_i \xleftarrow{\$} \mathcal{Y}_i$, but raise a bad flag if we also encounter a collision at the input of second blockcipher (`bad1`) or if $Y_i \oplus h_2(T_i)$ is already in its range (`bad2`). Should either of these events occur, we fall back to the lazy-sampling method of Game $G1$ to choose a new value for Y_i . Game $G3$ is identical to Game $G2$, except Y_i is not reassigned after a `bad` flag is set. Hence

$$\Pr[A^{G1} \Rightarrow 1] = \Pr[A^{G2} \Rightarrow 1] \leq \Pr[A^{G3} \Rightarrow 1] + \Pr[A^{G3} : \text{bad1} \vee \text{bad2}].$$

Next we modify the section of code in Game $G3$ that is executed when no collision occurs at π_1 ; i.e., when $X_i \oplus h_1(T_i) \neq X_j \oplus h_1(T_j)$ for all $j < i$. The behavior of the encryption oracle in this game during the i^{th} query can be completely described by the pair (P_i, Q_i) , where P_i is the output of π_1 and Q_i is the output of π_2 . The oracle's output, Y_i , is uniquely determined by Q_i , since $Y_i = Q_i \oplus h_2(T_i)$. Hence, we treat the pair (P_i, Q_i) as a single random variable; any method of assigning it a value that respects the joint distribution on P_i and Q_i preserves the black-box behavior of Game $G3$'s oracles.

Fix a query (X_i, T_i) . Suppose no collision occurs at π_1 (i.e., $X_i \oplus h_1(T_i) \neq X_j \oplus h_1(T_j)$ for all $j < i$). Call $s \in \{0, 1\}^n$ *possible* if $\Pr[Q_i = s] > 0$. A possible s is *fresh* when $s \neq Q_j$ for all $j < i$ (i.e., s is not yet in the range of π_2), and *stale* otherwise. The correspondence $Y_i = Q_i \oplus h_2(T_i)$ between Y_i and Q_i allows us to describe the distribution governing Y_i in terms of Q_i . To do so, we first define:

$$\begin{aligned} S_1 &= \{y \in \{0, 1\}^n : y \oplus h_2(T_i) \text{ is fresh}\} \\ S_2 &= \{y \in \{0, 1\}^n : y \oplus h_2(T_i) \text{ is stale}\} \\ S_3 &= \{y \in \mathcal{Y}_i : y \oplus h_2(T_i) \text{ is not possible}\} \\ S_4 &= \{y \in \overline{\mathcal{Y}_i} : y \oplus h_2(T_i) \text{ is not possible}\} \end{aligned}$$

(Hence $\{0, 1\}^n = S_1 \cup S_2 \cup S_3 \cup S_4$). One can show that for fixed $k = 1, 2, 3, 4$, any two values in S_k are equally likely to be returned by the Game $G3$'s encryption oracle on the i^{th} query. With a little effort, one can compute these probabilities in terms of $N = |\{p : \Pr [P_i = p] > 0\}|$ —the number of values not in π_1 's range—and each $|S_k|$. As Figure 3 shows, the resulting distribution is very close to the one an ideal cipher would provide.

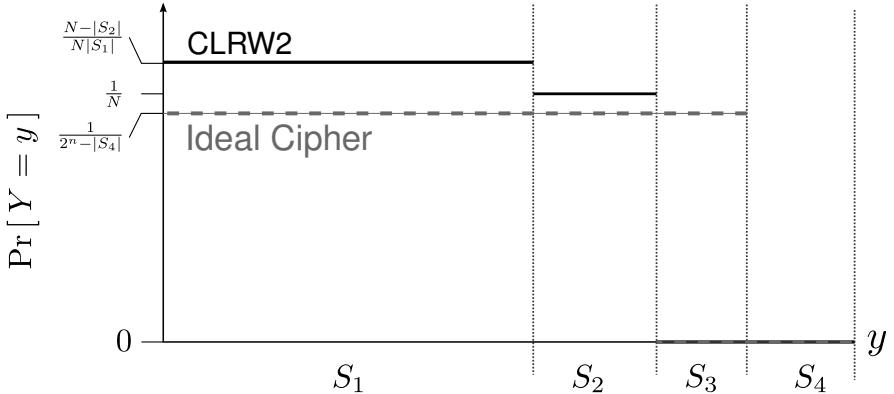


Fig. 3. When there is no collision at π_1 , the distribution governing \tilde{E} 's outputs is very close to the distribution an ideal cipher would provide. Horizontal scaling suggests plausible relative sizes of each $|S_k|$: likely $|S_1| \gg |S_2 \cup S_4| \gg |S_3|$. This graph is accurate for the oracles in Games 1–5.

The two distributions vary with each query (and in particular, with each tweak change), and an adaptive adversary could conceivably make the difference between them significant after a large number of queries. Intuitively, S_3 is the problematic set, and indeed, $\Pr [Y_i \leftarrow \mathcal{Y}_i ; Y_i \in S_3]$ is the statistical distance between the two distributions. The next few games reveal this more explicitly by “bubbling up” Y_i 's assignment in the pseudocode, then linking the two distributions together by setting a bad flag if $Y_i \leftarrow \mathcal{Y}_i$ gives $Y_i \in S_3$.

In Game $G4$, we flip a weighted coin to determine if Q_i is fresh or stale, and then choose (P_i, Q_i) after conditioning on the outcome.

Next, instead of using the coin flip to determine Q_i (or equivalently, Y_i), Game $G5$ samples $Y_i \leftarrow \mathcal{Y}_i$ to determine Q_i —and hence how the coin landed. Again, ultimately all that matters is the distribution on the joint random variable (P_i, Q_i) . Since $Q_i = Y_i \oplus h_2(T_i)$ may be neither fresh nor stale (i.e., not possible), Game $G5$ falls back to the technique of Game $G4$ when $Y_i \in S_3$, after setting bad_3 . This overwrites the value assigned to Y_i .

Finally, in Game $G6$, we do not do anything special after setting bad_3 ; we simply keep the value of Y_i we originally sampled from \mathcal{Y}_i .

Game $G7$ simplifies some of Game $G6$'s flow. In particular, Y_i is now always sampled from \mathcal{Y}_i (regardless of whether or not there is a collision at π_1), and its value is never overwritten; consequently, the line $Y_i \leftarrow \mathcal{Y}_i$ is moved to the start of the encryption oracle pseudocode. Since a family of random permutations would return Y_i from this precise distribution, Game $G7$ simulates an ideal cipher:

$$\Pr \left[\Pi \stackrel{\$}{\leftarrow} \text{BC}(\mathcal{T}, n) ; A^{\Pi, \Pi^{-1}} \Rightarrow 1 \right] = \Pr \left[A^{G7} \Rightarrow 1 \right].$$

As consequences of the Fundamental Lemma of Game Playing,

$$\begin{aligned} \Pr \left[A^{G3} \Rightarrow 1 \right] &= \Pr \left[A^{G4} \Rightarrow 1 \right] = \Pr \left[A^{G5} \Rightarrow 1 \right] \\ &\leq \Pr \left[A^{G6} \Rightarrow 1 \right] + \Pr \left[A^{G6} : \text{bad}_3 \right] \\ &= \Pr \left[A^{G7} \Rightarrow 1 \right] + \Pr \left[A^{G7} : \text{bad}_3 \right], \end{aligned}$$

and similarly

$$\Pr \left[A^{G3} ; \text{bad}_1 \vee \text{bad}_2 \right] \leq \Pr \left[A^{G7} ; \text{bad}_1 \vee \text{bad}_2 \right] + \Pr \left[A^{G7} ; \text{bad}_3 \right].$$

Using a standard hybrid argument, it follows that there exists an SPRP adversary B making q queries and running in time $O(t)$ such that

$$\mathbf{Adv}_{\tilde{E}}^{\text{sprp}}(A) \leq 2\mathbf{Adv}_E^{\text{sprp}}(B) + \Pr \left[A^{G7} ; \text{bad}_1 \vee \text{bad}_2 \right] + 2\Pr \left[A^{G7} ; \text{bad}_3 \right].$$

We wish to find an upper bound for the probabilities in this expression. Predictably, the difficulty here is that A is adaptive, and hence its queries are not independent of, for example, h_1 . In Game $G8$, we give the adversary control over what value is assigned to Y_i (or X_i , in the case of decryption queries), but insist that it be in \mathcal{Y}_i or \mathcal{X}_i , as appropriate. Because this new adversary \tilde{A} can compute \mathcal{Y}_i and \mathcal{X}_i , he may simulate the oracles of Game $G7$ if desired; hence, given q queries, \tilde{A} can set the **bad** flags in Game $G8$ with probability at least as high as any A can set the corresponding flags in Game $G7$. The oracle's outputs are now deterministic, and may be (trivially) computed by the adversary in advance. Hence, we may assume without loss of generality that \tilde{A} is non-adaptive.

One can show that in order to set bad_m ($m = 1, 2, 3$), \tilde{A} must make a sequence of queries such that there exist $i, j, k \leq q$ such that $j, k \neq i$, $X_i \oplus h_1(T_i) = X_j \oplus h_1(T_j)$, and either

1. $Y_i \oplus h_2(T_i) = Y_k \oplus h_2(T_k)$ or
2. $\pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i) = \pi_1(L_k) \oplus h_1(T_k) \oplus h_2(T_k)$,

where $L_i = X_i \oplus h_1(T_i)$, and similarly for L_k . Either case requires query i to “collide” with independently with two other queries in some fashion. The ϵ -AXU₂ property makes the first type of collision unlikely, and the fact that in this game, $\pi_1(L_i)$ will be sampled from a set of size at least $1/(2^n - 2q)$ makes it unlikely that $\pi_1(L_i)$ will be assigned the unique value that causes the second type of collision to occur (for given i, j , and k).

Let $\hat{\epsilon} = \max(\epsilon, 1/(2^n - 2q))$. By carefully computing these upper bounds for the probabilities of these collisions and taking a union bound over all permissible (i, j, k) pairs, one can show that

$$\mathbf{Adv}_{\tilde{E}}^{\text{sprp}}(A) \leq 2\mathbf{Adv}_E^{\text{sprp}}(B) + \frac{6q^3\hat{\epsilon}^2}{1-q^3\hat{\epsilon}^2},$$

completing the proof. \square

ATTACKS ON SIMPLER VARIANTS. Having seen our construction, one wonders if simpler variants work. For example, consider CLRW2 without the first $H_{h_2}(T)$ XOR operation, leaving

$$\tilde{E}_{h_1, h_2, K_1, K_2}(T, X) = H_{h_2}(T) \oplus E_{K_2}(H_{h_1}(T) \oplus E_{K_1}(H_{h_1}(T) \oplus X)).$$

This variation permits birthday-bound attack. Namely, an adversary could submit queries in pairs, (T_i, X') and (T_i, X'') , where X' and X'' are fixed, and a new random tweak is used for each pair. By remembering $\tilde{E}(T_i, X') \oplus \tilde{E}(T_i, X'')$ values, which are independent of H_{h_2} , it could detect collisions in H_{h_1} , say by using a hash table. That is, if $H_{h_1}(T_i) = H_{h_1}(T_j)$, then $\tilde{E}(T_i, X') \oplus \tilde{E}(T_i, X'') = \tilde{E}(T_j, X') \oplus \tilde{E}(T_j, X'')$. The converse is false, but false positives could be weeded out by testing a small number of X -values. Such an adversary would gain advantage close to one. Similar variations on \tilde{E} permit analogous attacks, though we believe (but do not prove) that omitting the second $H_{h_1}(T)$ XOR operation yields a construction secure against adversaries constrained to chosen-plaintext attacks.

One might also wish to try setting $K_2 = K_1$. While we know of no attacks here, modifying our proof to accommodate this change would be non-trivial. In particular, bounding certain probabilities required us to trace back through a game's execution history to determine when π_1 became defined at particular points (L_i or L_k in the above proof); this task would be messier and more difficult to verify if $\pi_2 = \pi_1$. Still, this may merit future investigation.

4 PRF-Security of TBC-MAC

THE TBC-MAC FUNCTION FAMILY. Fix $k, n > 0$ and let $\tilde{E}: \{0, 1\}^k \times \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a tweakable blockcipher. We define the TBC-MAC function family $\text{TBCMAC}[\tilde{E}]: \{0, 1\}^k \times (\{0, 1\}^n)^+ \rightarrow \{0, 1\}^n$ as follows. On input $K \in \{0, 1\}^k$ and $M \in (\{0, 1\}^n)^+$, let $\text{TBCMAC}[\tilde{E}]_K(T, M) = T_b$ where $T_0 = 0^n$; let $M_1, \dots, M_b \leftarrow M$, and $T_i \leftarrow \tilde{E}_K(T_{i-1}, M_i)$ for $i \in \{1, \dots, b\}$. To extend the domain to $\{0, 1\}^*$, one could introduce an explicit, unambiguous padding rule mapping $\{0, 1\}^* \rightarrow (\{0, 1\}^n)^+$, say mapping $M \mapsto M \| 10^r$ where r is the smallest integer needed to reach a block boundary. But for simplicity we assume that all strings input to $\text{TBCMAC}[\tilde{E}]$ are block-aligned. We extend this assumption

by writing $\text{TBCMAC}^{\text{pf}}$ for the TBC-MAC construction restricted to prefix-free encoded, block-aligned inputs.

BUILDING FROM A “NARROW” TWEAKSIZE TBC. Our first result in this section is a natural one. We prove that TBC-MAC is a secure PRF if the underlying TBC \tilde{E} , with n -bit tweaks and blocksize, is secure as a tweakable-PRP. One might hope that the security bound for $\text{TBCMAC}[\tilde{E}]$ is better than for CBC-MAC over an n -bit blockcipher, since the former is intuitively a “stronger” object than the latter. This is not the case. This is because the IV is fixed; thus an adversary can ask a series of distinct one-block messages and wait for a collision. Considering the information-theoretic setting, the fixed IV effectively reduces the ideal cipher to a random permutation in the first round, and so the standard PRP-PRF distinguishing attack forces us to accept birthday-bound security. The following theorem closely follows the code-based game-playing proof of CBC-MAC due to Bellare and Rogaway [5]. We note that a tighter bound could be achieved (with more work) following the techniques of Bellare et al. [4]. The proof appears in the full version.

Theorem 2. (TBCMAC is a PRF.) Fix $n > 0$. Let $\tilde{E}: \{0, 1\}^n \times (\{0, 1\}^n \times \{0, 1\}^n) \rightarrow \{0, 1\}^n$ be a tweakable blockcipher. Let A be an adversary running in time t , asking q queries, each of length at most ℓ blocks of n -bits. Then

$$\mathbf{Adv}_{\text{TBCMAC}[\tilde{E}]}^{\text{prf}}(A) \leq \mathbf{Adv}_{\tilde{E}}^{\widetilde{\text{prp}}}(B) + \frac{(q\ell)^2}{2^n}$$

for an adversary B that runs in time $t' = t + \mathcal{O}(\ell q)$ and asks at most $q' = q\ell$ queries. ■

BUILDING FROM A “WIDE” TWEAKSIZE TBC. The LRW2 and CLRW2 constructions each give TBC that can handle tweaks that are potentially much larger than the blocksize. So we now consider the security of a nonce-based version of TBC-MAC based upon such a TBC. In particular, fix $k, n, b > 0$ and let $\tilde{E}: \{0, 1\}^k \times (\{0, 1\}^{n+b+1} \times \{0, 1\}^n) \rightarrow \{0, 1\}^n$ be a tweakable blockcipher with tweaksize $n + b + 1$ bits and blocksize n bits. For an ℓ -block message M_1, \dots, M_ℓ where $\ell > 1$, nonce $N \in \{0, 1\}^b$, and a fixed $T_0 = IV$, define $\text{TBCMAC2}[\tilde{E}]_K(N, M)$ as $T_\ell = \tilde{E}_K(T_{\ell-1} \parallel 1 \parallel N, M_\ell)$ where for $i = 1$ to $\ell - 1$, $T_i = \tilde{E}_K(T_{i-1} \parallel 0 \parallel 0^b, M_i)$. We say that a PRF-adversary A is *nonce-respecting* (for TBCMAC2) if it never repeats a nonce. The *multiplicity* α of a nonce N is the number of times it is used in an attack, e.g. $\alpha = 1$ for every nonce if the attack is nonce-respecting.

Theorem 3. (TBCMAC2 is a PRF.) Fix $n > 0$ and $b \geq 0$. Let $\tilde{E}: \{0, 1\}^n \times (\{0, 1\}^{n+b+1} \times \{0, 1\}^n) \rightarrow \{0, 1\}^n$ be a tweakable blockcipher. Let $\text{TBCMAC2}[\tilde{E}]$ be as described above. Let A be an adversary that runs in time t , asks q queries for the form (N, M) where the length of M is at most ℓ blocks. Assume that there are r distinct values of N among these queries, and let $\alpha_1, \dots, \alpha_r$ denote the multiplicities of these. Then

$$\mathbf{Adv}_{\text{TBCMAC2}[\tilde{E}]}^{\text{prf}}(A) \leq \mathbf{Adv}_{\tilde{E}}^{\text{ppr}}(B) + \frac{1}{2^{n+1}} \left(\sum_{i=1}^r \alpha_i(\alpha_i - 1) \right) + \sum_{i=1}^r \binom{\alpha_i}{2} \frac{(2\ell + 1)(2\ell)}{2^n}$$

where B runs in time $t' = t + \mathcal{O}(q\ell)$ and asks at most $q' = q\ell$ queries. Specifically, if A is nonce-respecting, $\mathbf{Adv}_{\text{TBCMAC2}[\tilde{E}]}^{\text{prf}}(A) \leq \mathbf{Adv}_{\tilde{E}}^{\text{ppr}}(B)$. ■

The proof of Theorem 3 appears in the full version.

5 Unforgeability-Preservation of TBC-MAC

TBC-MAC preserves the unforgeability of its underlying TBC when the TBC-MAC inputs are prefix-free. Since, qualitatively, this amounts to a new application of an existing result by Maurer and Sjödin [22], we defer our proof until the full version.

Theorem 4. ($\text{TBCMAC}^{\text{pf}}$ preserves UF-CMA.) Fix $k, n > 0$, and let $\tilde{E} : \{0, 1\}^k \times \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a TBC. Let A be an adversary for $\text{TBCMAC}^{\text{pf}}[\tilde{E}]$ that runs in time t , asks q queries, these totaling σ blocks of n -bits in length. Then there exist adversaries B and C such that

$$\mathbf{Adv}_{\text{TBCMAC}^{\text{pf}}[\tilde{E}]}^{\text{uf-cma}}(A) \leq \frac{\sigma(\sigma - 1)}{2} \mathbf{Adv}_{\tilde{E}}^{\text{uf-cma}}(B) + \mathbf{Adv}_{\tilde{E}}^{\text{uf-cma}}(C)$$

where B runs in time $t_B \leq t$, asks $q_B \leq \sigma$ queries totalling $\sigma_B \leq 2\sigma$ blocks; and where C runs in time $t_C = t$, asks $q_C = \sigma$ queries totalling $\sigma_C = 2\sigma$ blocks. ■

However, if adversaries may mount an attack using non-prefix-free inputs, it is possible to forge TBC-MAC.³ The following lemma says that there exists a TBC \tilde{F} that is unforgeable if some underlying TBC \tilde{E} is. Liskov et al. [21] provide a TBC \tilde{E} with the required signature. The proof appears in the full version.

Lemma 1. Let $\tilde{E} : \{0, 1\}^k \times \{0, 1\}^{3n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a tweakable blockcipher. Let $\tilde{F} : \{0, 1\}^k \times \{0, 1\}^{2n} \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ be a tweakable blockcipher defined by $\tilde{F}_K(T_L \parallel T_R, X_L \parallel X_R) = X_L \oplus T_R \parallel \tilde{E}_K(X_L \parallel T_L \parallel T_R, X_R)$. Then $\mathbf{Adv}_{\tilde{F}}^{\text{uf-cma}}(A) \leq \mathbf{Adv}_{\tilde{E}}^{\text{uf-cma}}(B)$ where the resources of adversaries A and B are the same. ■

We now show that TBC-MAC instantiated with \tilde{F} admits efficient forging attacks if arbitrary inputs are allowed.

Theorem 5. (TBCMAC is not UF-CMA preserving.) Let \tilde{E} be a tweakable blockcipher and let \tilde{F} be as defined in Lemma 1. Then there exists an adversary A that asks $q = 2$ queries totalling $\sigma = 12$ blocks of n -bits such that $\mathbf{Adv}_{\text{TBCMAC}[\tilde{F}]}^{\text{uf-cma}}(A) = 1$. ■

³ We note that Bellare and Ristenpart [3] have already shown that the Merkle-Damgård iteration is not unforgeability preserving for arbitrary inputs. However, their counterexample does not suffice here, because the compression function they build is not a TBC.

Proof. Consider the adversary A that queries $Y^1 \leftarrow \text{TBCMAC}[\tilde{F}]_K(0^{2n} \parallel 0^{2n})$, and then forges with $X^* = 0^{2n}$ and $Y^* = 0^n \parallel Y_L^1$. The forgery is valid; we leave the confirmation of this fact to the interested reader. \square

Acknowledgments. The authors of this work were supported by NSF grants CNS 0627752 and CNS 0845610.

References

1. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
2. Bellare, M., Goldreich, O., Krawczyk, H.: Stateless Evaluation of Pseudorandom Functions: Security beyond the Birthday Barrier. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 270–287. Springer, Heidelberg (1999)
3. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
4. Bellare, M., Pietrzak, K., Rogaway, P.: Improved Security Analyses for CBC MACs. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 527–545. Springer, Heidelberg (2005)
5. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
6. Black, J., Cochran, M.: MAC Reforgeability. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 345–362. Springer, Heidelberg (2009)
7. Black, J., Rogaway, P., Shrimpton, T., Stam, M.: An analysis of the blockcipher-based hash functions from PGV. Journal of Cryptology 23(4), 320–325 (2010)
8. Chakraborty, D., Sarkar, P.: A New Mode of Encryption Providing a Tweakable Strong Pseudo-random Permutation. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 293–309. Springer, Heidelberg (2006)
9. Coron, J.-S., Dodis, Y., Mandal, A., Seurin, Y.: A Domain Extender for the Ideal Cipher. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 273–289. Springer, Heidelberg (2010)
10. Crowley, P.: Mercy: A Fast Large Block Cipher for Disk Sector Encryption. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 49–63. Springer, Heidelberg (2001)
11. Damgård, I.B.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
12. Dodis, Y., Pietrzak, K., Puniya, P.: A New Mode of Operation for Block Ciphers and Length-Preserving MACs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 198–219. Springer, Heidelberg (2008)
13. Dodis, Y., Steinberger, J.: Message Authentication Codes from Unpredictable Block Ciphers. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 267–285. Springer, Heidelberg (2009)
14. Halevi, S., Rogaway, P.: A Tweakable Enciphering Mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
15. Halevi, S., Rogaway, P.: A Parallelizable Enciphering Mode. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 292–304. Springer, Heidelberg (2004)
16. Halevi, S.: Invertible Universal Hashing and the TET Encryption Mode. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 412–429. Springer, Heidelberg (2007)

17. Goldenberg, D., Hohenberger, S., Liskov, M., Schwartz, E.C., Seyalioglu, H.: On Tweaking Luby-Rackoff Blockciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 342–356. Springer, Heidelberg (2007)
18. Garay, J., Kolesnikov, V., McLellan, R.: MAC Precomputation with Applications to Secure Memory. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 427–442. Springer, Heidelberg (2009)
19. Jaulmes, É., Joux, A., Valette, F.: On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 237–251. Springer, Heidelberg (2002)
20. Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 306–327. Springer, Heidelberg (2011)
21. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
22. Maurer, U., Sjödin, J.: Single-Key AIL-MACs from Any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
23. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
24. Minematsu, K.: Beyond-Birthday-Bound Security Based on Tweakable Block Cipher. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 308–326. Springer, Heidelberg (2009)
25. Minematsu, K.: How to Thwart Birthday Attacks against MACs via Small Randomness. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 230–249. Springer, Heidelberg (2010)
26. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
27. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: ACM Conference on Computer and Communication Security – CCS 2001, pp. 196–205. ACM Press (2001)
28. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
29. Schroeppel, R.: The hasty pudding cipher. NIST AES proposal (1998), <http://www.cs.arizona.edu/~rcs/hpc>
30. Bellare, M., Kohno, T., Lucks, S., Ferguson, N., Schneier, B., Whiting, D., Callas, J., Walker, J.: Provable Security Support for the Skein Hash Family, <http://www.skein-hash.info/sites/default/files/skein-proofs.pdf>
31. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. Internet RFC 4346 (2006)
32. Wang, P., Feng, D., Wu, W.: HCTR: A Variable-Input-Length Enciphering Mode. In: Feng, D., Lin, D., Yung, M. (eds.) CISC 2005. LNCS, vol. 3822, pp. 175–188. Springer, Heidelberg (2005)
33. Yasuda, K.: The Sum of CBC MACs Is a Secure PRF. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 366–381. Springer, Heidelberg (2010)
34. Yasuda, K.: A New Variant of PMAC: Beyond the Birthday Bound. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 596–609. Springer, Heidelberg (2011)
35. Zhang, L., Wu, W., Wang, P., Zhang, L., Wu, S., Liang, B.: Constructing Rate-1 MACs from Related-Key Unpredictable Block Ciphers: PGV Model Revisited. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 250–269. Springer, Heidelberg (2010)

Breaking and Repairing GCM Security Proofs

Tetsu Iwata¹, Keisuke Ohashi¹, and Kazuhiko Minematsu²

¹ Nagoya University, Japan

iwata@cse.nagoya-u.ac.jp, k_oohashi@echo.nuee.nagoya-u.ac.jp

² NEC Corporation, Japan

k-minematsu@ah.jp.nec.com

Abstract. In this paper, we study the security proofs of GCM (Galois/Counter Mode of Operation). We first point out that a lemma, which is related to the upper bound on the probability of a counter collision, is invalid. Both the original privacy and authenticity proofs by the designers are based on the lemma. We further show that the observation can be translated into a distinguishing attack that invalidates the main part of the privacy proof. It turns out that the original security proofs of GCM contain a flaw, and hence the claimed security bounds are not justified. A very natural question is then whether the proofs can be repaired. We give an affirmative answer to the question by presenting new security bounds, both for privacy and authenticity. As a result, although the security bounds are larger than what were previously claimed, GCM maintains its provable security. We also show that, when the nonce length is restricted to 96 bits, GCM has better security bounds than a general case of variable length nonces.

Keywords: GCM, counter-example, distinguishing attack, proof of security.

1 Introduction

GCM (Galois/Counter Mode of Operation) is the authenticated encryption mode of blockciphers designed by McGrew and Viega [26,27]. The mode is based on the counter mode encryption and the polynomial hash function, and the designers presented proofs of security both for privacy and authenticity [26,27]. It was selected as the NIST recommended blockcipher mode in 2007 [15], and is widely used in practice, e.g., in [1,2,4,5,6,7,8,14,17,19,20,25,33,34].

The security of GCM has been extensively evaluated. Ferguson pointed out that a forgery is possible if the tag length is short [16]. Joux showed that a part of the secret key can be obtained if the nonce is reused [21]. Handschuh and Prenneel discussed weak keys of GCM and presented generalizations of Joux's attack [18]. Saarinen pointed out that GCM has more weak keys than previously known, and used the weak keys for forgery attacks [32]. See also [31] for comprehensive discussions on various aspects on GCM.

Despite aforementioned attacks, it is widely considered that the provable security results of GCM are sound, in the sense that the previous attacks do not

contradict the claimed security bounds by the designers, and that no flaw in the proofs has been identified. Some of these attacks show the tightness of the security bounds, and others are outside the security model (e.g., nonce reuse). Therefore, there is no attack that undermines the security bounds or their proofs.

GCM uses the counter mode encryption, and the initial counter value is derived from a nonce, where there are two different ways to generate the initial counter value depending on the length of the nonce. When the nonce length is 96 bits, the initial counter value is the nonce padded with a constant. When the nonce length is not 96 bits, the polynomial hash function is applied to the nonce to obtain the initial counter value. In order to prove the security of GCM, one has to show that the probability of a counter collision is small. McGrew and Viega presented a lemma showing the upper bound on the probability in [27], which is the basis for both the privacy and authenticity proofs.

In this paper, we first point out that the claimed lemma cannot be valid; the probability of a counter collision is larger than claimed. We show concrete counter-examples of two distinct nonces that invalidate the lemma. It turns out that the original security proofs (both for privacy and authenticity) of GCM contain a flaw, and hence the claimed security bounds are not justified.

We next translate the above observation into a distinguishing attack. The attack is simple and efficient. However, from the practical perspective, the success probability of the attack is insignificantly small, and it does not contradict the security bounds by the designers. On the other hand, the success probability is large enough to invalidate the main part of the privacy proof. In more detail, there are three terms in the privacy bound of GCM. The first one comes from the difference between a random permutation and a random function, the second one is the main part of the privacy proof that bounds the distinguishing probability of ciphertexts of GCM based on a random function from random strings (of the same lengths as the ciphertexts), and the last one bounds the forgery probability. The success probability of our distinguishing attack is larger than the second term, invalidating the main part of the privacy proof. Consequently, the security of GCM is not supported by the proofs.

Then a very natural question is whether the proofs can be repaired, or more generally, whether the security of GCM can ever be proved. In order to answer the question, we first introduce a combinatorial problem of quantifying a cardinality of a certain set of bit strings. The problem belongs to one of the problems of counting the number of output differences with non-zero probability of S-functions [29], which presents tools to analyze ARX systems (e.g., see [23]). One possible approach to solve the problem is to follow [29] (or [22]). In this paper, we take another approach and present a solution to the problem by giving a recursive formula that quantifies the cardinality. Basing on the solution, we present new security bounds on GCM, both for privacy and authenticity.

As a result, although the security bounds are larger than what were previously claimed, we show that GCM maintains its provable security. We also present provable security results of GCM when the nonce length is restricted to 96 bits, in which case GCM has better security bounds than a general case.

2 Preliminaries

Let $\{0, 1\}^*$ be the set of all bit strings, and for an integer $\ell \geq 0$, let $\{0, 1\}^\ell$ be a set of ℓ -bit strings. For a bit string $X \in \{0, 1\}^*$, $|X|$ is its length in bits, and $|X|_\ell = \lceil |X|/\ell \rceil$ is the length in ℓ -bit blocks. The empty string is denoted as ε . Let 0^ℓ and 1^ℓ denote the bit strings of ℓ zeros and ones, respectively. We use the prefix $0x$ for the hexadecimal notation, e.g., $0x63$ is $01100011 \in \{0, 1\}^8$. We also write $(0x)^\ell$ to mean $0^{4\ell}$. For a bit string X and an integer ℓ such that $|X| \geq \ell$, $\text{msb}_\ell(X)$ is the most significant ℓ bits (the leftmost ℓ bits) of X , and $\text{lsb}_\ell(X)$ is the least significant ℓ bits (the rightmost ℓ bits) of X . For $X, Y \in \{0, 1\}^*$, we write $X \parallel Y$, (X, Y) , or simply XY to denote their concatenation. For a bit string X whose length in bits is a multiple of ℓ , we write its partition into ℓ -bit strings as $(X[1], \dots, X[x]) \xleftarrow{\ell} X$, where $X[1], \dots, X[x] \in \{0, 1\}^\ell$ are unique bit strings such that $X[1] \parallel \dots \parallel X[x] = X$. For non-negative integers a and ℓ with $a \leq 2^\ell - 1$, let $\text{str}_\ell(a)$ be its ℓ -bit binary representation, i.e., if $a = a_{\ell-1}2^{\ell-1} + \dots + a_12 + a_0$ for $a_{\ell-1}, \dots, a_1, a_0 \in \{0, 1\}$, then $\text{str}_\ell(a) = a_{\ell-1} \dots a_1 a_0 \in \{0, 1\}^\ell$. For a bit string $X = X_{\ell-1} \dots X_1 X_0 \in \{0, 1\}^\ell$, let $\text{int}(X)$ be the integer $X_{\ell-1}2^{\ell-1} + \dots + X_12 + X_0$. For a finite set \mathcal{X} , $\#\mathcal{X}$ denotes its cardinality, and $X \xleftarrow{\$} \mathcal{X}$ means the uniform sampling of an element from \mathcal{X} and assigning it to X .

Throughout this paper, we fix a block length n and a blockcipher $E : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, where \mathcal{K} is a non-empty set of keys. Unless otherwise specified, we let $n = 128$. We write E_K for the permutation specified by $K \in \mathcal{K}$, and $C = E_K(M)$ for the ciphertext of a plaintext $M \in \{0, 1\}^n$ under the key $K \in \mathcal{K}$. The set of n -bit strings, $\{0, 1\}^n$, is also regarded as $\text{GF}(2^n)$, the finite field with 2^n elements. An n -bit string $a_{n-1} \dots a_1 a_0 \in \{0, 1\}^n$ corresponds to a formal polynomial $a(\mathbf{x}) = a_{n-1} + a_{n-2}\mathbf{x} + \dots + a_1\mathbf{x}^{n-2} + a_0\mathbf{x}^{n-1} \in \text{GF}(2)[\mathbf{x}]$. When $n = 128$, the irreducible polynomial used in GCM is $p(\mathbf{x}) = 1 + \mathbf{x} + \mathbf{x}^2 + \mathbf{x}^7 + \mathbf{x}^{128}$.

3 Specification of GCM

We follow [27,28] with some notational changes. GCM is parameterized by a blockcipher $E : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and a tag length τ , where $64 \leq \tau \leq n$. We write $\text{GCM}[E, \tau]$ for GCM that uses E and τ as parameters. Let $\text{GCM-}\mathcal{E}$ be the encryption algorithm and $\text{GCM-}\mathcal{D}$ be the decryption algorithm, which are defined in Fig. 1. GCM uses the counter mode encryption and the polynomial hash function over $\text{GF}(2^n)$ as its subroutines. They are denoted CTR and GHASH and are defined in Fig. 2. Figure 3 illustrates the overall structure of $\text{GCM-}\mathcal{E}$.

$\text{GCM-}\mathcal{E}$ takes a key $K \in \mathcal{K}$, a nonce $N \in \{0, 1\}^*$, an associated data $A \in \{0, 1\}^*$, and a plaintext $M \in \{0, 1\}^*$ as inputs, and returns a pair of a ciphertext $C \in \{0, 1\}^*$ and a tag $T \in \{0, 1\}^\tau$ as an output, where $1 \leq |N| \leq 2^{n/2} - 1$, $0 \leq |A| \leq 2^{n/2} - 1$, $0 \leq |M| \leq n(2^{32} - 2)$, and $|C| = |M|$. We write $(C, T) \leftarrow \text{GCM-}\mathcal{E}_K^{N, A}(M)$. $\text{GCM-}\mathcal{D}$ takes a key $K \in \mathcal{K}$, a nonce $N \in \{0, 1\}^*$, an associated data $A \in \{0, 1\}^*$, a ciphertext $C \in \{0, 1\}^*$, and a tag $T \in \{0, 1\}^\tau$ as inputs, and returns either a plaintext $M \in \{0, 1\}^*$ or the symbol \perp indicating that the inputs are invalid. We write $M \leftarrow \text{GCM-}\mathcal{D}_K^{N, A}(C, T)$ or $\perp \leftarrow \text{GCM-}\mathcal{D}_K^{N, A}(C, T)$.

Algorithm GCM- $\mathcal{E}_K^{N,A}(M)$	Algorithm GCM- $\mathcal{D}_K^{N,A}(C, T)$
<ol style="list-style-type: none"> 1. $L \leftarrow E_K(0^n)$ 2. if $N = 96$ then $I[0] \leftarrow N \parallel 0^{31}1$ 3. else $I[0] \leftarrow \text{GHASH}_L(\varepsilon, N)$ 4. $m \leftarrow M _n$ 5. $S \leftarrow \text{CTR}_K(I[0], m)$ 6. $C \leftarrow M \oplus \text{msb}_{ M }(S)$ 7. $\bar{T} \leftarrow E_K(I[0]) \oplus \text{GHASH}_L(A, C)$ 8. $T \leftarrow \text{msb}_\tau(\bar{T})$ 9. return (C, T) 	<ol style="list-style-type: none"> 1. $L \leftarrow E_K(0^n)$ 2. if $N = 96$ then $I[0] \leftarrow N \parallel 0^{31}1$ 3. else $I[0] \leftarrow \text{GHASH}_L(\varepsilon, N)$ 4. $\bar{T}^* \leftarrow E_K(I[0]) \oplus \text{GHASH}_L(A, C)$ 5. $T^* \leftarrow \text{msb}_\tau(\bar{T}^*)$ 6. if $T \neq T^*$ then return \perp 7. $m \leftarrow C _n$ 8. $S \leftarrow \text{CTR}_K(I[0], m)$ 9. $M \leftarrow C \oplus \text{msb}_{ C }(S)$ 10. return M

Fig. 1. The encryption and decryption algorithms of GCM

Algorithm CTR $_K(I[0], m)$	Algorithm GHASH $_L(A, C)$
<ol style="list-style-type: none"> 1. for $j \leftarrow 1$ to m do 2. $I[j] \leftarrow \text{inc}(I[j - 1])$ 3. $S[j] \leftarrow E_K(I[j])$ 4. $S \leftarrow (S[1], S[2], \dots, S[m])$ 5. return S 	<ol style="list-style-type: none"> 1. $a \leftarrow n A _n - A$ 2. $c \leftarrow n C _n - C$ 3. $X \leftarrow A \parallel 0^a \parallel C \parallel 0^c \parallel \text{str}_{n/2}(A) \parallel \text{str}_{n/2}(C)$ 4. $(X[1], \dots, X[x]) \xleftarrow{n} X$ 5. $Y \leftarrow 0^n$ 6. for $j \leftarrow 1$ to x do 7. $Y \leftarrow L \cdot (Y \oplus X[j])$ 8. return Y

Fig. 2. Subroutines used in the encryption and decryption algorithms

In the definition of CTR, for a bit string $X \in \{0, 1\}^n$, $\text{inc}(X)$ treats the least significant 32 bits (the rightmost 32 bits) of X as a non-negative integer, and increments this value modulo 2^{32} , i.e.,

$$\text{inc}(X) = \text{msb}_{n-32}(X) \parallel \text{str}_{32}(\text{int}(\text{lsb}_{32}(X)) + 1 \bmod 2^{32}).$$

For $r \geq 0$, we write $\text{inc}^r(X)$ to denote the r times iterative applications of inc on X , and $\text{inc}^{-r}(X)$ to denote the r times iterative applications of the inverse function of inc on X . We use the convention that $\text{inc}^0(X) = X$. In the definition of GHASH, the multiplication in line 7 is over $\text{GF}(2^n)$. We remark that, if $|N| \neq 96$, we have $\text{GHASH}_L(\varepsilon, N) = X[1] \cdot L^x \oplus \dots \oplus X[x] \cdot L$, where $X = (X[1], \dots, X[x]) = N \parallel 0^{n|N|_n - |N|} \parallel \text{str}_n(|N|)$.

4 Security Definitions

An adversary is a probabilistic algorithm that has access to one or more oracles. Let $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \dots}$ denote an adversary \mathcal{A} interacting with oracles $\mathcal{O}_1, \mathcal{O}_2, \dots$, and $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \dots} \Rightarrow 1$ denote the event that \mathcal{A} , after interacting with $\mathcal{O}_1, \mathcal{O}_2, \dots$,

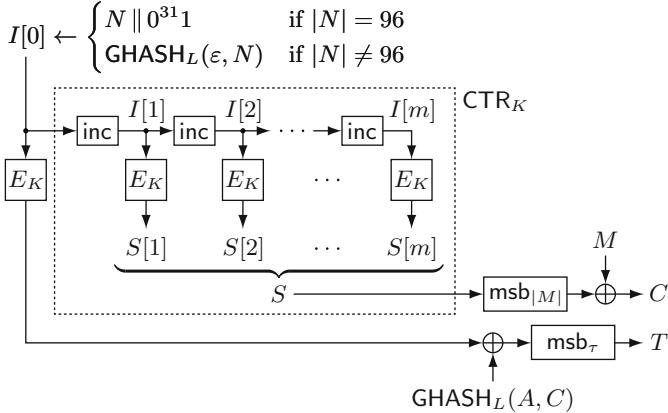


Fig. 3. The encryption algorithm of GCM

outputs 1. The resources of \mathcal{A} are measured in terms of time and query complexities. The time complexity includes the description size of \mathcal{A} , and we fix a model of computation and a method of encoding. The query complexity includes the number of queries, the total length of queries, and the maximum length of queries, and a more precise definition is given in each theorem statement.

Following [10,30], we consider two security notions for GCM: privacy and authenticity. For privacy, we consider an adversary \mathcal{A} that has access to a GCM encryption oracle or a random-bits oracle. The GCM encryption oracle takes (N, A, M) and returns $(C, T) \leftarrow \text{GCM-}\mathcal{E}_K^{N,A}(M)$. The random-bits oracle, $\$$, takes (N, A, M) and returns $(C, T) \leftarrow \$\{0, 1\}^{|M|+\tau}$. We define

$$\mathbf{Adv}_{\text{GCM}[E, \tau]}^{\text{priv}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\text{GCM-}\mathcal{E}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\$} \Rightarrow 1],$$

where the first probability is defined over the randomness of \mathcal{A} and the choice of K , and the last is over the randomness of \mathcal{A} and the random-bits oracle. We assume that \mathcal{A} is nonce-respecting: \mathcal{A} does not make two queries with the same nonce.

For authenticity, we consider an adversary \mathcal{A} that has access to GCM encryption and decryption oracles. The GCM decryption oracle takes (N, A, C, T) and returns $M \leftarrow \text{GCM-}\mathcal{D}_K^{N,A}(C, T)$ or $\perp \leftarrow \text{GCM-}\mathcal{D}_K^{N,A}(C, T)$. We define

$$\mathbf{Adv}_{\text{GCM}[E, \tau]}^{\text{auth}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\text{GCM-}\mathcal{E}_K, \text{GCM-}\mathcal{D}_K} \text{ forges}],$$

where the probability is defined over the randomness of \mathcal{A} and the choice of K , and the adversary forges if the GCM decryption oracle returns a bit string (other than \perp) for a query (N, A, C, T) , but (C, T) was not previously returned to \mathcal{A} from the encryption oracle for a query (N, A, M) . As in the privacy notion, we assume that \mathcal{A} is nonce-respecting: \mathcal{A} does not make two queries to the encryption oracle with the same nonce. We remark that nonces used for the

encryption queries can be used for decryption queries and vice-versa, and that the same nonce can be repeated for decryption queries. Without loss of generality, we assume that \mathcal{A} does not make trivial queries: if the encryption oracle returns (C, T) for a query (N, A, M) , then \mathcal{A} does not make a query (N, A, C, T) to the decryption oracle, and \mathcal{A} does not repeat a query to the decryption oracle.

In [27], McGrew and Viega analyzed the security of GCM, and there are differences between the above security notions. In [27], for privacy, the adversary has access to both the encryption and decryption oracles, while we chose to follow a more standard notion [10,30] where the privacy adversary has access to the encryption oracle only. Another difference is the assumption about the nonce reuse. In [27], the adversary is not allowed to reuse a nonce within decryption queries (but nonces used in encryption queries can be used in decryption queries and vice-versa), while our adversary can reuse nonces within decryption queries.

For the blockcipher $E : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, we consider the PRP notion [24]. Let $\text{Perm}(n)$ be the set of all permutations on $\{0, 1\}^n$. We say that P is a random permutation if $P \xleftarrow{\$} \text{Perm}(n)$. We define

$$\mathbf{Adv}_E^{\text{prp}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{E_K} \Rightarrow 1] - \Pr[P \xleftarrow{\$} \text{Perm}(n) : \mathcal{A}^P \Rightarrow 1],$$

where the probabilities are defined over the randomness of \mathcal{A} , and the choices of K and P , respectively. We write $\text{GCM}[\text{Perm}(n), \tau]$ for GCM that uses a random permutation P as a blockcipher E_K , and we write the corresponding encryption and decryption algorithms as $\text{GCM-}\mathcal{E}_P$ and $\text{GCM-}\mathcal{D}_P$, respectively.

We also consider GCM that uses a random function as E_K , which is naturally defined as the invertibility of E_K is irrelevant in the definition of GCM. Let $\text{Rand}(n)$ be the set of all functions from $\{0, 1\}^n$ to $\{0, 1\}^n$. We say that F is a random function if $F \xleftarrow{\$} \text{Rand}(n)$, and write $\text{GCM}[\text{Rand}(n), \tau]$ for GCM that uses F as E_K . We write the corresponding encryption and decryption algorithms as $\text{GCM-}\mathcal{E}_F$ and $\text{GCM-}\mathcal{D}_F$, respectively.

5 Breaking GCM Security Proofs

5.1 Review of [27, Lemma 3], [27, Theorem 1], and [27, Theorem 2]

In this section, we first review a lemma in [27] that was used to derive the provable security results on GCM. Consider $\text{GCM}[\text{Rand}(n), \tau]$, GCM with E_K being a random function F , and the privacy notion for it. Let (N_1, A_1, M_1) and (N_2, A_2, M_2) be two encryption queries, where $N_1 \neq N_2$ and $|N_1|, |N_2| \neq 96$. Let $I_1[0] \leftarrow \text{GHASH}_L(\varepsilon, N_1)$, and $I_1[j] \leftarrow \text{inc}(I_1[j-1])$ for $1 \leq j \leq m_1$, where $m_1 = |M_1|_n$. Similarly, let $I_2[0] \leftarrow \text{GHASH}_L(\varepsilon, N_2)$, and $I_2[j] \leftarrow \text{inc}(I_2[j-1])$ for $1 \leq j \leq m_2$, where $m_2 = |M_2|_n$. If we have

$$\{I_1[0], I_1[1], \dots, I_1[m_1]\} \cap \{I_2[0], I_2[1], \dots, I_2[m_2]\} = \emptyset$$

and $I_i[j] \neq 0^n$ for $i = 1, 2$ and $0 \leq j \leq m_i$, then the two masks $S_1 = (S_1[1], \dots, S_1[m_1])$ and $S_2 = (S_2[1], \dots, S_2[m_2])$, produced from the counter mode encryption based on F , are uniformly distributed over $(\{0, 1\}^n)^{m_1}$ and $(\{0, 1\}^n)^{m_2}$, respectively. Furthermore, $F(I_1[0])$ and $F(I_2[0])$ are uniform random n -bit strings, and hence the probability distribution of strings returned from the encryption oracle is identical to that from the random-bits oracle.

Therefore, in order to prove the security of GCM, one has to show that the probability of a counter collision, $I_1[j_1] = I_2[j_2]$ for some $0 \leq j_1 \leq m_1$ and $0 \leq j_2 \leq m_2$, is small. We see that the event is equivalent to $\text{inc}^{j_1}(\text{GHASH}_L(\varepsilon, N_1)) = \text{inc}^{j_2}(\text{GHASH}_L(\varepsilon, N_2))$. Let $\text{Coll}_L(r, N_1, N_2)$ denote the event

$$\text{inc}^r(\text{GHASH}_L(\varepsilon, N_1)) \oplus \text{GHASH}_L(\varepsilon, N_2) = 0^n.$$

We need to bound $\Pr[L \stackrel{\$}{\leftarrow} \{0, 1\}^n : \text{Coll}_L(r, N_1, N_2)]$, where $r = j_1 - j_2$, which we write $\Pr_L[\text{Coll}_L(r, N_1, N_2)]$ for simplicity. Since $-m_2 \leq r \leq m_1$ and $0 \leq m_1, m_2 \leq 2^{32} - 2$, the range of r is $-(2^{32} - 2) \leq r \leq 2^{32} - 2$.

In [27, Lemma 3], McGrew and Viega showed the following lemma (notation has been adapted to this paper).

Lemma 1 ([27]). *For any $-(2^{32} - 2) \leq r \leq 2^{32} - 2$, N_1 , and N_2 such that $N_1 \neq N_2$, $|N_1|, |N_2| \neq 96$, and $|N_1|_n, |N_2|_n \leq \ell_N$, $\Pr_L[\text{Coll}_L(r, N_1, N_2)] \leq (\ell_N + 1)/2^n$.*

Based on the lemma, [27, Theorem 1] states that the privacy advantage of GCM[Perm(n), τ], GCM with E_K being a random permutation P , is at most

$$\frac{0.5(\sigma/n + 2q)^2}{2^n} + \frac{2q(\sigma/n + 2q)\lceil\ell_N/n + 1\rceil}{2^n} + \frac{q\lceil\ell/n + 1\rceil}{2^\tau}, \quad (1)$$

and [27, Theorem 2] states that the authenticity advantage is at most

$$\frac{0.5(\sigma/n + 2q)^2}{2^n} + \frac{2q(\sigma/n + 2q + 1)\lceil\ell_N/n + 1\rceil}{2^n} + \frac{q\lceil\ell/n + 1\rceil}{2^\tau}, \quad (2)$$

where q is the maximum number of queries (either encryption or decryption queries), σ is the total length in bits of the plaintexts (either in encryption queries or returned from the decryption oracle), ℓ_N is the maximum length in bits of nonces in queries (either encryption or decryption queries), and ℓ is the maximum value of $|A_j| + |C_j|$, where A_j and C_j are the associated data and ciphertext, respectively, in the j -th query (either in decryption queries or returned from the encryption oracle). Note that the definitions of privacy and authenticity advantages are slightly different from ours, as explained in Sect. 4.

It is easy to see that the lemma is correct when $r = 0$: $\text{Coll}_L(0, N_1, N_2)$ is the event $\text{inc}^0(\text{GHASH}_L(\varepsilon, N_1)) \oplus \text{GHASH}_L(\varepsilon, N_2) = 0^n$, and we see that the left hand side is a non-trivial polynomial in L of degree at most $\ell_N + 1$ over $\text{GF}(2^n)$, and hence there are at most $\ell_N + 1$ values of L that satisfy the equality.

However, for $r \neq 0$, it is not clear if $\text{inc}^r(\text{GHASH}_L(\varepsilon, N_1))$ is a polynomial of degree at most $\ell_N + 1$ even if this is the case for $\text{GHASH}_L(\varepsilon, N_1)$, and the analysis for this case is missing in the proof of [27, Lemma 3]. The lemma is crucial in that it is used in the proofs for both privacy and authenticity.

5.2 Invalidating [27, Lemma 3]

Let $r = 1$, $N_1 = (0x0)^{17} \parallel 0x2 = 0^{68} \parallel 0010$, and $N_2 = (0x0)^{17} \parallel 0x6 = 0^{68} \parallel 0110$, where $|N_1| = |N_2| = 72$. Then $\text{Coll}_L(r, N_1, N_2)$ is equivalent to

$$\text{inc}^1(U_1 \cdot L^2 \oplus V \cdot L) \oplus (U_2 \cdot L^2 \oplus V \cdot L) = 0^n, \quad (3)$$

where $U_1 = (0x0)^{17} \parallel 0x2 \parallel (0x0)^{14}$, $U_2 = (0x0)^{17} \parallel 0x6 \parallel (0x0)^{14}$, and $V = (0x0)^{30} \parallel 0x48$. In binary, $U_1 = 0^{68} \parallel 0010 \parallel 0^{56}$, $U_2 = 0^{68} \parallel 0110 \parallel 0^{56}$, and $V = 0^{120} \parallel 01001000$. Now [27, Lemma 3] states that $\Pr_L[\text{Coll}_L(r, N_1, N_2)] \leq 2/2^n$, i.e., (3) has at most two solutions. However, one can verify (with the help of some software, e.g., [3]) that (3) has 32 solutions, which are listed in Appendix A. In other words, $\Pr_L[\text{Coll}_L(r, N_1, N_2)] \geq 32/2^n$ holds, and hence [27, Lemma 3] cannot be valid.

We present one more observation regarding the counter-example. Consider

$$\text{inc}^2(U_1 \cdot L^2 \oplus V \cdot L) \oplus (U_2 \cdot L^2 \oplus V \cdot L) = 0^n, \quad (4)$$

$$\text{inc}^4(U_1 \cdot L^2 \oplus V \cdot L) \oplus (U_2 \cdot L^2 \oplus V \cdot L) = 0^n, \quad (5)$$

where the values of U_1 , U_2 , and V are as above. Then one can verify that (4) has 31 solutions, and that (5) has 30 solutions, which are also listed in Appendix A. The 93 values of L are all distinct, and are also different from 0^n , which is a solution for $\text{inc}^0(U_1 \cdot L^2 \oplus V \cdot L) \oplus (U_2 \cdot L^2 \oplus V \cdot L) = 0^n$. Therefore we have

$$\Pr_L \left[\bigvee_{r=0,1,2,4} \text{Coll}_L(r, N_1, N_2) \right] \geq \frac{94}{2^n}. \quad (6)$$

We remark that we exclude the case $r = 3$ since $\text{Coll}_L(3, N_1, N_2)$ has no solution. In Appendix A, we present other examples of (N_1, N_2) that satisfy (6) and also invalidate [27, Lemma 3].

We next show that the above observation is not merely spotting of a subtle error in the proofs of GCM. The observation can actually be translated into a distinguishing attack.

5.3 Distinguishing Attack

Consider GCM with E_K being a random function F , and the privacy notion for it. We remark that the analysis of this idealized version of GCM is essential since the main part of the privacy proof is the analysis of this case. Let N_1 and N_2 be as in Sect. 5.2, $A_1 = \varepsilon$, $M_1 = 0^{5n}$, $A_2 = \varepsilon$, and $M_2 = 0^n$.

Let \mathcal{A} be an adversary that has access to an oracle \mathcal{O} which is either the GCM encryption oracle or the random-bits oracle. \mathcal{A} works as follows.

1. First, \mathcal{A} makes two queries, (N_i, A_i, M_i) for $i = 1, 2$, and obtains $(C_i, T_i) \leftarrow \mathcal{O}(N_i, A_i, M_i)$.
2. Let $(C_1[1], \dots, C_1[5]) \xleftarrow{n} C_1$ and output 1 if

$$C_1[1] = C_2 \text{ or } C_1[2] = C_2 \text{ or } C_1[3] = C_2 \text{ or } C_1[5] = C_2. \quad (7)$$

First, suppose that \mathcal{O} is the GCM encryption oracle. If $\text{Coll}_L(0, N_1, N_2) \vee \text{Coll}_L(1, N_1, N_2) \vee \text{Coll}_L(2, N_1, N_2) \vee \text{Coll}_L(4, N_1, N_2)$, then we see that \mathcal{A} outputs 1. Otherwise the probability distributions of $C_1[1]$, $C_1[2]$, $C_1[3]$, $C_1[5]$, and C_2 are exactly the same as those of returned by the random-bits oracle. In particular, (7) is satisfied with the same probability for the GCM encryption oracle and for the random-bits oracle. Therefore, we have

$$\mathbf{Adv}_{\text{GCM}[\text{Rand}(n), \tau]}^{\text{priv}}(\mathcal{A}) = \Pr[\mathcal{A}^{\text{GCM-}\mathcal{E}_F} \Rightarrow 1] - \Pr[\mathcal{A}^{\$} \Rightarrow 1] \geq \frac{94}{2^n}. \quad (8)$$

Now using the notation of (1) and (2), our adversary has the following query complexity: $q = 2$, $\sigma = 6n$, $\ell_N = 72$, and $\ell = 5n$. Then (1) is $50/2^n + 80/2^n + 12/2^\tau = 130/2^n + 12/2^\tau$. Therefore, the attack does not contradict the claimed privacy bound (1).

However, (1) allows the use of the GCM decryption oracle, and rounding up the details makes it sufficiently large so that our attack is tolerated in appearance. Now if we take a closer look at (1), the second term, $80/2^n$, is the main part of the privacy proof that bounds $\mathbf{Adv}_{\text{GCM}[\text{Rand}(n), \tau]}^{\text{priv}}(\mathcal{A})$, while the first term is from the application of the PRP/PRF switching lemma [11] and the last term bounds the forgery probability due to the use of the decryption oracle. Therefore, the above attack does invalidate the main part of the privacy proof, and we also see that it invalidates the second term of (2), which is $88/2^n$.

We have already shown that the claimed bound on $\Pr_L[\text{Coll}_L(r, N_1, N_2)]$ is invalid, which implies that the claimed security bounds, (1) and (2), are not justified. Furthermore, the above attack invalidates the main part of the privacy proof. Therefore, at this point, it is fair to conclude that the security of GCM is not supported by the proofs. We note that, although our attack does not work with 96-bit nonces, this statement holds even in this case since (1) and (2) cover a general case including the case that the nonce length is restricted to 96 bits.

5.4 Remarks

Our attack is efficient. It uses two oracle calls, and the lengths of the queries are short. However, the success probability, although large enough to invalidate the second terms in (1) and (2), is insignificantly small in practice and it has a limited practical implication. We also note that many standards require or recommend using GCM with 96-bit nonces, in which case the attack does not work. Indeed, in many RFCs, such as RFC 4106 (IPsec) [34], 5647 (SSH) [20], 5288 (SSL) [33], the nonce length is fixed to 96 bits, and RFC 5084 [19] and 5116 [25] recommend 96-bit nonces. For IEEE standards, some strictly require 96-bit nonces (e.g. IEEE 802.1AE [5]) and some do not (e.g. IEEE P1619.1 [6]). There are cases where non-96-bit nonces are allowed, including NIST SP 800-38D [15], ISO/IEC 19772 [7], PKCS #11 [4], and most software libraries (e.g., Gladman's code [17], CRYPTO++ [14], Java SE [2], and BouncyCastle [1]). Finally, NSA Suite B Cryptography includes GCM with a strong recommendation (but not mandatory; see e.g. [8]) of using it with 96-bit nonces.

We emphasize that, even when non-96-bit nonces are allowed, our attack has a limited practical implication. However, it does undermine the provable security of GCM, making its provable security open. A very natural question is then whether the security of GCM can ever be proved. In the following sections, we present an affirmative answer to this question.

6 Towards Repairing the Proofs

6.1 Combinatorial Problem

To prove the security of GCM, the first step is to derive the upper bound on $\Pr_L[\text{Coll}_L(r, N_1, N_2)]$. The obvious bound is $\Pr_L[\text{Coll}_L(r, N_1, N_2)] \leq (\ell_N + 1)/2^{n-32}$, which can be shown by ignoring the least significant 32 bits. In this section, we derive a better upper bound on $\Pr_L[\text{Coll}_L(r, N_1, N_2)]$, and we first introduce a combinatorial problem for this goal.

For $0 \leq r \leq 2^{32} - 1$, let

$$\mathbb{Y}_r \stackrel{\text{def}}{=} \{\text{str}_{32}(\text{int}(Y) + r \bmod 2^{32}) \oplus Y \mid Y \in \{0, 1\}^{32}\}. \quad (9)$$

We also let $\alpha_r \stackrel{\text{def}}{=} \#\mathbb{Y}_r$ and $\alpha_{\max} \stackrel{\text{def}}{=} \max\{\alpha_r \mid 0 \leq r \leq 2^{32} - 1\}$. For given r , it is not hard to experimentally derive the value of α_r by exhaustively evaluating $\text{str}_{32}(\text{int}(Y) + r \bmod 2^{32}) \oplus Y$ for all $Y \in \{0, 1\}^{32}$. For example, we have

$$\alpha_0 = 1, \alpha_1 = 32, \alpha_2 = 31, \alpha_3 = 61, \alpha_4 = 30, \alpha_5 = 89, \dots$$

and the problem is to identify α_{\max} .

6.2 Relation to the Security of GCM

We show that identifying α_r gives the upper bound on $\Pr_L[\text{Coll}_L(r, N_1, N_2)]$.

Lemma 2. *For any $0 \leq r \leq 2^{32}-1$, N_1 , and N_2 such that $N_1 \neq N_2$, $|N_1|, |N_2| \neq 96$, and $|N_1|_n, |N_2|_n \leq \ell_N$, $\Pr_L[\text{Coll}_L(r, N_1, N_2)] \leq \alpha_r(\ell_N + 1)/2^n$.*

Proof. Let $Y_1, \dots, Y_{\alpha_r} \in \{0, 1\}^{32}$ be the α_r elements of \mathbb{Y}_r . Let $\mathcal{Y}_1, \dots, \mathcal{Y}_{\alpha_r} \subseteq \{0, 1\}^{32}$ be the α_r disjoint subsets of $\{0, 1\}^{32}$ such that

$$\mathcal{Y}_j = \{Y \in \{0, 1\}^{32} \mid \text{str}_{32}(\text{int}(Y) + r \bmod 2^{32}) \oplus Y = Y_j\}$$

for $1 \leq j \leq \alpha_r$, $\mathcal{Y}_1 \cup \dots \cup \mathcal{Y}_{\alpha_r} = \{0, 1\}^{32}$, and $\mathcal{Y}_j \cap \mathcal{Y}_{j'} = \emptyset$ for $1 \leq j < j' \leq \alpha_r$. Observe that if $Y \in \mathcal{Y}_j$, then $\text{str}_{32}(\text{int}(Y) + r \bmod 2^{32})$ can be replaced with $Y \oplus Y_j$.

For $1 \leq j \leq \alpha_r$, let D_j be the event $\text{Coll}_L(r, N_1, N_2) \wedge \text{lsb}_{32}(\text{GHASH}_L(\varepsilon, N_1)) \in \mathcal{Y}_j$. Since D_1, \dots, D_{α_r} are disjoint events, we have

$$\Pr_L[\text{Coll}_L(r, N_1, N_2)] = \sum_{1 \leq j \leq \alpha_r} \Pr_L[D_j]. \quad (10)$$

Recall that $\text{Coll}_L(r, N_1, N_2)$ is the event $\text{inc}^r(\text{GHASH}_L(\varepsilon, N_1)) \oplus \text{GHASH}_L(\varepsilon, N_2) = 0^n$, and since $\text{lsb}_{32}(\text{GHASH}_L(\varepsilon, N_1)) \in \mathcal{Y}_j$, $\text{inc}^r(\text{GHASH}_L(\varepsilon, N_1))$ can be replaced with $\text{GHASH}_L(\varepsilon, N_1) \oplus (0^{n-32} \parallel Y_j)$, implying that the event D_j is equivalent to

$$\text{GHASH}_L(\varepsilon, N_1) \oplus \text{GHASH}_L(\varepsilon, N_2) \oplus (0^{n-32} \parallel Y_j) = 0^n \quad (11)$$

and $\text{lsb}_{32}(\text{GHASH}_L(\varepsilon, N_1)) \in \mathcal{Y}_j$. We see that (11) is a non-trivial equation in L of degree at most $\ell_N + 1$ over $\text{GF}(2^n)$, and hence it has at most $\ell_N + 1$ solutions. From (10), we obtain the lemma. \square

6.3 Deriving α_r and α_{\max}

The problem introduced in Sect. 6.1 can be solved by exhaustively evaluating (9) for all $0 \leq r \leq 2^{32} - 1$, which is computationally costly. Another possible approach is to follow the framework in [29] (or to use tools in [22]).

Instead of following these approaches, in this section, we present a recursive formula to efficiently compute α_r . Let $r \geq 0$ be a given integer, ℓ be the number of runs of ones in the binary representation of r (e.g., if r is 00110101 in binary, then $\ell = 3$), and v_ℓ be an integer with $r \leq 2^{v_\ell} - 1$. Suppose $\text{str}_{v_\ell}(r) = 0^{s_\ell}1^{t_\ell}\dots0^{s_1}1^{t_1}0^{s_0}$, where $s_{\ell-1}, \dots, s_1 \geq 1$, $s_\ell, s_0 \geq 0$, $t_\ell, \dots, t_1 \geq 1$, and $v_\ell = (s_\ell + \dots + s_1 + s_0) + (t_\ell + \dots + t_1)$.

Define

$$A_\ell \stackrel{\text{def}}{=} \#\{\text{str}_{v_\ell}(\text{int}(Y) + r \bmod 2^{v_\ell}) \oplus Y \mid Y \in \{0, 1\}^{v_\ell}\}.$$

Note that, when $v_\ell = 32$, α_r is equal to A_ℓ . The next proposition gives an efficient recursive formula to compute A_ℓ .

Proposition 1. *For any $\ell \geq 1$,*

$$A_\ell = \begin{cases} t_\ell A_{\ell-1} + B_{\ell-1} & \text{if } s_\ell = 0, \\ s_\ell B_\ell + A_{\ell-1} & \text{if } s_\ell \geq 1, \end{cases} \quad (12)$$

where $B_j = t_j A_{j-1} + B_{j-1}$ for $1 \leq j \leq \ell$, $A_j = s_j B_j + A_{j-1}$ for $1 \leq j \leq \ell - 1$, $A_0 = 1$, and $B_0 = 0$.

An elementary proof of the proposition is given in the full version of this paper. Given Proposition 1, it is not hard to experimentally derive α_{\max} by directly evaluating (12). We have $\alpha_{\max} = 3524578$, where $\alpha_r = \alpha_{\max}$ holds when $r = 0x2aaaaaab, 0xaaaaaaaaab, 0x5555555555$, and $0xd555555555$.

From Lemma 2 and since $3524578 \leq 2^{22}$, we obtain the following corollary.

Corollary 1. *For any $0 \leq r \leq 2^{32} - 1$, N_1 , and N_2 such that $N_1 \neq N_2$, $|N_1|, |N_2| \neq 96$, and $|N_1|_n, |N_2|_n \leq \ell_N$, $\Pr_L[\text{Coll}_L(r, N_1, N_2)] \leq 2^{22}(\ell_N + 1)/2^n$.*

If the upper bound of r is smaller than $2^{32} - 1$, then depending on the value, the constant, 2^{22} , in Corollary 1 can be replaced by a smaller constant. Specifically, there are 303 values of $1 \leq r \leq 2^{32} - 1$ that satisfy $\max\{\alpha_0, \dots, \alpha_{r-1}\} < \alpha_r$, and

Table 1. List of (r, α_r) (left) and the relation between r_{\max} and $\beta(r_{\max})$ (right)

r	α_r	Range of r_{\max}	$\beta(r_{\max})$
0x00000001	32	0x00000001–0x00000002	2^0
0x00000003	61	0x00000003–0x00000004	2^6
0x00000005	89	0x00000005–0x0000000a	2^7
0x0000000b	143	0x0000000b–0x00000024	2^8
0x00000025	294	0x00000025–0x00000054	2^9
0x00000055	538	0x00000055–0x0000012a	2^{10}
0x0000012b	1115	0x0000012b–0x00000454	2^{11}
0x00000455	2113	0x00000455–0x00000954	2^{12}
0x00000955	4124	0x00000955–0x000024aa	2^{13}
0x000024ab	8579	0x000024ab–0x00005554	2^{14}
0x00005555	17389	0x00005555–0x00012aaa	2^{15}
0x00012aab	34702	0x00012aab–0x00049554	2^{16}
0x00049555	69742	0x00049555–0x000aaaaa	2^{17}
0x000aaaaab	138117	0x000aaaaab–0x00255554	2^{18}
0x00255555	262471	0x00255555–0x00955554	2^{19}
0x00955555	559000	0x00955555–0x02555554	2^{20}
0x02555555	1127959	0x02555555–0x0a555554	2^{21}
0x0a555555	2116814	0x0a555555–0xffffffff	2^{22}

a list of the 303 values of (r, α_r) can be used to obtain a smaller constant. The list can be found in the full version of this paper. Table 1 (left) is the excerpt from the list for better readability and usability. For each $i = 5, 6, \dots, 22$, Table 1 (left) lists the minimum value of r , among the 303 values, such that $2^{i-1} < \alpha_r \leq 2^i$.

Table 1 (right) is obtained from Table 1 (left) and shows the relation between the range of r_{\max} and the corresponding constant $\beta(r_{\max})$, where r_{\max} is the upper bound of r and $\beta(r_{\max})$ is the upper bound of α_r . For example, if $r_{\max} = 2^{10} = 0x400$, then it is within the range of 0x0000012b–0x00000454 and hence $\beta(r_{\max}) = 2^{11}$. See also Fig. 4 for a graph showing the relation between r_{\max} and $\beta(r_{\max})$.

We have the following corollary.

Corollary 2. *For any $0 \leq r \leq r_{\max}$, N_1 , and N_2 such that $N_1 \neq N_2$, $|N_1|, |N_2| \neq 96$, and $|N_1|_n, |N_2|_n \leq \ell_N$, $\Pr_L[\text{Coll}_L(r, N_1, N_2)] \leq \beta(r_{\max})(\ell_N + 1)/2^n$.*

We note that for $r_{\max} = 0$, from $\alpha_0 = 1$, $\beta(r_{\max})$ is defined to be 1.

7 Repairing GCM Security Proofs

In this section, basing on the results of the previous section, we present new security bounds on GCM. We also present overviews of the proofs.

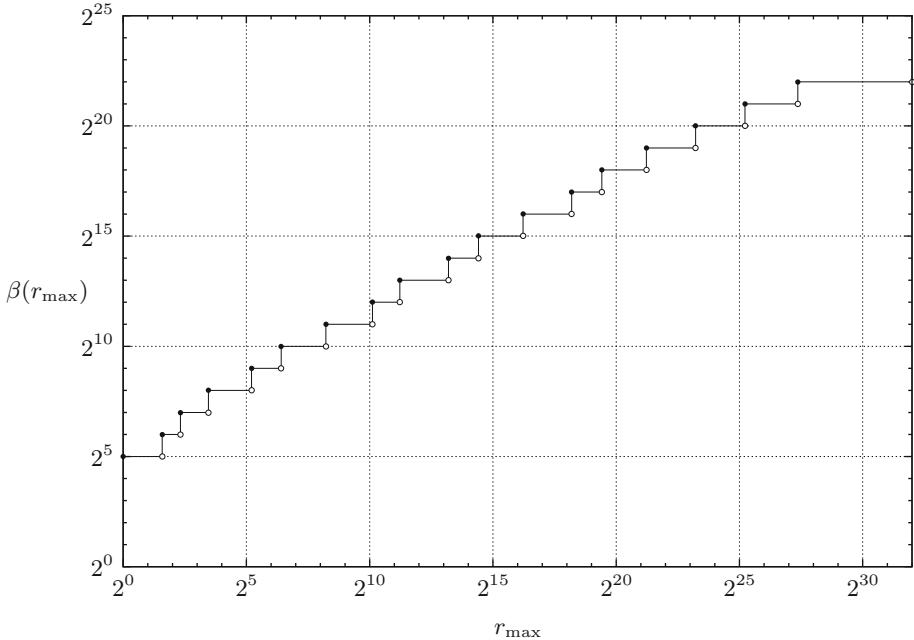


Fig. 4. Relation between r_{\max} and $\beta(r_{\max})$

7.1 Privacy Result

In the privacy result, if \mathcal{A} makes q queries $(N_1, A_1, M_1), \dots, (N_q, A_q, M_q)$, then the total plaintext length is $m_1 + \dots + m_q$, and the maximum nonce length is $\max\{n_1, \dots, n_q\}$, where $|N_i|_n = n_i$ and $|M_i|_n = m_i$. We have the following information theoretic result.

Theorem 1. Let $\text{Perm}(n)$ and τ be the parameters of GCM. Then for any \mathcal{A} that makes at most q queries, where the total plaintext length is at most σ blocks and the maximum nonce length is at most ℓ_N blocks,

$$\mathbf{Adv}_{\text{GCM}[\text{Perm}(n), \tau]}^{\text{priv}}(\mathcal{A}) \leq \frac{0.5(\sigma + q + 1)^2}{2^n} + \frac{2^{22}q(\sigma + q)(\ell_N + 1)}{2^n}.$$

Observe that the security bound is essentially the same as the original one (1), except that we have a constant, 2^{22} , in the second term, and we do not have a term that corresponds to the forgery probability.

We next present a corollary showing that GCM has a better security bound if the nonce length is restricted to 96 bits.

Corollary 3. Assume that the nonce length is restricted to 96 bits. Then,

$$\mathbf{Adv}_{\text{GCM}[\text{Perm}(n), \tau]}^{\text{priv}}(\mathcal{A}) \leq \frac{0.5(\sigma + q + 1)^2}{2^n}.$$

Let E be a blockcipher secure in the sense of the PRP notion. Then the corresponding complexity theoretic results, where E is used in place of $\text{Perm}(n)$, can be obtained by a standard argument (see e.g. [9]).

In the next section, we present an intuitive proof overview of Theorem 1. A complete proof is presented in the full version of this paper. A proof of Corollary 3 is obtained by modifying the proof of Theorem 1, and is omitted.

7.2 Proof Overview of Theorem 1

Suppose that \mathcal{A} has access to the GCM encryption oracle. We first replace the random permutation P by a random function F . The difference between the two advantage functions is at most $(\sigma + q + 1)^2 / 2^{n+1}$ from the PRP/PRF switching lemma [11]. Next, suppose that \mathcal{A} has made $i - 1$ queries $(N_1, A_1, M_1), \dots, (N_{i-1}, A_{i-1}, M_{i-1})$, obtained $(C_1, T_1), \dots, (C_{i-1}, T_{i-1})$, and is now making the i -th query (N_i, A_i, M_i) . For $1 \leq j \leq i$, let $\mathcal{I}_j = \{I_j[0], I_j[1], \dots, I_j[m_j]\}$ be a set of n -bit strings used as the inputs of F during the computation of (C_j, T_j) for the query (N_j, A_j, M_j) (other than 0^n used to generate L). Observe that, if $I_i[0], I_i[1], \dots, I_i[m_i]$ are not previously used, then (C_i, T_i) is a random string of $|M_i| + \tau$ bits. That is, unless

$$\mathcal{I}_i \cap (\{0^n\} \cup \mathcal{I}_1 \cup \dots \cup \mathcal{I}_{i-1}) \neq \emptyset \quad (13)$$

holds for some $1 \leq i \leq q$, the distribution of the output of the GCM encryption oracle (based on the random function F) is identical to that of the random-bits oracle, and hence \mathcal{A} is unable to distinguish between the two oracles. It can be shown that the probability of (13) is at most $2^{22}q(\sigma + q)(\ell_N + 1)/2^n$ by using Corollary 1. The result is obtained by summing up the two above-mentioned probabilities.

7.3 Authenticity Result

If \mathcal{A} makes q encryption queries $(N_1, A_1, M_1), \dots, (N_q, A_q, M_q)$ and q' decryption queries $(N'_1, A'_1, C'_1, T'_1), \dots, (N'_{q'}, A'_{q'}, C'_{q'}, T'_{q'})$, then the total plaintext length is $m_1 + \dots + m_q$, the maximum nonce length is $\max\{n_1, \dots, n_q, n'_1, \dots, n'_{q'}\}$, and the maximum input length is $\max\{a_1 + m_1, \dots, a_q + m_q, a'_1 + m'_1, \dots, a'_{q'} + m'_{q'}\}$, where $|N_i|_n = n_i$, $|A_i|_n = a_i$, $|M_i|_n = m_i$, $|N'_i|_n = n'_i$, $|A'_i|_n = a'_i$, and $|C'_i|_n = m'_i$. We have the following information theoretic result.

Theorem 2. *Let $\text{Perm}(n)$ and τ be the parameters of GCM. Then for any \mathcal{A} that makes at most q encryption queries and q' decryption queries, where the total plaintext length is at most σ blocks, the maximum nonce length is at most ℓ_N blocks, and the maximum input length is at most ℓ_A blocks,*

$$\begin{aligned} \mathbf{Adv}_{\text{GCM}[\text{Perm}(n), \tau]}^{\text{auth}}(\mathcal{A}) &\leq \frac{0.5(\sigma + q + q' + 1)^2}{2^n} \\ &+ \frac{2^{22}(q + q' + 1)(\sigma + q)(\ell_N + 1)}{2^n} + \frac{q'(\ell_A + 1)}{2^\tau}. \end{aligned} \quad (14)$$

As in the privacy result, the bound is essentially the same as the original one (2), except that we have a constant, 2^{22} , in the second term. The next corollary shows that we have a better security bound if the nonce length is restricted to 96 bits.

Corollary 4. *Assume that the nonce length is restricted to 96 bits. Then,*

$$\mathbf{Adv}_{\text{GCM}[\text{Perm}(n), \tau]}^{\text{auth}}(\mathcal{A}) \leq \frac{0.5(\sigma + q + q' + 1)^2}{2^n} + \frac{q'(\ell_A + 1)}{2^\tau}.$$

The corresponding complexity theoretic results can be obtained based on the PRP notion of a blockcipher E by a standard argument (see e.g. [9]).

We present an intuitive proof overview of Theorem 2 in the next section, and a complete proof is presented in the full version of this paper. A proof of Corollary 4 can be obtained from the proof of Theorem 2, and is omitted.

7.4 Proof Overview of Theorem 2

We replace the random permutation P by a random function F . From the PRP/PRF switching lemma [11], we have a term $(\sigma + q + q' + 1)^2/2^{n+1}$. We then consider the probability of a counter collision as in the privacy proof, but this time, we consider the counter values used for decryption queries as well. The probability can be shown to be at most $2^{22}(q + q' + 1)(\sigma + q)(\ell_N + 1)/2^n$ by using Corollary 1. Under the condition that there is no counter collision, the adversary is essentially asked to forge a message authentication code $(N, A, C) \rightarrow F(g(N)) \oplus \text{GHASH}_L(A, C)$, where $g(N) = N \parallel 0^{31}1$ if $|N| = 96$, and $g(N) = \text{GHASH}_L(\varepsilon, N)$ if $|N| \neq 96$. The probability can be shown to be at most $q'(\ell_A + 1)/2^\tau$, and we obtain the theorem by summing up the three probabilities.

7.5 Better Security Bounds

Use of Corollary 2. Suppose that, either in privacy or authenticity notions, \mathcal{A} makes q encryption queries $(N_1, A_1, M_1), \dots, (N_q, A_q, M_q)$. Let ℓ_M be the maximum plaintext length, which is $\max\{m_1, \dots, m_q\}$, where $|M_i|_n = m_i$. Theorem 1 and Theorem 2 assume $\ell_M = 2^{32} - 2$ and use Corollary 1 to obtain the results. However, if ℓ_M is known to be smaller, then Corollary 2 can be used to obtain better bounds. Specifically, in Theorem 1 and Theorem 2, if $\ell_M \leq r_{\max}$, then the constant becomes $\beta(r_{\max})$ instead of 2^{22} . For example, if ℓ_M is 2^{10} , then from Table 1 and by following the argument in Sect. 6.3, the constant becomes 2^{11} .

Use of [12, Theorem 2.3]. Using Bernstein's result [12, Theorem 2.3], we can further improve the authenticity bound (but not the privacy bound). For positive integer a , let $\delta_n(a) = (1 - (a - 1)/2^n)^{-a/2}$. With the same notation as in Theorem 2, the right hand side of (14) can be improved to the following bound.

$$\left[\frac{2^{22}(q + q' + 1)(\sigma + q)(\ell_N + 1)}{2^n} + \frac{q'(\ell_A + 1)}{2^\tau} \right] \cdot \delta_n(\sigma + q + q' + 1)$$

Note that when $a \ll 2^n$ we have $\delta_n(a) \approx (1 + a^2/2^{n+1})$. It was shown that $\delta_n(a) \leq 1.7$ when $a \leq 2^{64}$ and $n \geq 128$ [13]. Hence, for example, if $1 < q' \leq q \leq \sigma$, $n = \tau = 128$, and $\sigma + q + q' < 2^{64}$, we obtain the bound $(17 \cdot 2^{22}q\sigma\ell_N + 4q'\ell_A)/2^{128}$.

8 Conclusion

In this paper, we studied the security proofs of GCM. We first pointed out that the proofs contain a flaw, and translated the observation into a distinguishing attack that invalidates the main part of the privacy proof. We then showed that the proofs can be repaired by presenting new privacy and authenticity bounds. The bounds are larger than what were previously claimed in a general case of variable length nonces, but they are smaller when the nonce length is restricted to 96 bits. Many standards require or recommend using GCM with 96-bit nonces for efficiency reasons. Our results suggest that restricting GCM to 96-bit nonces is recommended from the provable security perspective as well. This follows [31], where GCM with 96-bit nonces is recommended as the use of variable length nonces increases the proof complexity and the proofs are infrequently verified.

We remark that since our attack only invalidates the second terms of (1) and (2), it does not exclude a possibility that the original security bounds, (1) and (2), can still be proved, and it would be interesting to see if our security bounds can be improved.

Acknowledgments. The authors would like to thank Masato Aikawa for discussions at the initial phase of this work, Yasushi Osaki and Xiangyu Quan for helping searching the counter-examples given in Sect. 5.1 and in Appendix A, Nicky Mouha for verifying the values of r that give $\alpha_r = \alpha_{\max}$ presented in Sect. 6.3, and participants of Dagstuhl Seminar 12031, Symmetric Cryptography, and the anonymous CRYPTO 2012 reviewers for helpful comments. The work by Tetsu Iwata was supported in part by MEXT KAKENHI, Grant-in-Aid for Young Scientists (A), 22680001.

References

1. Bouncy Castle, <http://www.bouncycastle.org/> (accessed on May 26, 2012)
2. Java Platform, Standard Edition 7, <http://docs.oracle.com/javase/7/docs/> (accessed on May 26, 2012)
3. Risa/Asir, <http://www.math.kobe-u.ac.jp/Asir/asir.html> (accessed on May 26, 2012)
4. PKCS #11 v2.20: Cryptographic Token Interface Standard. PKCS #11 v2.20 (2004), <http://www.rsa.com/rsalabs/node.asp?id=2133> (accessed on May 31, 2012)
5. IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Security. IEEE Std 802.1AE-2006 (2006)
6. IEEE Standard for Authenticated Encryption with Length Expansion for Storage Devices. IEEE Std 1619.1-2007 (2007)
7. Information Technology — Security Techniques — Authenticated Encryption, ISO/IEC 19772:2009. International Standard ISO/IEC 19772 (2009)
8. National Security Agency, Internet Protocol Security (IPsec) Minimum Essential Interoperability Requirements, IPMEIR Version 1.0.0 Core (2010), http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml

9. Bellare, M., Kilian, J., Rogaway, P.: The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. Syst. Sci.* 61(3), 362–399 (2000)
10. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
11. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
12. Bernstein, D.J.: Stronger Security Bounds for Permutations (2005), <http://cr.yp.to/papers.html> (accessed on May 31, 2012)
13. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC Security Bound from PRP-Advantage (2005), http://fastcrypto.org/umac/umac_security.pdf (accessed on May 31, 2012)
14. Dai, W.: Crypto++ Library, <http://www.cryptopp.com/> (accessed on May 26, 2012)
15. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007)
16. Ferguson, N.: Authentication Weaknesses in GCM. Public Comments to NIST (2005), <http://csrc.nist.gov/groups/ST/toolkit/BCM/comments.html>
17. Gladman, B.: <http://www.gladman.me.uk/> (accessed on May 26, 2012)
18. Handschuh, H., Preneel, B.: Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 144–161. Springer, Heidelberg (2008)
19. Housley, R.: Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS). IETF RFC 5084 (2007)
20. Igou, K.M., Solinas, J.A.: AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. IETF RFC 5647 (2009)
21. Joux, A.: Authentication Failures in NIST version of GCM. Public Comments to NIST (2006), <http://csrc.nist.gov/groups/ST/toolkit/BCM/comments.html>
22. Leurent, G.: ARXtools: A Toolkit for ARX Analysis. In: The Third SHA-3 Candidate Conference (2012), <http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/index.html>
23. Leurent, G., Thomsen, S.S.: Practical Near-Collisions on the Compression Function of BMW. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 238–251. Springer, Heidelberg (2011)
24. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.* 17(2), 373–386 (1988)
25. McGrew, D.A.: An Interface and Algorithms for Authenticated Encryption. IETF RFC 5116 (2008)
26. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004)
27. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode of Operation (Full Version). Cryptology ePrint Archive, Report 2004/193 (2004), <http://eprint.iacr.org/>
28. McGrew, D.A., Viega, J.: The Galois/Counter Mode of Operation (GCM). Submission to NIST (2005), http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html
29. Mouha, N., Velichkov, V., De Cannière, C., Preneel, B.: The Differential Analysis of S-Functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 36–56. Springer, Heidelberg (2011)

30. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 98–107. ACM (2002)
 31. Rogaway, P.: Evaluation of Some Blockcipher Modes of Operation. Investigation Reports on Cryptographic Techniques in FY 2010 (2011), <http://www.cryptrec.go.jp/english/> (accessed on May 31, 2012)
 32. Saarinen, M.J.O.: Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. Pre-proceedings of FSE 2012 (2012), <http://fse2012.inria.fr/> (accessed on March 17, 2012)
 33. Salowey, J., Choudhury, A., McGrew, D.A.: AES Galois Counter Mode (GCM) Cipher Suites for TLS. IETF RFC 5288 (2008)
 34. Viega, J., McGrew, D.A.: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). IETF RFC 4106 (2005)

A Solutions of (3), (4), and (5), and Examples of (N_1, N_2) Satisfying (6)

In Table 2, Table 3, and Table 4, we show a list of values of L that satisfy (3), (4), and (5), respectively. We see that the 93 values from these lists are all distinct, and are different from 0^n .

The counter-example presented in Sect. 5.2 was found by experimentally searching over the values of U_1 , U_2 , and V . We started by searching over random U_1 , U_2 , and V , and found that the values of the form $U_1 = 0^{8i} \| X \| 0^{n-8-8i}$, $U_2 = 0^{8i} \| Y \| 0^{n-8-8i}$, and $V \in \{0, 1\}^n$ have many examples that satisfy (6), where $X, Y \in \{0, 1\}^8$, $0 \leq i \leq 15$, and $\text{int}(V) = 8j$ for some $i + 1 \leq j \leq 16$ but $j \neq 12$. The examples include the following values:

$$(U_1, U_2, V) = \begin{cases} ((0x0)^{15} \parallel 0x2 \parallel (0x0)^{16}, (0x0)^{15} \parallel 0x6 \parallel (0x0)^{16}, (0x0)^{30} \parallel 0x70) \\ ((0x0)^{17} \parallel 0x2 \parallel (0x0)^{14}, (0x0)^{17} \parallel 0x6 \parallel (0x0)^{14}, (0x0)^{30} \parallel 0x70) \\ ((0x0)^{17} \parallel 0x4 \parallel (0x0)^{14}, (0x0)^{17} \parallel 0xc \parallel (0x0)^{14}, (0x0)^{30} \parallel 0x48) \end{cases}$$

Table 2. List of solutions of (3)

0x7f6db6d2db6db6db6db6db6492492492 0x7f6db6dadbd6b6db6db6492492492
0x81b6db776db6db6db6db6dadbd6b6db6 0x81b6db676db6db6db6db6dadbd6b6db6
0xbe00003c0000000000000003ffffffff 0xbe00001c0000000000000003ffffffff
0xc16db6aadbd6b6db6db6db1b6db6db6d 0xc16db6eadbd6b6db6db6db1b6db6db6d
0x3fb6db876db6db6db6db6d5249249249 0x3fb6db076db6db6db6db6d5249249249
0x000001dc00000000000001c000000000 0x000000dc000000000000001c000000000
0x7f6db56adb6db6db6db6d8e492492492 0x7f6db76adb6db6db6d8e492492492
0x81b6dc076db6db6db6db6aadbd6b6db6 0x81b6d8076db6db6db6db6aadbd6b6db6
0xbe00edc00000000000000e3ffffffff 0xbe006dc00000000000000e3ffffffff
0xc16dab6adb6db6db6db6c71b6db6db6d 0xc16dbb6adb6db6db6db6c71b6db6db6d
0x3fb6e0076db6db6db6db555249249249 0x3fb6c0076db6db6db6db555249249249
0x000076dc00000000000071c000000000 0x000036dc00000000000071c000000000
0x7f6d5b6adb6db6db6db638e492492492 0x7f6ddb6adb6db6db638e492492492
0x81b700076db6db6db6daaaadb6db6db6 0x81b600076db6db6db6daaaadb6db6db6
0xbe03b6dc00000000000038e3ffffffff 0xbe01b6dc00000000000038e3ffffffff
0xc16adb6adb6db6db6db1c71b6db6db6d 0x00000004000000000000000000000000000000

Table 3. List of solutions of (4)

0x7f6db6d6b6db6db6db6db6492492492 0x7f6db6dedb6db6db6db6492492492
0x81b6db736db6db6db6db6dadb6db6db6 0x81b6db636db6db6db6db6dadb6db6db6
0xbe0000380000000000000003fffffff 0xbe0000180000000000000003fffffff
0xc16db6aedb6db6db6db6db1b6db6db6d 0xc16db6eedb6db6db6db6db1b6db6db6d
0x3fb6db836db6db6db6db6d5249249249 0x3fb6db036db6db6db6db6d5249249249
0x000001d800000000000001c000000000 0x000000d800000000000001c000000000
0x7f6db56edb6db6db6db6d8e492492492 0x7f6db76edb6db6db6d8e492492492
0x81b6dc036db6db6db6db6aadb6db6db6 0x81b6d8036db6db6db6db6aadb6db6db6
0xbe00ed80000000000000e3fffffff 0xbe006d80000000000000e3fffffff
0xc16dab6edb6db6db6b6c71b6db6db6d 0xc16dbb6edb6db6db6db6c71b6db6db6d
0x3fb6e0036db6db6db6db555249249249 0x3fb6c0036db6db6db6db555249249249
0x000076d800000000000071c000000000 0x000036d8000000000000071c000000000
0x7f6d5b6edb6db6db6db638e492492492 0x7f6ddb6edb6db6db6db638e492492492
0x81b700036db6db6db6daaaadb6db6db6 0x81b600036db6db6db6db6daaaadb6db6db6
0xbe03b6d80000000000038e3fffffff 0xbe01b6d80000000000038e3fffffff
0xc16adb6edb6db6db6b1c71b6db6db6d

Table 4. List of solutions of (5)

0xbe076db800000000000071c7fffffff 0xc16c000edb6db6db6db5555b6db6db6d
0xc16e000edb6db6db6db5555b6db6db6d 0xfedb6db5b6db6db6db6c71c924924924
0xfedab6d5b6db6db6db6c71c924924924 0x00006db8000000000000e38000000000
0x0000edb8000000000000e380000000000 0x7f6d800edb6db6db6db6aaa492492492
0x7f6dc00edb6db6db6db6aaa492492492 0x40db76d5b6db6db6db6d8e36db6db6d
0x40db56d5b6db6db6db6d8e36db6db6d 0xbe000db8000000000001c7fffffff
0xbe001db8000000000000001c7fffffff 0xc16db00edb6db6db6db6d55b6db6d6d
0xc16db80edb6db6db6db6d55b6db6db6d 0xfedb6ed5b6db6db6db6db6d1c924924924
0xfedb6ad5b6db6db6db6db1c924924924 0x000001b80000000000000038000000000
0x000003b80000000000000038000000000 0x7f6db60edb6db6db6daaa492492492
0x7f6db70edb6db6db6db6daaa492492492 0x40db6dd5b6db6db6db6d36db6db6d
0x40db6d55b6db6db6db6db6d636db6db6d 0xbe0000380000000000000007ffffffffff
0xbe0000780000000000000007ffffffffff 0xc16db6cedb6db6db6db5b6db6db6d
0xc16db6eedb6db6db6db6db5b6db6db6d 0xfedb6db5b6db6db6db6db6c924924924
0xfedb6da5b6db6db6db6db6db6c924924924 0x00000008000000000000000000000000

These values are equivalent to $(0^{60} \parallel 0010 \parallel 0^{64}, 0^{60} \parallel 0110 \parallel 0^{64}, 0^{120} \parallel 01110000)$, $(0^{68} \parallel 0010 \parallel 0^{56}, 0^{68} \parallel 0110 \parallel 0^{56}, 0^{120} \parallel 01110000)$, and $(0^{68} \parallel 0100 \parallel 0^{56}, 0^{68} \parallel 1100 \parallel 0^{56}, 0^{120} \parallel 01001000)$ in binary. N_1 and N_2 are the most significant $\text{int}(V)$ bits of U_1 and U_2 , respectively.

On the Distribution of Linear Biases: Three Instructive Examples^{*}

Mohamed Ahmed Abdelraheem¹, Martin Ågren²,
Peter Beelen¹, and Gregor Leander¹

¹ Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark
`{M.A.Abdelraheem,P.Beelen,G.Leander}@mat.dtu.dk`

² Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118, SE-221 00 Lund, Sweden
`martin.agren@eit.lth.se`

Abstract. Despite the fact that we evidently have very good block ciphers at hand today, some fundamental questions on their security are still unsolved. One such fundamental problem is to precisely assess the security of a given block cipher with respect to linear cryptanalysis. In by far most of the cases we have to make (clearly wrong) assumptions, e.g., assume independent round-keys. Besides being unsatisfactory from a scientific perspective, the lack of fundamental understanding might have an impact on the performance of the ciphers we use. As we do not understand the security sufficiently enough, we often tend to embed a security margin – from an efficiency perspective nothing else than wasted performance. The aim of this paper is to stimulate research on these foundations of block ciphers. We do this by presenting three examples of ciphers that behave differently to what is normally assumed. Thus, on the one hand these examples serve as counter examples to common beliefs and on the other hand serve as a guideline for future work.

1 Introduction

IT Security plays an increasingly crucial role in everyday life and business. When talking on a mobile phone, when withdrawing money from an ATM or when buying goods over the internet, security plays a crucial role in both protecting the user and in maintaining public confidence in the system. Moreover, security techniques are often an enabler for innovative business models, e.g., iTunes and the Amazon Kindle require strong copyright protection mechanisms, or after-sale feature activation in modern cars. Virtually all modern security solutions are based on cryptographic primitives. Block ciphers are arguably the most widely used type of these primitives.

State-of-the-Art of Block Ciphers. While great progress has been made in designing and analyzing block ciphers, fundamental aspects of these ciphers are still not understood.

* The full version of this paper is available at ePrint.

Besides being unsatisfactory from a scientific perspective, the lack of fundamental understanding might have consequence on the performance of the ciphers we use. As we do not understand the security sufficiently, we tend to embed a security margin in the ciphers we are using. From an efficiency perspective, a security margin is nothing else than wasted performance. While for some applications this might not be critical, for others it certainly is. In particular, when it comes to cryptography in the emerging field of pervasive computing, the computational resources of the devices in question are often highly constrained, and we can only allow close to zero overhead for a security margin.

Especially for the key-scheduling algorithm, the fundamental part of a cipher that is responsible for generating key-material from a master-key to be used at several places in the algorithm (see Figure 1 for an example), simplifying assumptions are standard. While these assumptions are strictly speaking wrong, the hope is that the behaviour of the real cipher does not differ significantly from the simplified model.

Linear Cryptanalysis. One of the best known and most general applicable attacks on block ciphers is Matsui’s linear attack [15]. Since its introduction many extensions and improvements have been made, and we mention a selection here. A more precise estimate for the success probability and the data complexity are given by Selçuk [21]. The effect of using more than one linear trail, referred to as linear hulls, has been introduced by Nyberg [17]; see also Daemen and Rijmen [8]. This has been used for example by Cho [6]. Multi-dimensional linear attacks have been studied by Hermelin, Cho, and Nyberg [10] as a way to further reduce the data complexity of the basic attack. We also like to mention the critics on the concept of linear hulls by Murphy [16]; see Leander [13] for a further discussion. One of the most recent developments is the idea to make use of unbiased linear approximations by Bogdanov et al. [4].

However, despite its discovery more than 15 years ago, and the many extensions introduced since then some very fundamental properties are not yet well understood.

In a nutshell, for claiming a cipher secure against linear attacks, one has to demonstrate that the cipher does not possess certain statistical irregularities. In almost all cases the best we can do is to bound the correlation of a single linear trail (see [20] for an exception), as this roughly corresponds to bounding the number of active Sboxes. Thus, using for example the well established wide-trail strategy used in AES [8], obtaining strong bounds on the correlation of a single trail is quite easy nowadays. However, when it comes to bounding the correlation of a linear approximation, or linear hull to emphasize its relation to many linear trails, no general convincing arguments are available. More precisely, the task of understanding the distribution of linear correlations over the keys is unsolved.

In order to be able to do so, it is in by far most of the cases necessary to assume that all (round) keys are independently and uniformly chosen or make the even stronger (and clearly wrong) assumption that distinct linear trails are independent.

While independent round-keys are hardly ever used in any real cipher, this assumption is on the one hand needed to make the analysis feasible and on the other hand often does not seem problematic as even with the keys not independently and uniformly chosen, most ciphers (experimentally) do not behave different from the expectation.

However, those experimental confirmations that the cipher behaves as assumed are inherently insufficient. For a 128 bit block cipher a single linear approximation that for a fraction of 2^{-30} of all keys has a correlation greater than 2^{-30} is something that, on the one hand, we clearly want to avoid but, on the other hand, we will never discover by experiments only.

Thus, it is important to really understand the distribution of bias, where the distribution is taken over all possible keys. Only by studying the entire distribution can weak keys possibly be identified (this has been pointed out previously, cf. for example [8]). Promising results along these lines include the work of Daemen and Rijmen [9] where the problem is clearly stated and attempts are made to give general statements. Unfortunately, as we will discuss below, one of the most general theorems is strictly speaking wrong.

Our Contribution. The aim of this paper is to stimulate research on the foundations of block ciphers. We do this by presenting three examples of ciphers that behave differently to what is normally assumed. Thus, on the one hand these examples serve as counter examples to common beliefs and on the other hand serve as a guideline for future work. The value of our examples as guidelines for future work is specific for each example. The first example mainly limits the most general statements one can hope to prove, and in particular is a counter example to Theorem 22 in [9] where under rather natural conditions it was stated that the distribution of correlations is well approximated by a normal approximation and in particular one should expect many different possible values for the bias. The second example considers the influence of the key scheduling on the distribution of correlations. Here the variance (but not the shape) of the distribution significantly depends on the key-scheduling. For future work this suggests that highly non-linear key scheduling algorithms are superior to linear ones with respect to the distribution of correlations. Here highly non-linear has to be understood not as a vague criteria but in terms of minimizing the absolute values for all linear approximations. The last example is again related to key-scheduling but more so to symmetries in ciphers. We show a general equivalence of symmetries and linear approximations for weak-keys that exist for any number of rounds and illustrate this fact with an example.

The techniques used to analyze these examples are surprisingly diverse and of independent interest.

In order to facilitate the understanding of our examples without diving into too many details we give only an overview of the results of the first and the third example in Section 3. The details for the first example are postponed to Section 4 and for the third example to Section 5.

2 Notation and Preliminaries

Given an n bit function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, a *linear approximation* is an equation of the form

$$\langle \alpha, x \rangle + \langle \beta, F(x) \rangle = 0,$$

where $\langle \cdot, \cdot \rangle$ denotes an inner product. The vector α is called the input mask and β is the output mask. The *bias* $\epsilon_F(\alpha, \beta) \in [-1/2, 1/2]$ of a linear approximation is defined as

$$\text{Prob}(\langle \alpha, x \rangle + \langle \beta, F(x) \rangle = 0) = \frac{1}{2} + \epsilon_F(\alpha, \beta),$$

where the probability is taken over all inputs x . The bias is the value of major importance for linear attacks. However, due to scaling reasons, it is much more convenient to work with the *correlation* $c_F(\alpha, \beta) \in [-1, 1]$ defined by

$$c_F(\alpha, \beta) = 2\epsilon_F(\alpha, \beta).$$

Another measure that we are going to use is the Fourier-transformation of F ,

$$\widehat{F}(\alpha, \beta) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \beta, F(x) \rangle + \langle \alpha, x \rangle}.$$

Up to scaling $\widehat{F}(\alpha, \beta)$ is equivalent to the bias $\epsilon_F(\alpha, \beta)$ and the correlation $c_F(\alpha, \beta)$. More precisely,

$$\epsilon_F(\alpha, \beta) = \frac{c_F(\alpha, \beta)}{2} = \frac{\widehat{F}(\alpha, \beta)}{2^{n+1}}. \quad (1)$$

Linear Trails and Linear Hull. Given a composite function F , i.e., $F_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ such that $F = F_r \circ \dots \circ F_2 \circ F_1$, a *linear trail* θ is a collection of all intermediate masks

$$\theta = (\theta_0 = \alpha, \dots, \theta_r = \beta)$$

and its correlation is defined by

$$C_\theta = \prod_i c_{F_i}(\theta_i, \theta_{i+1}).$$

It is well known, see e.g., [8], that the correlation of a linear approximation is the sum of all correlations of linear trails starting with the same mask α and ending with the same mask β , i.e.,

$$c_F(\alpha, \beta) = \sum_{\theta \mid \theta_0 = \alpha, \theta_r = \beta} C_\theta. \quad (2)$$

In this paper we are concerned with keyed permutations, more precisely with key-alternating ciphers as defined for example in [8, Section 2.4.2] and depicted

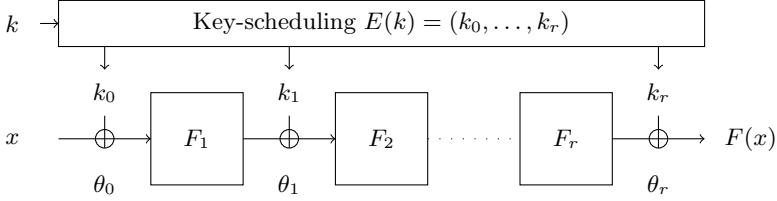


Fig. 1. A key-alternating cipher

in Fig. 1. An n bit key-alternating cipher with a k bit (master) key consists of round functions $F_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and a key-scheduling algorithm $E : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^{n(r+1)}$.

The dependence of the correlation of a linear trail is conceptually very simple, only the sign of the correlation depends on the key. More precisely,

$$C_\theta = (-1)^{\langle \theta, E(k) \rangle} \prod_i c_{F_i}(\theta_i, \theta_{i+1}).$$

Plugging this into Equation (2) leads to the following result.

$$c_F(\alpha, \beta) = \sum_{\theta \mid \theta_0=\alpha, \theta_r=\beta} (-1)^{\langle \theta, E(k) \rangle + \sigma_\theta} |C_\theta| \quad (3)$$

It is exactly this equation that is often referred to as the *linear hull*: The correlation of a linear approximation is the key-dependent signed sum of the correlation of all trails.

In this work we are interested in the distribution over the keys of the linear correlation. Here one can think of each linear trail C_θ as a random variable with a fixed absolute value that with probability $1/2$ is positive and with probability $1/2$ is negative. In this setting the linear hull is the sum of those random variables.

While in general not a lot is known about this distribution, two important characteristics can be stated, assuming independent round-keys. First, as the average of a sum of random variables is the sum of the averages, the average bias is zero. Here, independent round-keys are used to ensure that each single trail has average zero. Moreover, it is easy to see that two distinct linear trails C_θ and C'_θ are pairwise independent. It follows (cf. Theorem 7.9.1 in [8]) that with independent round-keys the variance of the distribution, i.e., the average square correlation, is the sum of the squares of the correlations of all trails. We summarize this in the following proposition.

Proposition 1. *Assuming independent round-keys, i.e., $k = n(r + 1)$ and $E(k) = k$, the average correlation is zero, i.e.,*

$$\mathbb{E}(C_\theta) = 0.$$

Moreover, the average square correlation is given by

$$\mathbb{E}(C_\theta^2) = \sum_i c_{F_i}(\theta_i, \theta_{i+1})^2.$$

Finally, we already note here an observation that we will make use of later.

Lemma 1. *If the key-scheduling $E : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^{n(r+1)}$ is linear then two distinct linear trails C_θ and C'_θ are either independent or $C_\theta = \pm c \cdot C'_\theta$ for a constant c . More precisely C_θ and C'_θ are independent if and only if $E^*(\theta + \theta') \neq 0$ where E^* is the adjoint linear mapping.*

Proof. The lemma follows directly from the observation that

$$\langle \theta, E(k) \rangle + \langle \theta', E(k) \rangle = \langle E^*(\theta), k \rangle + \langle E^*(\theta'), k \rangle = \langle E^*(\theta + \theta'), k \rangle$$

and the general remark that a linear function $\ell(\cdot) = \langle a, \cdot \rangle$ is either balanced (if $a \neq 0$) or constant (if $a = 0$). Thus two trails are independent if and only if $E^*(\theta + \theta') \neq 0$. \square

3 Our Results

In this section we briefly describe our examples, the results and their interpretation. As the first and the last example require more technical details for a full explanation, the exact analysis of those results are given in later sections.

3.1 Example I: The CUBE-Cipher

As a first example, we study the two round key-alternating cipher depicted in Fig. 2 with block size n , n odd. The round function $x \rightarrow x^3$ has to be understood as a mapping on the finite field \mathbb{F}_{2^n} with 2^n elements (as n is odd this is a bijection). The key consists of three independent subkeys k_0, k_1, k_2 each of n bits. Obviously, and for various reasons [12, Section 8.4], this is an artificial example of a block cipher. However, for the purpose of this counterexample that does not matter – it is a counterexample anyway. Moreover, as this cipher clearly belongs to the class of key-alternating ciphers, general theorems on these have to either explicitly exclude this (and similar) examples or the statements have to cover this strange behavior as well.

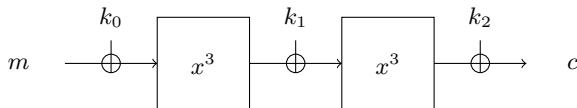


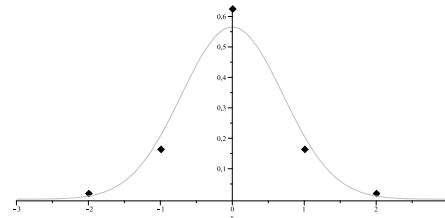
Fig. 2. The CUBE-cipher

As we will prove in Theorem 2, the number of trails is very large. Namely roughly 2^{n-2} trails with non-zero correlation exist. Furthermore, all trails have

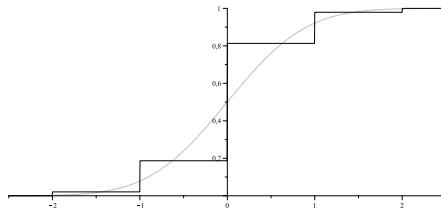
the same absolute correlation. This seems the ideal situation (we even have a parameter, n , that could go to infinity) for assuming that the distribution of correlations over the keys is very well approximated by a normal distribution, cf. Theorem 22 in [9].

Theorem 1 (Theorem 22 of [9]). *Given a key-alternating cipher with independent round-keys. If the number of linear trails with non-zero correlation is large and the square of the correlation of each linear trail is small compared to the variance of the distribution then the distribution is well approximated by a normal distribution.*

The intuition why a normal distribution should be a good approximation is that in this case the linear hull is the sum of a huge amount of pairwise(!) independent random variables. However, it turns out to be wrong. In particular, for any n , the correlation of the cipher actually takes only 5 different values. Thus, the roughly 2^{n-2} random variables are not independent at all. As an example of the real distribution compared to the assumed normal distribution we consider the case $n = 31$ (other cases behave very similarly). Figs. 3(a) and 3(b) show both distributions and make clear that the normal approximation is not a very good approximation and in particular does not get substantially better when n increases. In Section 4 we prove that in general only 5 values are obtained and we furthermore study the exact distribution of those 5 values.



(a) The (normalized) distribution of the CUBE-cipher vs the normal distribution



(b) The (normalized) cumulative probability distribution of the CUBE-cipher vs the normal distribution

Fig. 3. (Normalized) distributions of the CUBE-cipher vs the normal distribution

3.2 Example II: PRESENT with Identical Round-Keys

Our second example is related to the block cipher PRESENT by Bogdanov et al., see [3] for details. As was previously shown, e.g., by Ohkuma [19], for an increasing number of rounds PRESENT exhibits many linear trails with only one active Sbox per round. Due to the design criteria of the Sbox, it follows that all those trails have the same linear bias. Besides, those trails are the ones with a maximal correlation (in absolute terms).

It was experimentally confirmed in [19] that the distribution of the correlation nicely follows a normal-distribution with mean zero and variance $2^{(-2r-1)^2} N$ where N is the number of the linear trails with only one active Sbox per rounds. Thus, experimentally, we can notice two facts: Firstly, for PRESENT different trails behave like independent random variables (in contrast to the CUBE-cipher) and secondly, the contribution of the non-optimal trails does not influence the distribution significantly.

Let us now come to a variant of PRESENT with identical round-keys¹ (and round-constants to avoid trivial slide attacks [1]). As it turns out this is an intriguing example of the influence of the key-scheduling on the distribution of the correlations. We started by performing experiments on a large number of random keys and observed that the total variance of the bias distribution for some linear approximations of PRESENT with identical round-keys is consistently bigger than that of standard PRESENT for any number of rounds ≥ 5 . Fig. 4 shows the distribution of the linear correlation for the identical round-keys case vs. the original PRESENT key-scheduling for 17 rounds.

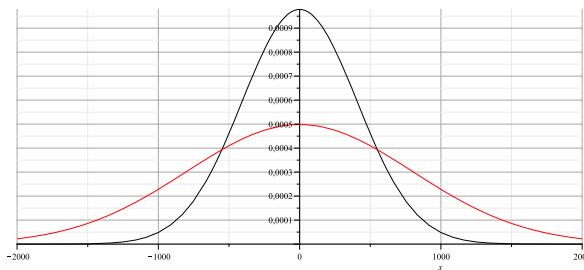


Fig. 4. The (normalized) probability distribution of the PRESENT-cipher with the usual vs the identical round-keys case

The difference is significant in the sense that more rounds of PRESENT with identical round-keys are vulnerable to linear attacks for a non-negligible fraction of keys. In other words, in this example it is indeed the choice of the key-scheduling that makes the cipher secure or insecure against linear cryptanalysis.

¹ Note that identical round-keys have been used before, see for example the block cipher NOEKEON [7].

To illustrate the difference consider a 20 round version. The fraction of keys with a squared bias larger than 2^{-55} is 33.7% in the case of identical round-keys but only 1.1% in the case of the standard PRESENT key-scheduling.

For computing the variance of a sum of random variables it is sufficient to study the pairwise covariance of the summands. Now, as mentioned above in Lemma 1 for a linear key-scheduling algorithm there are only two possibilities for the covariance. Either two trails are independent or identical up to a constant factor. In our particular case this constant is either 1 or -1 as all trails we consider have the same absolute correlation. Note furthermore that, following Lemma 1, two trails C_θ and C'_θ are identical (up to a constant ± 1) iff $E^*(\theta + \theta') = 0$ where E denotes the key-scheduling function. For identical round-keys, we have that $E(k) = (k, \dots, k)$ and thus

$$E^*(\theta) = E^*(\theta_0, \dots, \theta_r) = \sum_{i=0}^r \theta_i.$$

In other words two trails are identical iff the (xor) sums of all intermediate masks are identical. While in general, computing the number of trails is much more efficient than listing all trails, it is still feasible for $r \leq 20$ to compute the list of trails and sort this list according to the sum of the intermediate masks. Thus, for $r \leq 20$ we can relatively easily compute the expected variance for the PRESENT-variant with identical round-keys. Table 1 shows the expected variance (Var_2) of the bias distribution of the optimal linear approximation for a specific one bit input and output difference. For PRESENT with identical round-keys the expected variance is very close to the observed variance (ObsVar) sampled over 20000 random keys. Table 1 also shows the expected variance of the bias distribution of the optimal linear approximation of (standard) PRESENT (Var_1) along with the number of trails (N_1) with one active Sbox per round, and the number of trails (N_2) where the sign depends on the keybits, for number of rounds r , $15 \leq r \leq 20$.

Table 1. Analytical and experimental data on r -round reduced PRESENT (possibly with identical round-keys). N_1 is the number of all linear trials with one active Sbox. N_2 is the number of trails (among N_1) that don't behave the same (their absolute values are equal but the correlation sign is different and it changes according to the subkeys). Var_1 gives the expected variance of the bias of the optimal linear approximation of (standard) PRESENT, while Var_2 corresponds to PRESENT with identical round-keys. ObsVar is the experimentally observed variance sampled over 20000 random keys.

r	N_1	N_2	$\log_2(\text{Var}_1)$	$\log_2(\text{Var}_2)$	$\log_2(\text{ObsVar})$
15	166375	12016	-44.66	-42.71	-42.71
16	435600	20039	-47.26	-45.15	-45.28
17	1140480	31799	-49.88	-47.63	-47.61
18	2985984	48223	-52.49	-50.03	-50.12
19	7817472	69528	-55.10	-52.50	-52.52
20	20466576	95125	-57.71	-54.88	-54.92

It is important to note that, while the CUBE-cipher is certainly an artificial design, PRESENT with identical round-keys is not. In this context we like to mention that it is precisely the behavior described here that resulted in the need to choose a different Sbox in the PRESENT-inspired sponge-based hash-function SPONGENT by Bogdanov et al. [2]. SPONGENT can be seen as a fixed key and large block size variant of PRESENT with identical round-keys.

3.3 Example III: PRINTCIPHER or Block Ciphers with Symmetries

It was already pointed out by Leander et al. [14] that for PRINTCIPHER-48 [11] by Knudsen et al. there exist strongly biased linear relations for any number of rounds. More precisely,

Proposition 2 (Corollary 2 in [14]). *For a fraction of 2^{-28} of all keys and for any round $r \leq 48$ there exists at least one linear approximation for PRINTCIPHER-48 with correlation at least $2^{-16} - 2^{-32}$.*

Here (cf. Section 5), we extend upon this analysis by showing an equivalence between a submatrix A of the correlation matrix that has an eigenvector with eigenvalue one and a round function that has an invariant subspace. This is crucial as this implies that this sub-matrix A , when taken to the r -th power, does not converge to the all-zero matrix. In particular in the case where there is a unique eigenvector with eigenvalue of norm 1, A^r converges to a non-zero constant. *This is equivalent to saying that trails with all intermediate masks determined by the invariant subspace cluster significantly for any number of rounds.*

Note that an invariant subspace in particular captures, as a special case, what is usually referred to as symmetries. Thus, besides PRINTCIPHER one could also imagine an identical-round-key variant of AES with round constants that do not destroy the symmetries introduced by the very structured and byte oriented linear layer of AES. This example reveals two interesting points. First, in such a situation of trail clustering, increasing the number of rounds *does not* help and secondly without the link to invariant subspaces it seems very hard to understand why certain trails should cluster even for an AES-like design that follows the wide-trail strategy. Moreover this clustering is not inherently limited to ciphers with weak mixing (e.g., PRINTCIPHER), but is rather a problem for all ciphers exhibiting symmetries.

Interestingly, in a restricted sense to be discussed in Section 5, the reciprocal statement holds as well. That is, if the cipher does not exhibit symmetries, then no sub-matrices (of a certain type) have eigenvectors (of a certain type) with eigenvalue 1. Thus by avoiding symmetries one also ensures that trail clustering for any number of rounds is highly unlikely.

Fig. 5 shows the difference of the distribution of the correlation for non-weak keys vs. the distribution for weak keys for a 24-bit version of PRINTCIPHER. Both graphs can be nicely approximated by normal distributions, however, the mean of the distribution for weak keys differ significantly from the origin.

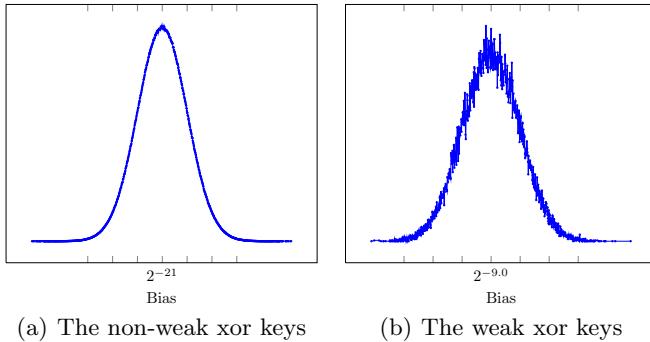


Fig. 5. The distribution of PRINTCIPHER-24 biases for a fixed permutation key. The experimentally observed means m are indicated. In both distributions, the standard deviation σ is approximately $2^{-13.0}$. Ticks have been placed at $m+k\sigma$, $k \in \{-3, \dots, 3\}$.

4 Example I: The CUBE-Cipher

In this section we give a detailed analysis of the distribution of the correlations in the CUBE-cipher.²

We denote the block size by n , where in this example n has to be odd. First note that the initial and the final key-addition do not change the distribution. Thus, to simplify notation, we ignore them from now on. We therefore have to consider the function $F_k(x) = (x^3 + k)^3$. Moreover, to make the analysis easier, we focus on the input and output mask $1 \in \mathbb{F}_{2^n}$. That is, we are interested in the distribution of $\widehat{F}_k(1, 1)$ for varying key k .

As a first step we show that the corresponding linear hull contains a very large number of trails with non-zero correlations. More precisely, the following holds (cf. the full version for the proof).

Theorem 2. *The number t of trails with non-zero correlation of the form*

$$1 \xrightarrow{x^3} \alpha \xrightarrow{x^3} 1$$

is

$$t = \frac{2^n + 1 - (a_1^n + a_2^n + a_3^n + a_4^n)}{4},$$

where a_i are the four (complex) roots of the polynomial $x^4 + x^3 + 2x + 4$. Furthermore, each trail has a correlation of $\pm 2^{1-n}$.

The next proposition shows that, despite the huge number of non-zero trails, only 5 values occur for the correlations.

² Due to page limitations most of the proofs are given in the full version.

Proposition 3. $\widehat{F}_k(1, 1) \in \{0, \pm 2^{\frac{n+1}{2}}, \pm 2^{\frac{n+3}{2}}\}$

Proof. We denote by $\mu(x) = (-1)^{\text{Tr}(x)}$, where $\text{Tr}(x) = x + x^2 + x^4 + \dots x^{2^{n-1}}$ is the trace mapping and note that $\text{Tr}(xy)$ is the natural inner product on \mathbb{F}_{2^n} .

$$\begin{aligned}\widehat{F}_k(1, 1)^2 &= \sum_{x, y} \mu((x^3 + k)^3 + (y^3 + k)^3 + x + y) \\ &= \sum_{x, y} \mu(((x + y)^3 + k)^3 + (y^3 + k)^3 + x) \\ &= \sum_x \mu((x^3 + k)^3 + x + k^3) \sum_y \mu((x^{64} + (k^{16} + k^4)x^{16} + (k^8 + k^2)x^4 + x) y^8) \\ &= 2^n \sum_{x \in M} \mu((x^3 + k)^3 + x + k^3)\end{aligned}$$

where

$$M = \{x \mid x^{64} + (k^{16} + k^4)x^{16} + (k^8 + k^2)x^4 + x = 0\}.$$

Thus, we have to understand the kernel (and in particular its size) of the \mathbb{F}_2 -linear mapping

$$P(x) = x^{64} + (k^{16} + k^4)x^{16} + (k^8 + k^2)x^4 + x.$$

As a polynomial, P splits nicely into factors, i.e.,

$$P(x) = A_1(x^3)A_2(x^3)A_3(x^3)A_4(x^3)x,$$

with

$$\begin{aligned}A_1(x) &= x^3 + k^2x + 1 & A_3(x) &= x^6 + x^4 + (k^4 + k^2 + 1)x^2 + x + 1 \\ A_2(x) &= x^3 + (k^2 + 1)x + 1 & A_4(x) &= x^9 + x^3 + (k^8 + k^2)x + 1.\end{aligned}$$

For now, we show that $A_3(x)$ does not have any roots. Note that this is actually enough to prove the proposition, as this upper bounds the number of elements in M by $16 = 9 + 3 + 3 + 1$ and for general reasons we know that $|M| = 2^i$ with i odd. Thus $|M| \in \{2, 8\}$. Assume that $A_3(x) = 0$. Then

$$k^4 + k^2 + 1 = \frac{x^6 + x^4 + x + 1}{x^2} = x^4 + x^2 + \frac{1}{x} + \frac{1}{x^2}.$$

Applying the trace mapping to both sides implies $\text{Tr}(1) = 0$.³ A contradiction, as n is odd. \square

This already demonstrates an unexpected behavior. The following theorem, proven in the full version of the paper, allows us to compute the exact distribution of the 5 occurring values for reasonably large n .

³ Note that $\text{Tr}(x) = \text{Tr}(x^2)$ for all $x \in \mathbb{F}_{2^n}$.

Theorem 3. *We have*

$$\#\{k \in \mathbb{F}_{2^n} \mid \widehat{F}_k(1,1)^2 = 2^{n+3}\} = \frac{1}{3} \#\{\beta \in \mathbb{F}_{2^n} \mid \beta \text{ satisfies Eqns. (4)}\},$$

where

$$\mathrm{Tr} \left(\left(\frac{1}{(\beta^2 + \beta)} \right)^{1/9} \right) = \mathrm{Tr} \left(\left(\frac{\beta^3}{\beta^2 + \beta} \right)^{1/9} \right) = \mathrm{Tr} \left(\left(\frac{(1+\beta)^3}{\beta^2 + \beta} \right)^{1/9} \right) = 1. \quad (4)$$

The advantage of the above theorem is that it gives a fast way to compute the number A of $k \in \mathbb{F}_{2^n}$ such that $\widehat{F}_k(1,1) = 2^{n+3}$. Let us further denote by B the number of $k \in \mathbb{F}_{2^n}$ such that $\widehat{F}_k(1,1) = 2^{n+1}$. Then clearly

$$\sum_k \widehat{F}_k(1,1)^2 = A2^{n+3} + B2^{n+1}.$$

On the other hand, since knowing the number of trails with non-zero correlation together with their correlation values, corresponds to knowing the average square correlation, we have

$$\sum_k \widehat{F}_k(1,1)^2 = 2^{-n} \sum_{\alpha} \widehat{C}(1,\alpha)^2 \widehat{C}(\alpha,1)^2.$$

Using the above and Theorem 2 we see that

$$A2^{n+3} + B2^{n+1} = \sum_k \widehat{F}_k(1,1)^2 = 2^n(2^n + 1 - a_1^n - a_2^n - a_3^n - a_4^n),$$

where, as in Theorem 2, a_i are the four (complex) roots of the polynomial $x^4 + x^3 + 2x + 4$. Thus using Proposition 3 and the symmetry of the distribution (which can easily be proven in this case) one can now obtain the complete distribution for how many k , $F_k(1,1)$ obtains a particular value in $\{\pm 2^{(n+3)/2}, \pm 2^{(n+1)/2}, 0\}$ for reasonably large values of n . We give some examples below.

n	$-2^{(n+3)/2}$	$-2^{(n+1)/2}$	0	$2^{(n+1)/2}$	$2^{(n+3)/2}$
1	0	1	0	1	0
3	1	0	6	0	1
5	0	6	20	6	0
9	10	90	312	90	10
19	10868	87078	328396	87078	10868
31	44732008	357939982	1342139668	357939982	44732008

5 Example III: PRINTCIPHER or Block Ciphers with Symmetries

The results in this section, especially as summarized in Theorem 5, are quite general, applying to any block cipher (permutation) exhibiting these kinds of

symmetries. For this reason, we do not describe PRINTCIPHER in detail here, but refer to the full version or [11] instead. We only note here that the round-key is the same in every round (there is a small round constant).

PRINTCIPHER is used for experiments in Section 3.3 and below. Smaller-state versions are not formally defined but are easy to extrapolate from [11]. We use the same round constants as in the first rounds of PRINTCIPHER-48. A PRINTCIPHER-key can be split into a permutation key and an xor key.

The Invariant Subspace. Let us define a subspace $U \subset \mathbb{F}_2^n$, the orthogonal subspace $U^\perp = \{y : \langle x, y \rangle = 0, \forall x \in U^\perp\}$ and a constant $d \in \mathbb{F}_2^n$. Then, the invariant subspace property (cf. [14]) can be expressed as $F_i(U + d) = U + d$. In the case of PRINTCIPHER, the exact definitions of U , U^\perp , and d can be found in the full version of the paper. We only note that for PRINTCIPHER, $|U + d| = 2^{16}$, and the trails do not involve the round constants so the invariant subspace extends to the entire cipher F , regardless of the number of rounds. However, even if an invariant subspace only occurs for some rounds of the cipher, it can certainly be interesting in linear cryptanalysis as seen below.

Understanding the Large Correlations. The correlation matrix (cf. [8]) $M_i = (c_{F_i}(\alpha, \beta))_{\alpha\beta}$ collects all correlation coefficients for a single round. We are interested in the submatrix $A = (a_{\alpha\beta})_{\alpha, \beta \in U^\perp}$ constructed through $a_{\alpha\beta} = c_{F_i}(\alpha, \beta)$ and its powers A^r . Thus A collects the correlations where input and output masks only involve the bits that govern the invariant subspace. In any correlation matrix, the first row and column are all-zero except for $c(0, 0) = 1$. We extract the sub-matrix $B = (a_{\alpha\beta})_{\alpha, \beta \in U^\perp \setminus \{0\}}$, since it will be slightly more convenient to use.

We should identify A_i with F_i , but the round constants do not affect A_i , so all A_i are equal. In particular $A_r A_{r-1} \dots A_1 = A^r$. Note how A^r describes the contribution to the linear hull from following trails with intermediate masks in U^\perp . More specifically, we can write Equation (3) as

$$c_F(\alpha, \beta) = \sum_{\theta \mid \theta_0 = \alpha, \theta_r = \beta, \theta_i \in U^\perp, \forall i} (-1)^{\langle \theta, E(k) \rangle + \sigma_\theta} |C_\theta| + \sum_{\theta \mid \theta_0 = \alpha, \theta_r = \beta, \exists i: \theta_i \notin U^\perp} (-1)^{\langle \theta, E(k) \rangle + \sigma_\theta} |C_\theta|,$$

where the first sum corresponds to element (α, β) of A^r . If elements of A^r have a large magnitude, then the corresponding elements of M^r have (at least) the same magnitude, unless the contributions from trails that go outside U^\perp (essentially) cancel those from inside.

We now examine the asymptotic behaviour of A^r . Define $v = (v_\alpha)_{\alpha \in U^\perp}$ by $v_\alpha = (-1)^{\langle d, \alpha \rangle}$.

Lemma 2. v^T is an eigenvector to A with eigenvalue 1, i.e., $Av^T = v^T$.

We prove this lemma in the full version of the paper.

Now, in the case where there is no other (non-trivial) eigenvector with eigenvalue 1 the sequence A^r will converge (see the theorem below). This motivates the following definition.

Definition 1. *If the algebraic multiplicity of A 's eigenvalue 1 is two and A has no other eigenvalue of absolute value 1, we say that A (or the corresponding cipher) has a stable symmetry. (The eigenvectors are that given in Lemma 2, and the vector $(1, 0, 0, \dots, 0)^T$.)*

For the following theorem, we use that A has eigenvalues with absolute value at most 1, the Schur decomposition of A and the relation between convergence of A^r and the spectrum of A [5].

Theorem 4. *If A has a stable symmetry then $B^r \rightarrow C = \frac{1}{2^{\dim(U)} - 1} u^T u$, $r \rightarrow \infty$, $u = (v_\alpha)_{\alpha \in U^\perp \setminus \{0\}}$.*

If other contributions to $c_F(\alpha, \beta)$ are negligible, then all characteristics with non-zero $\alpha, \beta \in U^\perp$ have $c_F(\alpha, \beta) \approx \pm 2^{-\dim(U)}$ so bias $\epsilon_F(\alpha, \beta) \approx \pm 2^{-\dim(U)-1}$.

Equivalence between Eigenvectors and Invariant Subspaces. With the following theorem, which we prove in the full version, we establish a loose relation between symmetries in block ciphers and susceptibility to linear cryptanalysis. In the case of PRINTCIPHER, this was a negative result, but in case of block ciphers without symmetries, it is positive.

Theorem 5. *Consider an invertible vectorial Boolean function F , a subspace U , the orthogonal subspace U^\perp and a vector d . Define $A = (a_{\alpha\beta})_{\alpha, \beta \in U^\perp}$ and $v = (v_\alpha)_{\alpha \in U^\perp}$, $v_\alpha = (-1)^{\langle d, \alpha \rangle}$. Then $Av^T = v^T$ if and only if $F(U + d) = U + d$.*

Experimental Results on PRINTCIPHER. We have implemented PRINTCIPHER-48 for a key from the class of weak keys used as the main example in [14]. This allowed us to derive A and verify that $Av^T = v^T$. We could also derive the biases for 16 characteristics with $\alpha, \beta \in U^\perp$. All of them were close to $\pm 2^{-17}$ as suggested by the above analysis. This gives some circumstantial support to the idea that $B^{48} \approx C$, that PRINTCIPHER has a stable symmetry, and that this is the main contribution to $c_F(\alpha, \beta)$.

On PRINTCIPHER-12, we can derive B_{12} analogously to above. Here the stable symmetry can then be confirmed by deriving the eigenvalues numerically for all possible matrices B_{12} . Also, the convergence can be observed experimentally. Fig. 6(a) shows B_{12}^{100} for a non-weak key, while Fig. 6(b) corresponds to a weak key. The matrices clearly differ both in terms of magnitude and structure. Furthermore convergence is rather fast, B_{12}^{10} is already very close to the expected limit.

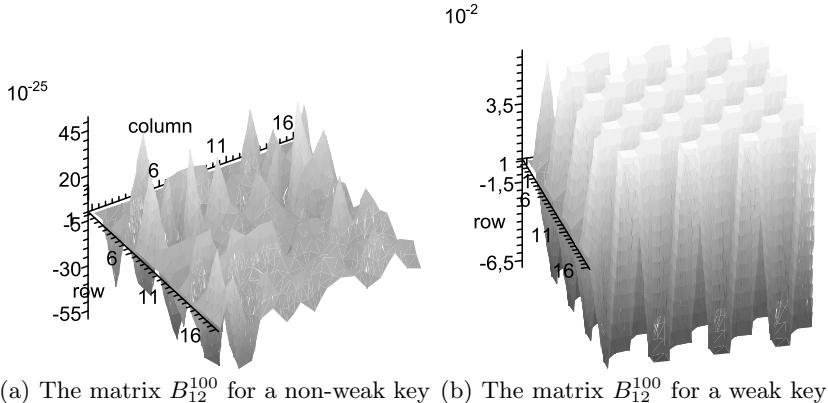


Fig. 6. The matrix B_{12}^{100} for two different keys

6 Conclusion and Future Work

We presented and analyzed three interesting examples of ciphers with a non-expected distribution of correlations. The first example mainly limits the most general statements one can hope to prove. General theorems on key-alternating ciphers have to deal with this strange behavior as well.

The second example considered the influence of the key scheduling on the distribution of correlations. For future work this suggests that highly non-linear key scheduling algorithms might be preferable (cf. also [18]). To see this consider the covariance between two different non-zero trails C_θ and $C_{\theta'}$ for $\theta = (\alpha, \theta_1, \dots, \theta_{r-1}, \beta)$ and $\theta' = (\alpha, \theta'_1, \dots, \theta'_{r-1}, \beta)$ in the case where the key-length equals the block length. Given $E(k) = (E_0(k), \dots, E_r(k))$ we assume furthermore that all E_i are permutations and wlog $E_1(k) = k$. Denoting $\gamma = \theta_1 + \theta'_1$, $\delta = (\theta_2 + \theta'_2, \dots, \theta_{r-1} + \theta'_{r-1})$ and $E'(k) = (E_2(k), \dots, E_{r-1}(k))$, in this setup the covariance is essentially determined by

$$\sum_k (-1)^{\langle \theta + \theta', E(k) \rangle} = \sum_k (-1)^{\langle \delta, E'(k) \rangle + \langle \gamma, k \rangle},$$

which is nothing else than the Fourier coefficient $\widehat{E'}(\gamma, \delta)$. Thus minimizing all covariances corresponds to minimizing the absolute value of $\widehat{E'}(\gamma, \delta)$ which in turn corresponds exactly to maximizing the nonlinearity of E' .

The last example is again related to key-scheduling but more so to symmetries in ciphers. We show a general equivalence of symmetries and linear approximations for weak keys that exist for any number of rounds. This is actually a positive result as it suggests that avoiding these symmetries makes clustering of trails unlikely. Future work is needed to either make this equivalence tighter or find examples of round-independent trail clustering that does not originate from symmetries.

We hope that this work stimulates further research on this fundamental topic.

Acknowledgments. The second author was supported by the Swedish Foundation for Strategic Research (SSF) through its Strategic Center for High Speed Wireless Communication at Lund. The third and fourth authors gratefully acknowledge the support from the Danish National Research Foundation and the National Science Foundation of China (Grant No. 11061130539) for the Danish-Chinese Center for Applications of Algebraic Geometry in Coding Theory and Cryptography.

References

1. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
2. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I.: SPONGENT: A Lightweight Hash Function. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 312–325. Springer, Heidelberg (2011)
3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
4. Bogdanov, A., Wang, M.: Zero correlation linear cryptanalysis with reduced data complexity. In: FSE 2012 (to appear, 2012)
5. Buchanan, M.L., Parlett, B.N.: The uniform convergence of matrix powers. Numerische Mathematik 9(1), 51–54 (1966)
6. Cho, J.Y.: Linear Cryptanalysis of Reduced-Round PRESENT. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
7. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: Nessie proposal: NOEKEON (2000), <http://gro.noekeon.org/Noekeon-spec.pdf>
8. Daemen, J., Rijmen, V.: The design of Rijndael: AES - the Advanced Encryption Standard. Springer, Heidelberg (2002)
9. Daemen, J., Rijmen, V.: Probability distributions of correlation and differentials in block ciphers. IACR Cryptology ePrint Archive, 2005:212 (2005)
10. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional Extension of Matsui’s Algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009)
11. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTCIPHER: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
12. Knudsen, L.R., Robshaw, M.: The Block Cipher Companion. Information security and cryptography. Springer, Heidelberg (2011)
13. Leander, G.: On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 303–322. Springer, Heidelberg (2011)
14. Leander, G., Abdelraheem, M.A., AlKhazaimi, H., Zenner, E.: A Cryptanalysis of PRINTCIPHER: The Invariant Subspace Attack. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 206–221. Springer, Heidelberg (2011)
15. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
16. Murphy, S.: The effectiveness of the linear hull effect. Technical report, RHUL-MA-2009-19 (2009)

17. Nyberg, K.: Linear Approximation of Block Ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
18. Nyberg, K.: Comments on key-scheduling. Personal communication (2012)
19. Ohkuma, K.: Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009)
20. Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 247–260. Springer, Heidelberg (2003)
21. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. *J. Cryptology* 21(1), 131–147 (2008)

Substitution-Permutation Networks, Pseudorandom Functions, and Natural Proofs*

Eric Miles and Emanuele Viola

Northeastern University
`{enmiles,viola}@ccs.neu.edu`

Abstract. This paper takes a new step towards closing the troubling gap between pseudorandom functions (PRF) and their popular, bounded-input-length counterparts. This gap is both quantitative, because these counterparts are more efficient than PRF in various ways, and methodological, because these counterparts usually fit in the substitution-permutation network paradigm (SPN) which has not been used to construct PRF.

We give several candidate PRF \mathcal{F}_i that are inspired by the SPN paradigm. This paradigm involves a “substitution function” (S-box). Our main candidates are:

$\mathcal{F}_1 : \{0,1\}^n \rightarrow \{0,1\}^n$ is an SPN whose S-box is a random function on b bits given as part of the seed. We prove unconditionally that \mathcal{F}_1 resists attacks that run in time $\leq 2^{eb}$. Setting $b = \omega(\lg n)$ we obtain an inefficient PRF, which however seems to be the first such construction using the SPN paradigm.

$\mathcal{F}_2 : \{0,1\}^n \rightarrow \{0,1\}^n$ is an SPN where the S-box is (patched) field inversion, a common choice in practical constructions. \mathcal{F}_2 is computable with Boolean circuits of size $n \cdot \log^{O(1)} n$, and in particular with seed length $n \cdot \log^{O(1)} n$. We prove that this candidate has exponential security $2^{\Omega(n)}$ against linear and differential cryptanalysis.

$\mathcal{F}_3 : \{0,1\}^n \rightarrow \{0,1\}$ is a non-standard variant on the SPN paradigm, where “states” grow in length. \mathcal{F}_3 is computable with size $n^{1+\epsilon}$, for any $\epsilon > 0$, in the restricted circuit class TC^0 of unbounded fan-in majority circuits of constant-depth. We prove that \mathcal{F}_3 is almost 3-wise independent.

$\mathcal{F}_4 : \{0,1\}^n \rightarrow \{0,1\}$ uses an extreme setting of the SPN parameters (one round, one S-box, no diffusion matrix). The S-box is again (patched) field inversion. We prove that this candidate fools all parity tests that look at $\leq 2^{0.9n}$ outputs.

Assuming the security of our candidates, our work also narrows the gap between the “Natural Proofs barrier” [Razborov & Rudich; JCSS ’97] and existing lower bounds, in three models: unbounded-depth circuits, TC^0 circuits, and Turing machines. In particular, the efficiency of the circuits computing \mathcal{F}_3 is related to a result by Allender and Koucký [JACM ’10] who show that a lower bound for such circuits would imply a lower bound for TC^0 .

* Supported by NSF grant CCF-0845003.

1 Introduction

This paper takes a new step towards closing the troubling gap between pseudo-random functions ([17], cf. [16, §3.6]) and their popular, bounded-input-length counterparts. These counterparts are mostly obtained in two ways. One is to use bounded-input-length hash functions such as the SHA-1 compression function, or block ciphers such as the Advanced Encryption Standard (AES) by Daemen and Rijmen [10]. We note that the latter satisfy the additional constraint of computing permutation functions.

This gap is both quantitative and methodological. It is quantitative because all candidate pseudorandom functions (hereafter, PRF) based on complexity-theoretic assumptions (e.g. [17, 21, 35, 20, 43]) have seed length at least quadratic in the input length n , which also implies a quadratic lower bound on the circuit size of such PRF. In contrast, bounded-input-length constructions often have seed length which *equals* the input length. This is for example the case with the 128-bit version of AES.

It is methodological because many modern bounded-input-length hash functions and block ciphers are constructed using the *substitution-permutation network* (SPN) paradigm. This is for example the case with two of the finalists for the ongoing SHA-3 cryptographic hash function competition, namely Grøstl [13] and JH [45], and also the AES block cipher. An SPN is computed over a number of rounds, where each round “confuses” the input by dividing it into bundles and applying a substitution function (S-box) to each bundle, and then “diffuses” the bundles by applying a matrix with certain “branching” properties (cf. [42]). No piece of this structure appears to have been used to construct PRF. In fact, until the present paper no asymptotic analysis of the SPN structure was given. This is in stark contrast with the seminal work of Luby and Rackoff [31] that gave such an analysis for the so-called *Feistel network* structure (which in particular was the basis for the block cipher DES, the predecessor to AES). Moreover the SPN structure is tailored to resist two general attacks on block ciphers which appear to be ignored in the PRF literature, namely linear and differential cryptanalysis.

In this paper we give several candidate PRF that are inspired by the SPN structure, though unlike popular constructions we do not require that an SPN computes a permutation function. Each of the many hash functions and block ciphers based on the SPN structure (e.g. those mentioned above) suggests different choices for the parameters, S-boxes, and diffusion matrices. As a first step we choose to follow the design considerations behind the AES block cipher, and particularly its S-box. We do this for two reasons. First, it is a well-documented, widely-used block cipher that has been around for over a decade. Second, the algebraic structure of its S-box lends itself to an asymptotic generalization; we exploit this fact in some of our results. We hope that future work will systematically address other available bounded-input-length constructions.

Some of our candidates have better parameters than previous candidates, where by parameters we refer to the seed length and the resources required to compute each function in various computational models:

1. We first consider an SPN with a random S-box (specified as part of the seed). We prove unconditionally that this resists attacks that run in time less than the seed length. For example we can set the seed length to n^c and withstand attacks running in time $n^{c'}$ for sufficiently large c and $c' = \Theta(c)$. (Note that being a PRF means that the seed length is n^c and that the function withstands all attacks running in time $n^{c'}$ for *any* c' .)

This result is analogous to that of Luby and Rackoff, who analyzed the Feistel network structure when a certain component is instantiated with a random function, and indeed we prove the same level of security (exponential in the input size of the random function). The techniques used are similar to those in the work by Naor and Reingold [34] that followed Luby and Rackoff's. To our knowledge this is the first construction of a (provably secure, inefficient) PRF using the SPN structure.

2. Using the AES S-box and a strengthened version of the AES diffusion matrix, we give a candidate computable with Boolean circuits of size $n \cdot \log^{O(1)} n$, and in particular with seed length $O(n \log^2 n)$. We prove that this candidate has exponential security $2^{\Omega(n)}$ against linear and differential cryptanalysis by extending a result due to Kang et al. [26].
3. Again using the AES S-box and a different diffusion matrix, we give a candidate computable with size $n^{1+\epsilon}$, for any $\epsilon > 0$, in the restricted circuit class TC^0 of unbounded fan-in majority circuits of constant-depth. The diffusion matrix used here blows up the state to size $O(n)$, and we output a single bit by taking the inner product of this state with a random string. We prove that this candidate is almost 3-wise independent.
4. We give another single-bit output candidate which uses an extreme setting of the SPN parameters (one round, one S-box, no diffusion matrix). This can be viewed as a slightly modified version of the Even-Mansour cipher [11] that uses the AES S-box in place of a random permutation. We prove that this candidate fools all parity tests that look at $\leq 2^{0.9n}$ outputs.
5. Our final candidate is a straightforward generalization of AES, and may be folklore. We show that it is computable by size $O(n^2)$, depth $O(n)$ Boolean circuits, and we further show that for each fixed seed k it is computable in time $O(n^2)$ by a single-tape Turing machine with $O(n^2)$ states. We do not have any proof of security, but the (heuristic) arguments underlying AES's security also apply to this candidate.

For context, we mention that Hoory, Magen, Myers and Rackoff [24] and Brodsky and Hoory [8], building on work by Gowers [19], study the random composition of a family of permutations. The SPN structure can be seen as falling into this framework, by taking each round as an individual permutation chosen randomly by the key. However, the permutations constructed in these works do not have the form of an SPN round, and furthermore the circuit complexity of the composed permutations is not of interest to them (their constructions have size and depth $\Omega(n^3)$).

Natural Proofs. The landscape of circuit lower bounds remains bleak, despite exciting recent results [44]. Researchers however have been successful in explaining this lack of progress by pointing out several “barriers,” i.e. establishing that certain proof techniques will not give new lower bounds [4,39,1].

Of particular interest to us is the Natural Proofs work by Razborov and Rudich [39]. They make the following two observations. First, most lower-bound proofs that a certain function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ cannot be computed by circuits C (e.g., $C =$ circuits of size n^2) entail an algorithm that runs in time polynomial in $N := 2^n$ and can distinguish truth-tables of n -bit functions $g \in C$ from truth-tables of random functions (i.e., a random string of length N). (For example, the algorithm corresponding to the restriction-based proof that Parity is not in AC^0 , given $f : \{0, 1\}^n \rightarrow \{0, 1\}$, checks if there is one of the $2^{O(n)} = N^{O(1)}$ restrictions of the n variables that makes f constant.) Informally, any proof that entails such an algorithm is called “natural.”

The second observation is that, under standard hardness assumptions, no algorithm such as the above one exists when C is a sufficiently rich class. This follows from the existence of PRF with security $2^{s^{\Omega(1)}}$ where s is the seed length (e.g. [17,21,35,20,43]) and by setting $s := n^c$ for a sufficiently large c .

The combination of the two observations is that no natural proof exists against circuits of size n^c , for some constant $c \geq 2$.

Moreover, the PRF construction [35] by Naor and Reingold is implementable in TC^0 , pushing the above second observation “closer” to the frontier of known circuit lower bounds. For completeness we also mention that this PRF achieves seed length $s = O(n^2)$ and is a candidate to having hardness $2^{\Omega(n)}$ under elliptic-curve conjectures.

The Gap between Lower Bounds and PRF. However, the natural proofs barrier still has a significant gap with known lower bounds, due to the lack of sufficiently strong PRF. For example, there is no explanation as to why one cannot prove superlinear-size circuit lower bounds. For this one would need a PRF $f_k : \{0, 1\}^n \rightarrow \{0, 1\}$ that is computable by linear-size circuits (hence in particular with $|k| = O(n)$) and with exponential hardness 2^n . (So that, given n , if one had a distinguisher running in time $2^{O(n)}$, one could pick a PRF on inputs of length bn for a large enough constant b , to obtain a contradiction.)

A recent work by Allender and Koucký [2] brings to the forefront another setting where the Natural Proofs barrier does not apply: proving lower bounds on TC^0 circuits of size $n^{1+\epsilon}$ and depth d , for any $\epsilon > 0$ and large enough $d = d(\epsilon)$. (As mentioned above, the Naor-Reingold PRF requires larger size.) This setting is especially interesting because [2] shows that such a lower bound for certain functions implies a “full-fledged” lower bound for TC^0 circuits of polynomial-size. Moreover even if the first lower bound were natural, the latter would not be, thus circumventing the Naor-Reingold PRF.

Another long-standing problem is to exhibit a candidate PRF in ACC^0 .

Of course, circuit models such as the above ones are only some of the models in which the gap between candidate PRF and lower bounds is disturbing.

Other such models include various types of Turing machines, and small-space branching programs. For example, there is no explanation as to why the lower bounds for single-tape Turing machines stop at quadratic time, cf. [30, §12.2].

Assuming the (exponential) security of some of our candidates, our work narrows this gap in three ways. First, Candidate 2 is computable by quasilinear-size Boolean circuits. Second, Candidate 3 is computable by TC^0 circuits of size $n^{1+\epsilon}$ and depth $d = d(\epsilon)$ for any $\epsilon > 0$. Third, for each fixed seed k Candidate 5 is computable in time $O(n^2)$ by a single-tape Turing machine with $O(n^2)$ states (note that the fixed-seed setting suffices for the Natural Proofs connection).

1.1 Background on SPNs

To formally define our candidates, we begin by reviewing the SPN structure (refer to Figure 1). The notation introduced in this section will be used throughout the paper.

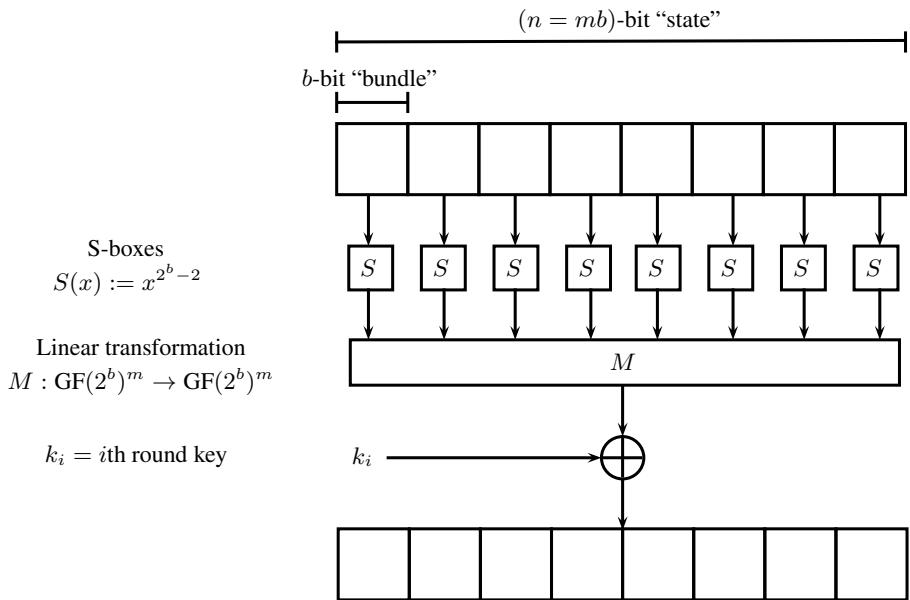


Fig. 1. One round of an SPN

An SPN $C_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is indexed by a key $k = (k_0, \dots, k_r) \in (\{0, 1\}^n)^{r+1}$, and is specified by the following three parameters and two functions:

- $r \in \mathbb{N}$, the number of *rounds*
- $b \in \mathbb{N}$, the *S-box input size*

- $m \in \mathbb{N}$, the *number of S-box invocations per round*
- $S : \text{GF}(2^b) \rightarrow \text{GF}(2^b)$, the *S-box*
- $M : (\text{GF}(2^b))^m \rightarrow (\text{GF}(2^b))^m$, the *linear transformation*.

The input/output size of C_k is given by $n := mb$. Throughout this paper, we assume a fixed canonical mapping between $\{0, 1\}^b$ and $\text{GF}(2^b)$.

C_k is computed over r rounds. The i th round ($1 \leq i \leq r$) is computed over three steps: (1) m parallel applications of S ; (2) application of M to the entire state; (3) XOR of the entire state with the round key k_i . Note that each round is identical except for step (3).¹

On input x , $C_k(x)$ gives $x \oplus k_0$ as input to the first round; the output of round i becomes the input to round $i + 1$ (for $1 \leq i < r$), and $C_k(x)$'s output is the output of the r th round.

Security against Linear and Differential Cryptanalysis. We now briefly review how the security of an SPN is evaluated against two general attacks on block ciphers: linear and differential cryptanalysis. (See the full version for a more extensive discussion.) Resistance to these attacks is typically seen as the main security feature of SPNs. Note that we consider here the basic versions of these attacks, and we leave to future work understanding the resistance of our candidates to more sophisticated attacks (such as those considered by Knudsen [28]).

For both linear and differential cryptanalysis, a crucial property in the security proof is that the linear transformation M has maximal *branch number*, defined as follows.

Definition 1. Let $M : \mathbb{F}^m \rightarrow \mathbb{F}^m$ be a linear transformation acting on vectors over a field \mathbb{F} . The *branch number* of M is

$$\text{Br}(M) = \min_{\alpha \neq 0^m} (w(\alpha) + w(M(\alpha))) \leq m + 1$$

where $w(\cdot)$ denotes the number of non-zero elements.

Linear cryptanalysis [32] exploits the existence of linear correlations to attack a block cipher C_k . For a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and input/output parities $\Gamma_x, \Gamma_y \in \{0, 1\}^n$, define the *correlation* of f with respect to Γ_x and Γ_y as

$$\text{Cor}_{\Gamma_x, \Gamma_y}(f) := 2 \cdot \Pr_x [\langle \Gamma_x, x \rangle = \langle \Gamma_y, f(x) \rangle] - 1.$$

For a block cipher C_k , the parameter of interest for linear cryptanalysis is

$$p_{\text{LC}}(C_k) := \max_{\Gamma_x, \Gamma_y \neq 0} \left(\mathbb{E}_k \left[\text{Cor}_{\Gamma_x, \Gamma_y}(C_k)^2 \right] \right).$$

Specifically, the attack requires an expected number of plaintext/ciphertext pairs proportional to $1/p_{\text{LC}}(C_k)$.

¹ SPNs are sometimes defined more generally, e.g. by allowing the S-box to vary across rounds or by allowing a more complex interaction with k than XOR.

Differential cryptanalysis [6] attacks a block cipher C_k by exploiting the relationship between the XOR difference of two inputs to C_k and the XOR difference of the corresponding outputs. For a function $f_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ parameterized by a key k , and input/output differences $\Delta_x, \Delta_y \in \{0, 1\}^n$, define the *difference propagation probability* (DPP) of f_k with respect to Δ_x and Δ_y as

$$\text{DPP}_{\Delta_x, \Delta_y}(f_k) := \Pr_{x, k} [f_k(x) \oplus f_k(x \oplus \Delta_x) = \Delta_y].$$

(If f is not parameterized by a key, k is ignored in this definition). For a block cipher C_k , the parameter of interest for differential cryptanalysis is

$$p_{\text{DC}}(C_k) := \max_{\Delta_x, \Delta_y \neq 0} (\text{DPP}_{\Delta_x, \Delta_y}(C_k)).$$

Specifically, the attack requires an expected number of plaintext/ciphertext pairs proportional to $1/p_{\text{DC}}(C_k)$.

The following theorem, due to Kang et al. [26], gives a bound on p_{LC} and p_{DC} for 2-round SPNs with maximal branch number.

Theorem 1. ([26], Thms. 5 & 6) *Let $C_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be an SPN with $r = 2$ rounds and S-box S . Let $q := \max_{\Gamma_x, \Gamma_y \neq 0} (\text{Cor}_{\Gamma_x, \Gamma_y}(S)^2)$ denote the maximum squared correlation of S , and let $p := \max_{\Delta_x, \Delta_y \neq 0} (\text{DPP}_{\Delta_x, \Delta_y}(S))$ denote the maximum DPP of S . If $\text{Br}(M) = m + 1$, then $p_{\text{LC}}(C_k) \leq q^m$ and $p_{\text{DC}}(C_k) \leq p^m$.*

For typical S-boxes, such as the one used in AES, one can have $q = p = 2^{-b+2}$, and so the theorem guarantees security exponential in $n = mb$. (For completeness we note that one cannot directly apply the above theorem to AES because it is a more complicated SPN.)

We extend this result to $r > 2$ rounds in the following theorem.

Theorem 2. *Let $C_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be an SPN with $r = 2\ell$ rounds for some $\ell \geq 1$ and S-box S . Let $q := \max_{\Gamma_x, \Gamma_y \neq 0} (\text{Cor}_{\Gamma_x, \Gamma_y}(S)^2)$ denote the maximum squared correlation of S , and let $p := \max_{\Delta_x, \Delta_y \neq 0} (\text{DPP}_{\Delta_x, \Delta_y}(S))$ denote the maximum DPP of S . If $\text{Br}(M) = m + 1$,*

$$1. \quad p_{\text{LC}}(C_k) \leq q^{\ell m} \cdot 2^{(\ell-1)n}. \quad 2. \quad p_{\text{DC}}(C_k) \leq p^{\ell m} \cdot 2^{(\ell-1)n}.$$

Intuitively, the S-box provides security q (resp. p) against linear (resp. differential) cryptanalysis, and this security multiplies across “active” S-boxes (instances of S that are evaluated with a non-zero input). The branch number $\text{Br}(M)$ guarantees that there exist $\geq m + 1$ such active S-boxes in any pair of consecutive rounds, hence the term $q^{\ell m} = q^{(r/2)m}$. We note that the factor $2^{(\ell-1)n}$ seems to be an artifact of our extension of [26], and it is open to get a tighter bound on p_{LC} and p_{DC} for $r > 2$ rounds ([26] only consider $r = 2$). Such an extension has been considered before, for example by Keliher et al. [27] and Cho et al. [9], but their results only apply in the fixed-parameter setting because they require extensive computer calculation. We are not aware of any other “closed form” bound for $r > 2$.

Security against Degree-Exploiting Attacks. While resistance to linear and differential cryptanalysis is the main security feature of the SPN structure (and indeed, “the most important criterion in the design” of AES [10, p. 81]), considerations are usually also taken to prevent attacks that would exploit algebraic structure in the cipher. In our candidates 2-5, we adopt essentially the same S-box that is used in AES.² This S-box is defined by $S(x) := x^{2^b - 2}$ and was chosen to allow the computation to have high degree when considered as a multivariate polynomial over GF(2). Specifically, the use of $x \mapsto x^{2^b - 2}$ results in each of S ’s output bits having (near-maximum) degree $b - 1$. Using instead $x \mapsto x^3$ would not diminish resistance to linear and differential cryptanalysis, but it would result in degree (only) 2 [37,36,29].

We need the degree of each output bit of our candidates (as a multivariate GF(2)-polynomial) to be $\geq \epsilon n$, for some constant ϵ , to resist attacks that exploit the degree of this polynomial. For completeness we present such an attack, showing that a PRF that has degree $o(n)$ cannot have hardness 2^n .

Theorem 3. *Let $F = \{f_k : \{0,1\}^n \rightarrow \{0,1\}\}_k$ be any set of functions such that, for each key k , the polynomial representation of f_k over GF(2) has degree $o(n)$. Then there is an adversary that runs in time $\leq 2^{\Omega(n)}$ and distinguishes a random $f_k \in F$ from a random function with advantage $\geq 1 - 2^{-2^{\Omega(n)}}$.*

The only non-linear operation in the entire cipher is the S-box, which for Candidates 2-5 has degree $b - 1$, and thus the maximum possible degree of each output bit for these candidates is $(b - 1)^r$. Hence we make sure that

$$b^r \geq n$$

in each of our candidates. (The distinction between $(b - 1)^r \geq \epsilon n$ and $b^r \geq n$ is unimportant, as in our candidates we can always increase r by a constant factor, except in Candidate 4 where we have $b = n$ and $r = 1$.) We do not know if $b^r \geq n$ is sufficient to guarantee degree $\Omega(n)$, and it is an interesting research direction to understand what restrictions (if any) on the SPN parameters ensure that the function has high degree.

Finally, although a block cipher’s security is often measured against *key-recovery* attacks, we share many researchers’ viewpoint that *distinguishing* attacks are the correct model. We also note that there is often an intimate connection between the two types, as many key recovery techniques, including linear and differential cryptanalysis, construct a distinguishing algorithm which is then used to select the correct round keys from a set of potential keys.

² Besides the obvious difference that in AES the value b is fixed to be 8, we omit the $\text{GF}(2)^b$ -affine function that is included in the AES S-box. Adding such a function would not affect the (asymptotic) circuit size of our candidates, and removing it does not affect resistance to linear/differential cryptanalysis. To our knowledge there are no known attacks against the AES variant that uses this “reduced” S-box.

2 Our Candidates

We now describe our candidates. Candidates 1, 2, and 5 output n bits, while Candidates 3 and 4 output 1 bit. We use \mathcal{F}_i to refer to the function computing Candidate i . In each candidate, the $(r+1)$ n -bit round keys are chosen independently and uniformly at random. (Popular constructions typically employ a so-called “key schedule” that generates the round keys from a key of size $\ll n(r+1)$.)

Candidate 1. Our first candidate \mathcal{F}_1 is an r -round SPN with an S-box that is chosen uniformly at random (i.e. specified as part of \mathcal{F}_1 ’s key) from the set of all functions mapping $\text{GF}(2^b)$ to itself. (Analyzing this candidate when S is a random *permutation* is a natural research direction which we do not address here.) The only restriction we make on \mathcal{F}_1 ’s linear transformation M is that it is invertible and has all entries $\neq 0$; we observe that this holds for any M with maximal branch-number. We show that any adversary A has small advantage in distinguishing \mathcal{F}_1 from a random function F .

Theorem 4. *If A makes at most q total queries to its oracle, then*

$$\left| \Pr_F[A^F = 1] - \Pr_{\mathcal{F}_1}[A^{\mathcal{F}_1} = 1] \right| < O(r^2 m^3 q^3) \cdot 2^{-b}.$$

The bound achieved here is similar to that of Luby and Rackoff [31] in the sense that it is exponentially small in the size of the random function, with a polynomial loss in the number of queries. (The fact that security degrades with the number of rounds, contrary to what one might expect, seems to be an artifact of the proof.) The proof of this theorem is very similar to that of [34, Thm. 3.2], and proceeds by bounding the collision probability between any two inputs to S in the final round. However we face an additional hurdle, namely that the inputs to the random function S in the final round depend on outputs of S in previous rounds.

By setting $b = \omega(\log n)$ and $r = \log n$, we get an inefficient PRF (with security $n^{\omega(1)}$). We also note that by setting $b = c \log n$ for some sufficiently large constant c , \mathcal{F}_1 is computable in time $n^{O(c)}$ and has security $n^{c'}$ for some $c' = \Omega(c)$.

Finally, note that Theorem 4 implies corresponding bounds on $p_{\text{LC}}(\mathcal{F}_1)$ and $p_{\text{DC}}(\mathcal{F}_1)$.

Candidate 2. In this candidate we set $b = \Theta(\log n)$, and we use the AES S-box on b bits (recall that it maps $x \mapsto x^{2^b-2}$). We use a linear transformation M with maximal branch number, and M is constructed from an error-correcting code in a similar manner to the linear transformation in AES. (AES’s linear transformation does not have maximal branch number however, a choice that was made to reduce computation time.) We set the number of rounds $r = \Theta(\log n)$ (observe that $b^r \geq n$).

We prove that Candidate 2 is computable by Boolean circuits of quasilinear-size $\tilde{O}(n) := n \cdot \log^{O(1)} n$. To show this, note that since r is logarithmic it is enough to show how to compute each round with these resources. Moreover, since b is logarithmic, computing the S-boxes comes at little cost.

Our main technical contribution in this candidate is to show how to efficiently compute the linear transformation M ; specifically, we show that it can be computed with size $\tilde{O}(n)$, for a total circuit size of $r \cdot (b^{O(1)} + \tilde{O}(n)) = \tilde{O}(n)$. A common method for constructing maximal-branch-number linear transformations is to use the generator matrix G of an $m \rightarrow 2m$ maximum distance separable (MDS) code; specifically, if $G^T = [I \mid A]$, then $M := A$ has maximal branch number. Our method for computing M efficiently has two parts. First, we use a result by Roth and Seroussi [41] that if G generates a Reed-Solomon code (which is well-known to be MDS), then M forms a $t \times t$ *Cauchy matrix* (a type of matrix specified by $O(t)$ elements). We then use a result by Gerasoulis [15] to compute the product of a vector (consisting of bundles of the state) and a Cauchy matrix in quasilinear time; this requires a simple adaptation of the algorithm in [15] to fields of characteristic 2.

By combining Theorem 2 with a theorem of Nyberg [36], we show that this candidate has exponential security against linear and differential cryptanalysis.

Theorem 5. 1. $p_{LC}(\mathcal{F}_2) \leq 2^{-\Omega(n)}$. 2. $p_{DC}(\mathcal{F}_2) \leq 2^{-\Omega(n)}$.

We do not know how to get a candidate computable by circuits of size $O(n)$.

Candidate 3. In the previous candidate, the components S and M remain essentially unchanged from AES. In Candidate 3, we also keep S the same (aside from the increase in input/output size), but we modify the linear transformation M .

Our observation is that the rationale for using a linear transformation with maximal branch number is just that it allows one to lower bound the number \mathcal{A} of so-called “active” S-boxes, which can be defined as follows. Let C be an SPN which uses the identity permutation for S and which has $k_i := 0$ for $0 \leq i \leq r$. Let $w_b : (\{0, 1\}^b)^m \rightarrow \mathbb{N}$ be the function that counts the number of non-zero b -bit bundles in its input. Then,

$$\mathcal{A} := \min_{0^n \neq x \in \{0, 1\}^n} \sum_{i=1}^r w_b(\text{state of } C(x) \text{ at the beginning of round } i).$$

This number \mathcal{A} is crucial in evaluating the security of SPNs against linear and differential cryptanalysis (cf. [26, 10]). With a simple modification to M , we get that a constant fraction of the S-boxes in each round are active. Specifically we use the full generator matrix of an error correcting code with minimum distance $\Omega(n)$, which comes at the expense of expanding the state from n bits to $O(n)$ bits at each round. To counteract the fact that such codes may have some output positions fixed to constant values (leading to a simple distinguishing attack), the computation of Candidate 3 concludes by taking the inner product of the state

with a uniform $O(n)$ -bit vector that is given as part of the seed. Candidate 3 therefore outputs a single bit.

We take $b = n^\epsilon$ and $r = O(1/\epsilon)$ for arbitrarily small $\epsilon > 0$, and so each round is computable in size

$$\frac{n}{b} \cdot \text{poly}(b) = n^{1+O(\epsilon)},$$

and the whole circuit also in size $n^{1+O(\epsilon)}$.

We further show that Candidate 3 is computable even by TC^0 circuits of size $n^{1+O(\epsilon)}$ for any $\epsilon > 0$ (with depth depending on ϵ), cf. § “The gap between lower bounds and PRF” above. The main technical difficulty in implementing this candidate with the required resources is that the S-box requires computing *inversion* in a field of size 2^b (recall $b = n^{\Omega(1)}$). To implement this in TC^0 we note (cf. [22]) that inverting the field element $\alpha(x)$ can be accomplished as:

$$\alpha(x)^{2^b - 2} = \alpha(x)^{\sum_{i=1}^{b-1} 2^i} = \prod_{i=1}^{b-1} \alpha(x)^{2^i} = \prod_{i=1}^{b-1} \alpha\left(x^{2^i}\right)$$

where the last equality follows from the fact that we are working in characteristic 2. By hard-wiring the $\leq b$ powers $x, x^2, \dots, x^{2^{b-1}}$ of x in the circuit, and using the fact that the iterated product of $\text{poly}(n)$ field elements is computable by $\text{poly}(n)$ -size TC^0 circuits (see e.g. [23, Corollary 6.5] and cf. [22]), we obtain a TC^0 circuit.

Because Candidate 3 deviates somewhat from the SPN structure, we cannot use Theorem 1, and indeed it is not clear how to define differential cryptanalysis for functions which output only one bit. However, we are able to leverage a technique from differential cryptanalysis to prove that Candidate 3 is almost 3-wise independent. We were unable to determine if this candidate is 4-wise independent.

Definition 2. A function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ parameterized by a key k is (d, ϵ) -wise independent if for any distinct $x_1, \dots, x_d \in \{0, 1\}^n$, the distribution $(f(x_1), \dots, f(x_d))$ induced by a uniform choice of k is ϵ -close to U_d in statistical distance.

Theorem 6. \mathcal{F}_3 is $(3, 2^{-\Omega(n)})$ -wise independent.

Finally, we mention that implicit in an assumption that Candidate 3 is indeed hard is the assumption that field inversion cannot be computed by unbounded fan-in constant depth circuits with parity gates $\text{AC}^0[\oplus]$. For otherwise, it can be shown that the whole candidate would be in that class, in contradiction with an algorithm in [39, §3.2.1] which distinguishes truth tables of $\text{AC}^0[\oplus]$ functions from random ones in quasipolynomial time. (M can be seen to be a linear operation over $\text{GF}(2)$, hence it can be computed easily with parity gates.) The question of whether field inversion is in $\text{AC}^0[\oplus]$ was raised by Healy and Viola in [22]. Their work, and later Kopparty’s [29], do show that several functions related to field inversion are not in $\text{AC}^0[\oplus]$.

Candidate 4. In this candidate, we use the extreme setting of parameters $b = n$ and $r = 1$. In other words, Candidate 4 consists of one round, and this round contains only a single S-box (and in particular no linear transformation). This construction can be seen as a concrete instantiation of the Even-Mansour block cipher [11], using the AES S-box in place of the random permutation oracle. While this setting does indeed preserve resistance to linear and differential cryptanalysis, we exhibit a simple attack, inspired by Jakobsen and Knudsen [25], in which we exploit the algebraic structure to recover the key with just 4 queries.

We then put forth a related candidate \mathcal{F}'_4 where we only output the Goldreich-Levin bit [18]: $\mathcal{F}'_4(x) := \langle (x + k_0)^{2^b - 2}, k_1 \rangle$. We prove that this candidate is a d -wise small-bias generator with error $d/2^n$ (cf. [33,3]), i.e. that it fools all parity tests that look at $\leq 2^{0.9n}$ outputs.

Theorem 7. *For any choice of $d \leq 2^n$, \mathcal{F}'_4 is a d -wise small-bias generator with error $d/2^n$. That is, for any distinct $a_1, \dots, a_d \in \{0, 1\}^n$:*

$$\left| \Pr_{k_0, k_1} \left[\sum_{i=1}^d \mathcal{F}'_4(a_i) = 0 \right] - \frac{1}{2} \right| < \frac{d}{2^n}.$$

Using Braverman's result [7] (cf. [5,40]) we obtain that this candidate also fools small-depth AC⁰ circuits of any size $w = 2^{n^{o(1)}}$ (that look at only w fixed output bits of the candidate).

Using the same ideas for Candidate 3, this candidate is also computable by poly-size TC⁰ circuits. For unbounded-depth circuits, a more refined size bound $\tilde{O}(n^2)$ follows from the exponentiation algorithm in [12].

Candidate 5. Our final candidate is a straightforward generalization of AES, and may be folklore. We set $b = 8$ as in AES and we again use AES's S-box. We also use the same linear transformation as in AES (which is slightly different from that of Candidate 2, cf. [10]), except for the necessary increase in the input/output size. We set the number of rounds $r = n$, and thus the size of the seed is $|k| = n(n+1)$.

Candidate 5 is computable by size $O(n^2)$, depth $O(n)$ Boolean circuits. For each fixed seed k , Candidate 5 is also computable in time $O(n^2)$ by a single-tape Turing machine with $O(n^2)$ states.

We do not know how to get a candidate computable in time $O(n)$ on a 2-tape Turing machine.

Due to space constraints, the technical details of most of our candidates and full proofs of all theorems are deferred to the full version of this paper. However, in the following subsection we explain Candidate 1's proof of security.

2.1 Security of Candidate 1

Recall that \mathcal{F}_1 is an SPN in which the S-box $S : \text{GF}(2^b) \rightarrow \text{GF}(2^b)$ is chosen uniformly at random and the linear transformation M is invertible and has

only non-zero entries. To tie the latter restriction to practical constructions, we observe that any M with maximal branch number suffices for this construction.

Claim. Let $M \in (\text{GF}(2^b))^{m \times m}$ be any matrix with maximal branch number $m + 1$. Then, all entries of M are non-zero and M is invertible.

Proof. Assume for contradiction that $M_{i,j} = 0$ for some $i, j \leq m$. Let $x \in (\text{GF}(2^b))^m$ be the vector such that $x_j = 1$ and $x_{j'} = 0$ for $j' \neq j$. Then $(Mx)_i = 0$, and so $\text{Br}(M) \leq w(x) + w(Mx) \leq m$.

To see that M is invertible, simply note that if $Mx = My$ for $x \neq y$, then $M(x+y) = 0^m$. Since $x+y \neq 0^m$, we would again have $\text{Br}(M) \leq m$. \square

For the remainder of this section, fix any invertible $M \in (\text{GF}(2^b))^{m \times m}$ such that all entries are non-zero. For any function $S : \text{GF}(2^b) \rightarrow \text{GF}(2^b)$ and any set of round keys $\mathbf{k} := (k_0, \dots, k_{r-1}) \in (\{0,1\}^n)^r$, let $\mathcal{F}_1 = \mathcal{F}_1(S, \mathbf{k})$ be the r -round SPN on $n := mb$ bits defined by these components, where the final round consists only of S-boxes (i.e. the final round omits the linear transformation and the key addition).

We make the standard assumption that the adversary A is deterministic, computationally unbounded, and never queries an oracle twice with the same input.

Proof Overview. The proof of Theorem 4 proceeds in two stages. In the first stage, we consider any set of distinct queries x_1, \dots, x_q , and we show that there is a low-probability event BAD over the choice of (S, \mathbf{k}) such that, conditioned on $\neg \text{BAD}$, $\{\mathcal{F}_1(x_i)\}_{i \leq q}$ is uniformly distributed. Essentially, BAD occurs when any of the x_i induce colliding queries to some pair of S-boxes in the final round. When considering distinct instances of the S-box in the final round, even under the same query to \mathcal{F}_1 , this event has low probability simply due to the fact that each b -bit block of k_{r-1} is uniform and independent. However when considering the *same* final round S-box (necessarily under two distinct queries to \mathcal{F}_1), bounding the collision probability is more involved and relies on the properties of M stated above.

In the second stage of the proof, we consider the distribution over transcripts of A 's interaction with its oracle. We show that the probability mass assigned to transcripts for which A outputs 1 differs by at most $\max_{\{x_1, \dots, x_q\}}(\Pr[\text{BAD}])$ (which is negligible by the first stage). To show this we employ a probability argument that has been used in a number of other works, e.g. [34,38,14].

The first stage actually shows that \mathcal{F}_1 is almost q -wise independent, or alternatively that it is pseudorandom against adversaries that make $\leq q$ non-adaptive queries. The technique used in the second stage is a rather generic way of extending the proof to adaptive queries; however we note that it crucially relies on the existence of the event BAD , and indeed it is not the case that any almost q -wise independent function is pseudorandom against adversaries making q adaptive queries.³ A different method (that does not give a useful bound in our setting)

³ This can be seen for example by considering the distribution over functions $f : [N] \rightarrow [N]$ in which each output is selected uniformly and independently with the restriction that $f(f(0)) := 0$. This is almost pairwise-independent, but trivially distinguishable with two adaptive queries.

for obtaining adaptive security from non-adaptive security is given by Hoory et al. [24, Prop. 3].

Stage 1. Fix distinct $x_1, \dots, x_q \in \{0, 1\}^n$. We view (S, \mathbf{k}) being chosen as follows:

1. Uniformly choose k_0, \dots, k_{r-3} .
2. Run the computation of rounds $1, \dots, r-2$ of $\mathcal{F}_1(x_i)$ for all $i \leq q$. Each time the S-box is evaluated on a previously-unseen input, choose the output uniformly at random. Let $H \subseteq GF(2^b)$ be the set of at most $qm(r-2)$ S-inputs whose output is determined after this step.
3. Uniformly choose k_{r-2} .
4. Uniformly choose the output of S for each round- $(r-1)$ S-box whose output is not already determined.
5. Uniformly choose k_{r-1} .
6. Uniformly choose the output of S on all remaining (round r) S-box inputs.

It is clear that, for any x_1, \dots, x_q , this distribution is uniform. Our analysis uses the state of the SPN's computation immediately before the final two rounds, and we denote these states for query x_i as $z_i^{(r-1)}$ and $z_i^{(r)}$.

We now define the event BAD . To reduce notation, we use the following definition.

Definition 3. Let $x, y \in (GF(2^b))^m$, and denote $x = x^{(1)} \dots x^{(m)}$ and $y = y^{(1)} \dots y^{(m)}$. Then, we say that x and y collide if $\exists \ell, \ell' : x^{(\ell)} = y^{(\ell')}$. Further, for any $T \subseteq GF(2^b)$, we say that x and T collide if $\exists \ell \leq m, t \in T : x^{(\ell)} = t$. Finally, we say that x self-collides if $\exists \ell \neq \ell' : x^{(\ell)} = x^{(\ell')}$.

Now, let $BAD = BAD(x_1, \dots, x_q)$ be the set of all (S, \mathbf{k}) such that at least one of the following holds:

- (a) $\exists h, h' \in H : S(h) = S(h')$.
- (b) $\exists i < q : z_i^{(r)} \text{ and } H \text{ collide.}$
- (c) $\exists i, i' \leq q : z_i^{(r)} \text{ and } (z_i^{(r-1)} + k_{r-2}) \text{ collide.}$
- (d) $\exists i \leq q : z_i^{(r)} \text{ self-collides.}$
- (e) $\exists i \neq i' \leq q : z_i^{(r)} \text{ and } z_{i'}^{(r)} \text{ collide.}$

It is crucial for us that determining whether BAD holds can be checked after step 5 in choosing (S, \mathbf{k}) . The following two lemmas show that BAD occurs with low probability, and that the query answers are uniformly distributed when conditioned on $\neg BAD$.

Lemma 1. $\Pr_{S, \mathbf{k}}[BAD] < O(r^2 m^3 q^3) \cdot 2^{-b}$.

Lemma 2. For any distinct x_1, \dots, x_q and any y_1, \dots, y_q :

$$\Pr_{S, \mathbf{k}} [\forall i \leq q : \mathcal{F}_1(x_i) = y_i \mid \neg BAD] = 2^{-qmb}.$$

Stage 2. The proof of Theorem 4 concludes by using the two preceding lemmas in a probability argument similar to [34, Thm. 3.2]. To do this, we consider the distribution over *transcripts* of A 's interaction with its oracles. A transcript is a sequence $\sigma = [(x_1, y_1), \dots, (x_q, y_q)]$ that contains the query/answer pairs arising from A 's interaction with its oracle. We use T_F to denote the transcript of A^F , and we use $A(\sigma)$ to denote A 's output after seeing transcript σ . (So note for instance that $\Pr_F[A^F = 1]$ and $\Pr_F[A(T_F) = 1]$ are semantically equivalent.)

Because A is deterministic, there is a deterministic function Q_A that determines its next query from the partial transcript so far. For a transcript σ , denote its prefixes by $\sigma_i := [(x_1, y_1), \dots, (x_i, y_i)]$. We say a transcript σ is *possible* for A if for all $i < q$: $Q_A(\sigma_i) = x_{i+1}$. Clearly for any *impossible* transcript σ , $\Pr[T_F = \sigma] = 0$ regardless of the distribution from which F is chosen. Also note that the assumption that A never makes the same query twice implies that in any possible transcript, $x_i \neq x_j$ for all $i \neq j$.

Proof (of Theorem 4). Let Γ be the set of possible transcripts such that $A(\sigma) = 1 \Leftrightarrow \sigma \in \Gamma$. Then,

$$\begin{aligned} & \left| \Pr_F[A^F = 1] - \Pr_{S,\mathbf{k}}[A^{\mathcal{F}_1} = 1] \right| \\ &= \left| \sum_{\sigma \in \Gamma} \left(\Pr_F[T_F = \sigma] - \Pr_{S,\mathbf{k}}[T_{\mathcal{F}_1} = \sigma] \right) \right| \\ &\leq \left| \sum_{\sigma \in \Gamma} \Pr_{S,\mathbf{k}}[\text{BAD}] \cdot \left(\Pr_F[T_F = \sigma] - \Pr_{S,\mathbf{k}}[T_{\mathcal{F}_1} = \sigma \mid \text{BAD}] \right) \right| \quad (1) \end{aligned}$$

$$+ \left| \sum_{\sigma \in \Gamma} \Pr_{S,\mathbf{k}}[\neg \text{BAD}] \cdot \left(\Pr_F[T_F = \sigma] - \Pr_{S,\mathbf{k}}[T_{\mathcal{F}_1} = \sigma \mid \neg \text{BAD}] \right) \right|. \quad (2)$$

Lemma 2 implies that (2) = 0, because $\Pr_F[T_F = \sigma] = 2^{-qmb}$ for any possible transcript σ . We rewrite (1) as

$$\left| \sum_{\sigma \in \Gamma} \left(\Pr_{S,\mathbf{k}}[\text{BAD}] \cdot \Pr_F[T_F = \sigma] \right) - \sum_{\sigma \in \Gamma} \left(\Pr_{S,\mathbf{k}}[\text{BAD}] \cdot \Pr_{S,\mathbf{k}}[T_{\mathcal{F}_1} = \sigma \mid \text{BAD}] \right) \right|.$$

Each of the two summations is bounded by $\alpha := \max_{\sigma \in \Gamma} \left(\Pr_{S,\mathbf{k}}[\text{BAD}] \right)$, since each is a convex combination of numbers that are bounded by α . Thus, the absolute value of their difference is bounded by α as well, and $\alpha < O(r^2 m^3 q^3) \cdot 2^{-b}$ by Lemma 1. \square

3 Conclusion and Future Work

Two obvious directions for future work are to extend the analysis of \mathcal{F}_1 to handle inverse queries (necessarily choosing the S-box as a random *permutation*), and to extend Theorem 6 to prove almost d -wise independence of \mathcal{F}_3 for $d > 3$. A

more foundational question left unanswered is to understand how the degree of each output bit of an SPN (as a polynomial in the input bits) is affected by the degree of the S-box and by the “mixing” properties of the linear transformation.

Exploring other choices of the S-box besides inversion may lead to more efficient constructions, and utilizing other properties of the linear transformation besides maximal-branch-number may allow stronger proofs of security. This could potentially give a (plausibly secure) SPN computable by circuits of size $O(n)$. Recall from §1 that a PRF computable with size $O(n)$ and with security 2^n would bring the Natural Proofs barrier to the current frontier of lower bounds against unbounded-depth circuits.

Abstracting from the SPN structure, one may arrive to the following paradigm for constructing PRF: alternate the application of (1) an error-correcting code and (2) a bundle-wise application of any local map that has high degree over GF(2) and resists attacks corresponding to linear and differential cryptanalysis. This viewpoint may lead to a PRF candidate computable in ACC^0 , since for (1) one just needs parity gates, while, say, taking parities of suitable mod 3 maps one should get a map that satisfies (2). However a good choice for this latter map is not clear to us at this moment.

We believe a good candidate PRF should be the simplest candidate that resists known attacks. As noted in [10], some of the choices in the design of AES are not motivated by any known attack, but are there as a safeguard (for example, one can reduce the number of rounds and still no attack is known). While this is comprehensible when having to choose a standard that is difficult to change or when deploying a system that is to be widely used, one can argue that a better way for the research community to proceed is to put forth the simplest candidate PRF, possibly break it, and iterate until hopefully converging to a secure PRF. We view this paper as a step in this direction.

Acknowledgments. We thank Guevara Noubir for helpful discussions, and Salil Vadhan for mentioning AES. We also thank the anonymous referees for very helpful feedback, including pointing us to [26].

References

1. Aaronson, S., Wigderson, A.: Algebraization: a new barrier in complexity theory. In: 40th ACM Symp. on the Theory of Computing, STOC, pp. 731–740 (2008)
2. Allender, E., Koucký, M.: Amplifying lower bounds by means of self-reducibility. J. of the ACM 57(3) (2010)
3. Alon, N., Goldreich, O., Hästad, J., Peralta, R.: Simple constructions of almost k -wise independent random variables. Random Structures & Algorithms 3(3), 289–304 (1992)
4. Baker, T., Gill, J., Solovay, R.: Relativizations of the $P=?NP$ question. SIAM J. Comput. 4(4), 431–442 (1975)
5. Bazzi, L.M.J.: Polylogarithmic independence can fool DNF formulas. SIAM J. Comput. 38(6), 2220–2272 (2009)
6. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)

7. Braverman, M.: Poly-logarithmic independence fools AC^0 circuits. In: 24th IEEE Conf. on Computational Complexity, CCC. IEEE (2009)
8. Brodsky, A., Hoory, S.: Simple permutations mix even better. Random Struct. Algorithms 32(3), 274–289 (2008)
9. Cho, H.-S., Sung, S.H., Kwon, D., Lee, J.-K., Song, J.H., Lim, J.: New Method for Bounding the Maximum Differential Probability for SPNs and ARIA. In: Park, C., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 21–32. Springer, Heidelberg (2005)
10. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
11. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. J. Cryptology 10(3), 151–162 (1997)
12. Gao, S., von zur Gathen, J., Panario, D., Shoup, V.: Algorithms for exponentiation in finite fields. J. Symb. Comput. 29(6), 879–889 (2000)
13. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl: a SHA-3 candidate (2011), <http://www.groestl.info>
14. Gentry, C., Ramzan, Z.: Eliminating Random Permutation Oracles in the Even-Mansour Cipher. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 32–47. Springer, Heidelberg (2004)
15. Gerasoulis, A.: A fast algorithm for the multiplication of generalized Hilbert matrices with vectors. Mathematics of Computation 50, 179–188 (1988)
16. Goldreich, O.: Foundations of Cryptography: Volume 1, Basic Tools. Cambridge University Press (2001)
17. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. of the ACM 33(4), 792–807 (1986)
18. Goldreich, O., Levin, L.: A hard-core predicate for all one-way functions. In: 21st ACM Symp. on the Theory of Computing, STOC, pp. 25–32 (1989)
19. Gowers, W.: An almost m -wise independent random permutation of the cube. Combinatorics, Probability and Computing 5(2), 119–130 (1996)
20. Haitner, I., Reingold, O., Vadhan, S.P.: Efficiency improvements in constructing pseudorandom generators from one-way functions. In: 42nd ACM Symp. on the Theory of Computing, STOC, pp. 437–446 (2010)
21. Hästad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (1999)
22. Healy, A., Viola, E.: Constant-Depth Circuits for Arithmetic in Finite Fields of Characteristic Two. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 672–683. Springer, Heidelberg (2006)
23. Hesse, W., Allender, E., Barrington, D.A.M.: Uniform constant-depth threshold circuits for division and iterated multiplication. J. Comput. System Sci. 65(4), 695–716 (2002); Special issue on complexity, 2001 (Chicago, IL)
24. Hoory, S., Magen, A., Myers, S., Rackoff, C.: Simple permutations mix well. Theor. Comput. Sci. 348(2-3), 251–261 (2005)
25. Jakobsen, T., Knudsen, L.: Attacks on block ciphers of low algebraic degree. Journal of Cryptology 14, 197–210 (2001)
26. Kang, J.S., Hong, S., Lee, S., Yi, O., Park, C., Lim, J.: Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks. ETRI Journal 23(4), 158–167 (2001)
27. Keliher, L., Meijer, H., Tavares, S.: New Method for Upper Bounding the Maximum Average Linear Hull Probability for SPNs. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 420–436. Springer, Heidelberg (2001)

28. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
29. Kopparty, S.: On the complexity of powering in finite fields. In: ACM Symp. on the Theory of Computing, STOC (2011)
30. Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press (1997)
31. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Comput. 17(2), 373–386 (1988)
32. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
33. Naor, J., Naor, M.: Small-bias probability spaces: efficient constructions and applications. SIAM J. Comput. 22(4), 838–856 (1993)
34. Naor, M., Reingold, O.: On the construction of pseudorandom permutations: Luby-Rackoff revisited. J. Cryptology 12(1), 29–66 (1999)
35. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. of the ACM 51(2), 231–262 (2004)
36. Nyberg, K.: Differentially Uniform Mappings for Cryptography. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 55–64. Springer, Heidelberg (1994)
37. Pieprzyk, J.: On bent permutations. In: Proceedings of the International Conference on Finite Fields, Coding Theory, and Advances in Communications and Computing, Las Vegas (August 1991)
38. Ramzan, Z., Reyzin, L.: On the Round Security of Symmetric-Key Cryptographic Primitives. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 376–393. Springer, Heidelberg (2000)
39. Razborov, A., Rudich, S.: Natural proofs. J. of Computer and System Sciences 55(1), 24–35 (1997)
40. Razborov, A.A.: A simple proof of Bazzi's theorem. ACM Transactions on Computation Theory (TOCT) 1(1) (2009)
41. Roth, R.M., Seroussi, G.: On generator matrices of MDS codes. IEEE Transactions on Information Theory 31, 826–830 (1985)
42. Shannon, C.: Communication theory of secrecy systems. Bell Systems Technical Journal 28(4), 656–715 (1949)
43. Vadhan, S.P., Zheng, C.J.: Characterizing pseudoentropy and simplifying pseudo-random generator constructions. In: ACM Symp. on the Theory of Computing, STOC (2012)
44. Williams, R.: Non-uniform ACC lower bounds. In: IEEE Conf. on Computational Complexity, CCC (2011)
45. Wu, H.: The hash function JH (2011),
<http://www3.ntu.edu.sg/home/wuhj/research/jh/index.html>

The End of Crypto

Jonathan Zittrain

Professor of Law School and Professor of Computer Science, Harvard University, USA

Abstract. This talk will reflect on the core purposes of cryptology, and the extent to which those purposes are served – and servable – in today’s digital environment.

Must You Know the Code of f to Securely Compute f ?

Mike Rosulek^{*}

Department of Computer Science, University of Montana
`mikero@cs.umt.edu`

Abstract. When Alice and Bob want to securely evaluate a function of their shared inputs, they typically first express the function as a (boolean or arithmetic) circuit and then securely evaluate that circuit, gate-by-gate. In other words, a secure protocol for evaluating f is typically obtained in a *non-black-box-way* from f itself. Consequently, secure computation protocols have high overhead (in communication & computation) that is directly linked to the circuit-description complexity of f .

In other settings throughout cryptography, black-box constructions invariably lead to better practical efficiency than comparable non-black-box constructions. Could secure computation protocols similarly be made more practical by eliminating their dependence on a circuit representation of the target function? Or, in other words, *must one know the code of f to securely evaluate f ?*

In this work we initiate the theoretical study of this question. We show the following:

1. A complete characterization of the 2-party tasks which admit such security against semi-honest adversaries. The characterization is inspired by notions of *autoreducibility* from computational complexity theory. From this characterization, we show a class of pseudorandom functions that *cannot* be securely evaluated (when one party holds the seed and the other holds the input) without “knowing” the code of the function in question. On the positive side, we show a class of functions (related to blind signatures) that can indeed be securely computed without “knowing” the code of the function.
2. Sufficient conditions for such security against malicious adversaries, also based on autoreducibility. We show that it is not possible to prove membership in the image of a one-way function in zero-knowledge, without “knowing” the code of the one-way function. We also describe a variant of the GMW compiler for transforming semi-honest to malicious security while preserving the specific black-box property considered here.

1 Introduction

In cryptography, a **black-box** construction is one that uses only the input/output behavior of its components [21,28]. By contrast, a non-black-box construction

* Supported by NSF grant CCF-1149647.

relies on the *code* of its components. Understanding exactly when non-black-box techniques are necessary is important for cryptography, since black-box constructions are typically much more efficient (in their computation and/or communication) than comparable non-black-box constructions.

Secure multi-party computation (MPC) allows mutually distrusting parties to securely evaluate a function f on their shared inputs. This powerful paradigm is well-known in the theoretical community but appears to be seldom used in practice. As a result, much current work focuses on improving the efficiency of MPC constructions to facilitate more widespread use. A recent line of work (see [20] and followup works [22,17,26]) has focused on improving efficiency by removing certain non-black-box techniques used in all earlier work. In particular, these results focus on the black-box use of the underlying *cryptographic primitives* (that is, one-way functions, trapdoor permutations, or standalone-secure oblivious transfer) used in the protocol.

One goal in this paper is to make explicit another non-black-box step inherent to all existing general-purpose MPC protocols. To build a secure protocol for a function f , the function must first be expressed as a low-level circuit ([30,19] use boolean circuits, [16,9] use arithmetic circuits, and [23] uses branching programs). Then, the protocol proceeds to securely evaluate the circuit, gate by gate. In other words, a secure protocol for f is *non-black-box in its usage of f itself*. While this framework provides a straight-forward way to achieve complete generality, it also inherently ties the efficiency (communication, computation, or both) of the protocol to the circuit-representation complexity of f . For this reason, an important line of research has streamlined many aspects of this non-black-box dependence, including techniques for optimizing circuits for MPC [24,27] and exploring alternative circuit representations [3].

It is unreasonable to expect that we can avoid this non-black-box step for *general-purpose* MPC (indeed, our results explicitly confirm this). Still, it is important to understand exactly to what extent the non-black-box dependence is inherent. For which *special-purpose* secure computation tasks can we avoid dependence on the code of the target function altogether (and hopefully construct highly efficient protocols)?

1.1 Overview of the Results

We initiate the theoretical study of when a protocol can securely compute f without “knowing” the code of f . When considering a (standard) secure protocol for a functionality f , that choice of f is fixed and the protocol is allowed to depend arbitrarily on f . In this case, it is not meaningful to place any syntactic restrictions on the protocol (e.g., that it use oracle access to a subroutine implementing f), since the protocol could have a circuit for f hard-coded anyway.

Instead, we model a protocol that “does not know” the code of its target functionality in the following way. The protocol is a pair of oracle machines that, when instantiated with any f from some larger *class* of functionalities, securely emulates a functionality related to f . By considering large classes of

functionalities, we prevent the protocol from hard-coding relevant circuit representations of the functionalities.

Definition (Informal). Let \mathcal{F} be an ideal (2-party) functionality implemented as an oracle machine, and let \mathcal{C} be a class of functions. Then a **functionally-black-box (FBB)** protocol for $\mathcal{F}^{\mathcal{C}}$ is a pair of oracle machines (π_A, π_B) such that, for all $f \in \mathcal{C}$, the instantiated protocol (π_A^f, π_B^f) is a secure protocol for \mathcal{F}^f .

As a natural example, \mathcal{C} can be a class of functions that take two inputs, and \mathcal{F} can be the simple functionality which collects x from Alice, y from Bob, queries its oracle to obtain $z = f(x, y)$, and gives the result to both parties (secure function evaluation). Or, \mathcal{F} can be the functionality which collects (x, w) from Alice, and then gives x to Bob if $f(x, w) = 1$ (zero-knowledge proof).

We point out that it is only the *protocol* which must treat f as a black-box. In particular, the order of quantifiers is such that adversaries and simulators attacking the f -instantiated protocol may depend arbitrarily on the choice of f (and hence could be said to “know” the code of f).

We put forth the FBB property as a *necessary* condition for highly efficient, practical MPC protocols. This work therefore focuses on a theoretical understanding of the FBB property, as a *proxy* for practical efficiency. However, FBB alone is not a sufficient condition for practical protocols. Indeed, the protocols that we construct in this work may not be considered “practical.”

Autoreducibility Characterization for 2-party Passive Security. In computational complexity theory, the notion of *autoreducibility* is a way to measure the “structure” within a set. Very generally, a set A is autoreducible if there exists an oracle machine M such that $M^A(x) = A(x)$, and yet M ’s oracle queries do not “depend too much” on x . An instance of autoreducibility which shows up frequently in cryptography is the notion of *random self-reducibility*. In that case, the oracle queries of M are distributed independently of the input x .

We define a variant of autoreducibility called **2-hiding autoreducibility**, which subsumes random self-reducibility and has some similarities to “2-oracle instance-hiding” autoreducibility defined by Beaver & Feigenbaum [5]. Intuitively, 2-hiding autoreducibility requires a single oracle machine M such that $M^f(x, y) = f(x, y)$ for every f in a large class \mathcal{C} ; furthermore, half of M ’s oracle queries are “independent” of x and the other half are “independent” of y , in some sense. We then show that 2-hiding autoreducibility completely characterizes FBB feasibility (for 2-party deterministic secure function evaluation):

Theorem (Informal). Let \mathcal{C} be a class of 2-input functions. Then functionally-black-box secure evaluation of \mathcal{C} is possible against semi-honest adversaries, in the presence of an arbitrary trusted setup, if and only if \mathcal{C} is 2-hiding autoreducible.

We also emphasize that *achieving the FBB property is not simply a matter of providing a very powerful trusted setup to the parties*. Indeed, to be meaningful, the trusted setup must be the same for every $f \in \mathcal{C}$. Since it is only the parties

and not the trusted setup that have access to f , it is not immediate that even a powerful setup can be useful. Subsequently, our characterization can be used to give impossibility results that hold even in the presence of an arbitrary setup. Without loss of generality, one can assume the presence of a *complete* trusted setup such as the oblivious transfer functionality [22], or a common reference string [14].

Non-trivial FBB Feasibility. There is a trivial sense in which some classes of functionalities admit FBB protocols. We say (informally) that a class \mathcal{C} is *learnable* if it is possible to efficiently obtain a circuit for f using only oracle access for f , for every $f \in \mathcal{C}$. Then every learnable class admits an FBB protocol of the following form: the parties independently query f to obtain a (canonical) circuit that evaluates f ; they then use a standard general-purpose construction such as [19,14,22]. Thus, the notion of functionally-black-box is most meaningful when considering non-learnable classes of functionalities. (We note that a similar triviality occurs in the context of obfuscation [4], with respect to this same notion of learnability.) Informally, it is possible to “securely compute f without knowing the code of f ” if f belongs to some class \mathcal{C} that admits FBB secure protocols but is not learnable from oracle queries.

Using our autoreducibility characterization, we explicitly show that non-trivial FBB protocols are possible. That is, learnability is a strictly stronger condition than FBB feasibility. Intuitively, learnability requires the *entire* function to be deduced from oracle access, while our autoreducibility characterization only requires M to deduce the correct answer on a single, given input. We show a class of functions (related to blind signatures) that admits FBB protocols, but is not explicitly learnable from oracle queries. Thus, in some cases it is in fact possible to securely compute f “without knowing” the code of f .

Infeasibility of PRF Evaluation. As another demonstration of our autoreducibility characterization, we show that it is impossible to securely evaluate arbitrary PRFs (where one party holds the seed and the other holds the PRF input)¹ in an FBB manner. Impossibility holds even if arbitrary trusted setups are available, and even if security is only required against semi-honest adversaries. The result also easily extends to the class of (strong) PRPs. We leave open the question of whether a natural *subclass* of PRFs admits FBB-secure protocols (in a non-trivial way).

Sufficient Conditions for Malicious Security. We define another variant of autoreducibility, called 1-hiding autoreducibility. The definition is similar to that of 2-hiding autoreducibility, except that in 1-hiding autoreducibility all of the oracle machine’s queries are independent of both x and y . We then show that 1-hiding autoreducible is a sufficient condition for FBB feasibility against *malicious* adversaries.

¹ Evaluating the AES block cipher in this way is now a relatively standard performance benchmark for MPC protocols [25].

We also show a variant of the GMW compiler to convert semi-honest-secure to malicious-secure protocols, in a way that preserves the FBB property. Just as the GMW compiler uses zero-knowledge proofs as its main component, we require an FBB protocol for the simple functionality that takes input x from Alice and gives $f(x)$ to Bob (FBB with respect to $f \in \mathcal{C}$). This functionality can be considered a zero-knowledge proof of membership in the image of f .

Zero-Knowledge. Beyond the GMW-inspired application described above, zero-knowledge proofs are interesting in their own right. ZK proofs are often the sole source of non-black-box behavior (and thus the efficiency bottleneck) in a protocol.

Let f be a deterministic function and define the relation $R_f(x, w) = 1 \Leftrightarrow f(w) = x$. We show that FBB zero-knowledge proofs are impossible for the class of relations $\{R_f \mid f \text{ is a OWF}\}$. In fact, our impossibility result is much stronger, ruling out even honest-verifier witness-hiding (instead of zero-knowledge), arguments rather than proofs; impossibility further applies to basic standalone security, and holds in the presence of arbitrary trusted setups.

1.2 Other Related Work

The notion of autoreducibility has proven to be a fruitful tool in complexity theory for quantifying the amount of structure in a set. For a gentle introduction to research on this topic, see [29,2].

In cryptography, autoreducibility has already been recognized as a tool for several interesting applications. Abadi, Feigenbaum, & Kilian [1] used “instance hiding” autoreducibility to reason about what would in today’s parlance be called a form of outsourced computation. Here, a client wishes to compute $f(x)$ using access to a powerful trusted server who can evaluate the function f ; the client wants to learn $f(x)$ but does not want his queries to f to leak any information about x . The notion was later extended by Beaver & Feigenbaum [5] to a setting involving multiple (non-colluding) servers. A summary of this line of work was given by Brassard [12]. Beaver *et al.* [6] also used autoreducibility to construct efficient perfect zero-knowledge proofs.

The question of FBB protocols requires a fundamentally different style of autoreducibility than studied in previous works. First, existing definitions consider an oracle machine which is required to work only for a single fixed language (or function); whereas here we require a single oracle machine which works for any function from a large class. Second, we must explicitly consider functions on two inputs, and make a distinction between oracle queries that depend on each of the inputs. In addition, our notion of “query independence” is specific to our application of secure protocols. Finally, many previous definitions of autoreducibility allow the oracle machine to be instantiated with an oracle that is distinct from (but depends on) the required output function — e.g., an oracle machine computes the function f when given oracle access to g , a low-degree encoding of f .

2 Preliminaries

A probability $p(n)$ is negligible if for all, $c > 0$, $p(n) < n^{-c}$ for all but finitely many n . We write $p(n) \approx q(n)$ if $|p(n) - q(n)|$ is negligible. A probability $p(n)$ is overwhelming if $p(n) \approx 1$.

When discussing security of MPC protocols, we use Canetti's framework of Universally Composable (UC) security [13]. The low-level details of the security model are not crucial to our results. We do, however, make one distinction about notation:

In the MPC literature, the notation π^f often refers to a protocol π where the parties can use a *shared* ideal functionality f (the f -hybrid model, in UC parlance). In this work, write π^f to instead denote a protocol machine equipped with (local) oracle access to *independent* instances of f . In that sense, our notation reflects the complexity-theoretic idea of an oracle computation.

Definition 1 (Related-Key security for PRFs [8]). Let Φ be a class of functions over $\{0, 1\}^k$. Then $F : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}^k$ is a **pseudorandom function secure under Φ -related-key attacks (Φ -RKA-PRF)** if for all efficient M , $|\Pr[M^{\mathcal{O}}(1^k) = b] - \frac{1}{2}|$ is negligible in k , in the following game:

Bit b is chosen at random and s is chosen uniformly from $\{0, 1\}^k$. If $b = 0$ then queries of the form $\mathcal{O}(\phi, x)$, where $\phi \in \Phi$ and $x \in \{0, 1\}^k$, are answered as $F(\phi(s), x)$. If $b = 1$ then the query $\mathcal{O}(\phi, x)$ is answered as $R_{\phi(s)}(x)$, where for each v , R_v is chosen as a random function from $\{0, 1\}^k \rightarrow \{0, 1\}^k$.

If \otimes is a group operation on $\{0, 1\}^k$, and $\Phi = \{\phi_\Delta \mid \Delta \in \{0, 1\}^k\}$ where $\phi_\Delta(x) = x \otimes \Delta$, then we say that Φ is **group-induced**. Bellare & Cash [7] give constructions of Φ -RKA-PRFs (and PRPs) for group-induced Φ .

Definition 2 (Blind signatures [15]). Let $\Sigma = (\text{KeyGen}, \text{Sign}, \text{Ver})$ be a signature scheme. A **blind signature protocol** for Σ is one in which the client has input $(1^k, m)$, the signer has input $(1^k, sk)$, and the client receives output $\text{Sign}(sk, m)$. The blindness condition is that the view of the signer is statistically independent of m .

3 Functionally Black-Box Protocols

We now give the formal definition of MPC protocols that “do not know” the code of their target function f , as described in the introduction:

Definition 3. Let \mathcal{C} be a class of functions, and let \mathcal{F} be an ideal functionality that is implemented as an (uninstantiated) oracle machine. Let π_A and π_B be interactive oracle machines. Then we say that $\pi = (\pi_A, \pi_B)$ is a **functionally-black-box (FBB) protocol** for $\mathcal{F}^\mathcal{C}$ in a certain security model if, for all $f \in \mathcal{C}$, the protocol (π_A^f, π_B^f) is a secure protocol (in the model in question) for the ideal functionality \mathcal{F}^f .

Importantly, the definition defers to the security condition *separately* for each instantiation (π_A^f, π_B^f) . Thus, adversaries and simulators attacking the f -instantiated protocol in the security definition can depend arbitrarily on the choice of $f \in \mathcal{C}$. This models the fact that adversaries are not restricted to use f as a black-box. Indeed, the intent is to characterize convenience/efficiency for the honest parties, without compromising any level of security.

Instantiations Used in This Work. Let \mathcal{F}_{SFE} be the non-reactive functionality that takes input x from Alice and y from Bob, queries its oracle f to obtain $z = f(x, y)$, and gives the output to both parties. Then an FBB protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ would allow the protocols to have only black-box access to the function f being evaluated.

As another example, let \mathcal{F}_{FZK} be the functionality that takes input w from Alice, queries its oracle f to obtain $x = f(w)$, and gives output x to Bob. When instantiated with function f , $\mathcal{F}_{\text{FZK}}^f$ is essentially a zero-knowledge argument (of knowledge) functionality for statements of the form “ $\exists w : x = f(w)$ ”. Note that in this setting we allow parties to have access to the function f rather than the (more restricted) NP relation $R_f(x, w) = 1 \Leftrightarrow f(w) = x$.

Simple Observations. Suppose \mathcal{C} is **learnable** in the following sense. There exists a PPT oracle TM M such that with overwhelming probability (in k), $M^f(1^k, 1^t)$ outputs a circuit that agrees with f on $\{0, 1\}^t$, for all $f \in \mathcal{C}$. In the simple case where M will always output a *canonical* circuit, then an FBB protocol can be obtained by having both parties (independently) running M , and then using an appropriate non-FBB protocol construction on the resulting circuit. Even if M does not reliably output the same circuit each time, an FBB protocol can be obtained using the approach outlined later in Theorem 2 (the class is 1-hiding autoreducible via the oracle machine which runs $C \leftarrow M^f(1^k, 1^{|x|+|y|}); C(x, y)$).

Suppose \mathcal{C} contains only functions whose input domains are **constant-sized** (i.e., not growing in size with k). Then \mathcal{C} is (canonically) learnable in the above sense, by exhaustively querying the function. For this reason, we only consider classes which contain functions over infinite domains.

4 Classification for Semi-honest Security

In this section we define a notion of autoreducibility, and then show that it completely characterizes feasibility of FBB MPC protocols against semi-honest adversaries.

Definition 4. Let \mathcal{C} denote a class of functions on two inputs. Let M be a PPT oracle machine, and suppose each oracle query is tagged with a label $i \in \{1, 2\}$ (say, by setting some internal variable at the time of the query). Then for $i \in \{1, 2\}$ define $\mathcal{Q}_i[M, f; z]$ to be the random variable containing the sequence of oracle queries made with label i during the computation $M^f(z)$. We say that \mathcal{C} is **2-hiding autoreducible** if there exists a PPT oracle machine M such that for all $f \in \mathcal{C}$:

1. For all x, y , we have that $\Pr[M^f(1^k, x, y) = f(x, y)]$ is overwhelming in k .
2. There exists a PPT machine $\mathcal{S}_{f,1}$ such that for all x, y , the following ensembles are indistinguishable (in k):

$$\{\mathcal{S}_{f,1}(1^k, f(x, y), x)\}_k \quad \text{and} \quad \{\mathcal{Q}_1[M, f; 1^k, x, y]\}_k.$$

3. There exists a PPT machine $\mathcal{S}_{f,2}$ such that for all x, y , the following ensembles are indistinguishable (in k):

$$\{\mathcal{S}_{f,2}(1^k, f(x, y), y)\}_k \quad \text{and} \quad \{\mathcal{Q}_2[M, f; 1^k, x, y]\}_k.$$

In other words, M is able to use oracle access to f to determine $f(x, y)$, yet its type-1 oracle queries to f are “independent” of y and its type-2 queries are “independent” of x , in some sense. A special case is when M ’s type-1 queries are distributed independently of y (and type-2 queries independently of x).

We now give our main classification for semi-honest security, which holds in the presence of arbitrary trusted setups. Without loss of generality, we can take the trusted setup to be the oblivious transfer functionality, which we denote by \mathcal{F}_{OT} .

Theorem 1. *There is a FBB protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ secure against PPT semi-honest adversaries in the \mathcal{F}_{OT} -hybrid model if and only if \mathcal{C} is 2-hiding autoreducible.*

Proof. (\Leftarrow) Suppose that the oracle machine M satisfies the definition of 2-hiding autoreducibility for \mathcal{C} . Without loss of generality, assume that the number of queries made by M depends only on the input 1^k (i.e., it does not depend on x or y), and that M strictly alternates between type-1 and type-2 queries. We then define the ideal functionality \mathcal{F}_M as in Figure 1.

There is a UC-secure protocol for \mathcal{F}_M in the \mathcal{F}_{OT} -hybrid model, so it suffices to design a FBB MPC protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ in the \mathcal{F}_M -hybrid model. (Note that the same \mathcal{F}_M will be used in the protocol for each $f \in \mathcal{C}$.) The FBB protocol for $\mathcal{F}_{\text{SFE}}^f$, $f \in \mathcal{C}$, is relatively straight-forward. On inputs x for Alice and y for Bob, the parties send (INIT, x) and (INIT, y) to \mathcal{F}_M , respectively. Whenever a party receives output (QUERY, q) from \mathcal{F}_M , for $q \neq \perp$, that party uses its local oracle to compute $r = f(q)$, and gives (RESP, r) to \mathcal{F}_M . When the parties receive (OUT, z) from \mathcal{F}_M , they output z .

Clearly the protocol is FBB. To show that this protocol is secure against semi-honest adversaries, it suffices to show that the view of an honest party can be simulated in the ideal world. The simulator for a semi-honest Alice is as follows. Fix $f \in \mathcal{C}$ and recall that the simulator for the f -instantiated protocol can depend arbitrarily on f . The simulator receives Alice’s input x from the environment. When Alice sends (INIT, x) to \mathcal{F}_M , the simulator sends x to the ideal functionality and receives $z = f(x, y)$. The simulator then computes $Q \leftarrow \mathcal{S}_{f,1}(1^k, z, x)$, where $\mathcal{S}_{f,1}$ is the machine guaranteed by the 2-hiding autoreducibility definition. For each query q in the sequence Q , the simulator gives (QUERY, q) and then (QUERY, \perp) to Alice on behalf of \mathcal{F}_M . Finally, the simulator gives output (OUT, z) to Alice on behalf of \mathcal{F}_M . It is clear that Alice’s view

The functionality maintains a configuration of the machine M in an internal variable S . This variable is manipulated by commands from Alice and Bob, as described below. After every such input, the functionality simulates M with configuration S . If M terminates with output z , give output (OUT, z) to both parties and halt. If M queries its oracle with a type-1 query q , then give output (QUERY, q) to Alice and (QUERY, \perp) to Bob, and wait. If M queries its oracle with a type-2 query q , then give output (QUERY, \perp) to Alice and (QUERY, q) to Bob, and wait.

1. On inputs (INIT, x) from Alice and (INIT, y) from Bob, initialize S as the initial configuration of oracle machine M on input $(1^k, x, y)$, where k is the global security parameter. Then simulate M as described above.
2. On input (RESP, r) from Alice, ensure that in configuration S , M is awaiting a reply to a type-1 oracle query (otherwise abort). Update S to reflect a response r on the oracle response tape, then simulate M as described above.
3. On input (RESP, r) from Bob, do the same as the previous step except ensure that M is awaiting a reply to a type-2 oracle query.

Fig. 1. Ideal functionality \mathcal{F}_M , parameterized by oracle PPT machine M

is computationally indistinguishable from that of the real interaction, by the guarantees of $\mathcal{S}_{f,1}$. The protocol is essentially symmetric with respect to Alice and Bob, and so security holds for a semi-honest Bob as well. Note that since we consider only semi-honest security, the adversary will indeed provide correct oracle responses to \mathcal{F}_M . Indeed, a malicious party could easily invalidate this security argument by providing incorrect oracle responses to \mathcal{F}_M .

(\Rightarrow) Suppose $\pi = (\pi_A, \pi_B)$ is an FBB protocol for $\mathcal{F}_{\text{SFE}}^C$ in the \mathcal{F}_{OT} -hybrid model. Define an oracle machine M as follows: It internally simulates instances of π_A , π_B , and \mathcal{F}_{OT} as in their protocol interaction. On input $(1^k, x, y)$ to M , it gives input $(1^k, x)$ to π_A and input $(1^k, y)$ to π_B . It then simulates the three sub-components in the natural way. Whenever π_A makes an oracle query, M makes the same query as a type-1 query and returns the result to the π_A component. Similarly, queries made by π_B are made as type-2 queries. The final output of π_A is taken to be the output of M .

By the correctness of the protocol π , we have that the output of M is equal to $f(x, y)$ with overwhelming probability for all x, y , when instantiated with f as its oracle. Now fix a particular $f \in \mathcal{C}$. The security of π instantiated with f implies the existence of a simulator \mathcal{S} in the $\mathcal{F}_{\text{SFE}}^f$ -ideal model. We define $\mathcal{S}_{f,1}$ to do the following, on input $(1^k, f(x, y), x)$: First, run \mathcal{S} on input $(1^k, f(x, y))$ against a semi-honest corrupt Alice with input $(1^k, x)$. Then \mathcal{S} generates a simulated view for Alice; output the sequence of simulated oracle queries contained in Alice's view. By the soundness of \mathcal{S} , the output of $\mathcal{S}_{f,1}$ is indistinguishable from the sequence of type-1 queries made by M , as required for 2-hiding autoreducibility. The required $\mathcal{S}_{f,2}$ algorithm is defined symmetrically.

Discussion. The protocol for realizing \mathcal{F}_M in the \mathcal{F}_{OT} -hybrid model uses the oracle machine M in a highly non-black-box manner. So while the protocol is black-box in the code of $f \in \mathcal{C}$, it is not black-box in the code of M . However, we note that the code of M may be significantly simpler than that of $f \in \mathcal{C}$ (for example, when M 's oracle queries are made uniformly), and also that M is fixed for the entire class \mathcal{C} .

4.1 A Positive Example, and Comparison to Learnability

Using our characterization, we can show that FBB feasibility is a *strictly weaker* condition than learnability (as defined in Section 3). In other words, it is indeed possible to securely evaluate certain classes of functions without “knowing” the code of the function.

Let $\Sigma = (\text{KeyGen}, \text{Sign}, \text{Ver})$ be an existentially unforgeable signature scheme. Without loss of generality, we assume that the **Sign** algorithm is deterministic (it can be derandomized using a PRF). Let (π_C, π_S) denote a blind signature protocol for this scheme (Definition 2), where C is the client and S is the signer. We call a blind signature protocol **modular** if the π_S protocol does not use the signing key except via oracle access to $\text{Sign}(sk, \cdot)$. That is, the signer executes the protocol as $\pi_S^{\text{Sign}(sk, \cdot)}(1^k, vk)$. As a concrete example, the Boneh-Lynn-Shacham signature scheme [11] supports such a blind signature protocol [10].

For a signing key sk , define the function $S_{sk}(x, y) = \text{Sign}(sk, x)$.

Lemma 1. *If Σ has a modular and semi-honest secure blind signature protocol, then the class $\mathcal{C}_\Sigma = \{S_{sk} \mid sk \in \{0, 1\}^*\}$ is 2-hiding autoreducible but not learnable. (In fact, the class \mathcal{C}_Σ is 1-hiding autoreducible; defined later in Section 5.)*

Proof. We construct a machine M for the definition of 2-hiding autoreducibility. On input $(1^k, x, y)$, the machine M simulates instances of $\pi_C(1^k, vk, x)$ and $\pi_S^{\text{Sign}(sk, \cdot)}(1^k, vk)$ in the natural way. Whenever π_S queries its oracle on message m , M makes a type-1 query (m, \perp) and uses the result as the response to π_S . Finally, M uses the final output of π_C as its own output.

By the correctness of the π protocol, M satisfies the desired correctness condition for 2-hiding autoreducibility. From the blindness property of π , it follows that the oracle queries made by π_S in the protocol are distributed independently of x . Hence, the entire set of queries made by M (all type-1) are distributed independently of (x, y) .

The fact that \mathcal{C}_Σ is not learnable follows from the existential unforgeability of Σ . Suppose a machine M , using oracle access to S_{sk} for a randomly chosen sk , is able to output a circuit correctly computing S_{sk} . Then the following is an attack in the unforgeability game for Σ : On input $(1^k, vk)$ run $M(1^k)$. Whenever M makes an oracle query (x, y) , request a signature on x in the game and return the result to M . When M outputs a circuit C , choose any x^* such that no query

of the form (x^*, \cdot) was ever made. Then run $C(x^*, \perp)$, which by assumption is a valid signature on x^* ; hence, a forgery.²

4.2 A Negative Example: Infeasibility of PRFs

In this section, we treat pseudorandom functions as functions of two arguments: the first argument being the seed and the second argument being the PRF input. Thus, a functionality evaluating a PRF in our terminology corresponds to a functionality which takes a seed from Alice and an input from Bob, and evaluates the PRF accordingly.

We now show that FBB protocols are impossible for the class of all pseudorandom functions. While this claim is sensible at an intuitive level (pseudorandomness precludes significant structure like that required for 2-hiding autoreducibility), the proof has some subtlety. To apply the security of the PRF we must have its seed (secretly) chosen at random, whereas in the 2-hiding autoreducibility definition, the oracle machine is given Alice’s input (taken to be the PRF seed) and can arbitrarily query the *unseeded* PRF $f(\cdot, \cdot)$ as an oracle. We show that, given a PRF secure against *related-key attacks*, we can “embed” an additional seed into the PRF oracle in a way that allows the PRF security to apply to the 2-hiding autoreducibility interaction.

Lemma 2. *Define $\mathcal{C}_{\text{PRF}} = \{f \mid f \text{ is a PRF}\}$. If Φ -RKA-secure PRFs exist for group-induced Φ (Definition 1), and injective PRGs exist, then \mathcal{C}_{PRF} is not 2-hiding autoreducible, and thus there is no FBB protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\text{PRF}}}$, even against semi-honest adversaries and in the \mathcal{F}_{OT} -hybrid model.*

Additionally, we point out that the proof goes through with minimal modification with respect to the class of pseudorandom *permutations* (as before, assuming the existence of RKA-secure PRPs). Regarding the condition in the lemma statement, Bellare & Cash [7] give constructions of suitable PRFs (and PRPs) under either the DDH or DLIN assumptions, and also assuming the existence of collision-resistant hash functions.

Proof. Let f be a PRF secure against group-induced RKA attacks. For concreteness and clarity, we write the allowed group operation as \oplus . Let $g : \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ be an injective PRG and define the following function for an arbitrary string s :

$$f_s(x, y) = f(s \oplus g(x), g(y)).$$

So that f_s is defined for inputs of arbitrary length, we assume that the fixed string s is padded with zeroes or truncated to the appropriate length ($|g(x)|$) in the expression $s \oplus g(x)$. Now we claim that for each (fixed) string $s \in \{0, 1\}^*$,

² The same argument holds with a significantly weaker requirement on learnability — the circuit C need only agree with S_{sk} on some noticeable fraction of inputs, and this only with noticeable probability.

the function f_s is a PRF (interpreting x as its seed). Consider an efficient oracle machine A . We have:

$$\begin{aligned} \Pr_{x \leftarrow \{0,1\}^k} [A^{f(s \oplus g(x), \cdot)}(1^k) = 1] &\approx \Pr_{x' \leftarrow \{0,1\}^{2k}} [A^{f(s \oplus x', \cdot)}(1^k) = 1] \\ &\approx \Pr_{x' \leftarrow \{0,1\}^{2k}} [A^{f(x', \cdot)}(1^k) = 1] \approx \Pr_R [A^R(1^k) = 1]. \end{aligned}$$

In the above, R is a random oracle; thus f_s is a PRF. Indistinguishability holds due to the pseudorandomness of g , the fact that \oplus is a group operation, and the pseudorandomness of f , respectively.

Suppose for the sake of contradiction that oracle machine M satisfies the condition required for 2-hiding autoreducibility of the class of pseudorandom functions. Then for *every* PRF h and all strings x and y , we have that $\Pr[M^h(1^k, x, y) = h(x, y)]$ is overwhelming in k . In particular, the same probability is overwhelming for random choice of $x, y \in \{0,1\}^k, s \in \{0,1\}^{2k}$, and setting $h = f_s$ (since *each* f_s is a PRF). In the RKA-PRF security game for f , the oracle machine is allowed to make queries of the form (ϕ, z) and obtain either $f(s \oplus \phi, z)$ for randomly chosen s , or $R_\phi(z)$, where each R_ϕ is an independent random function. Hence,

$$\begin{aligned} \Pr_{\substack{x,y \leftarrow \{0,1\}^k \\ s \leftarrow \{0,1\}^{2k}}} [M^{f_s}(1^k, x, y) = f_s(x, y)] &\approx \Pr_{\substack{x,y \leftarrow \{0,1\}^k \\ s \leftarrow \{0,1\}^{2k}}} [M^{R_g(\cdot, \cdot)}(1^k, x, y) = R_{g(x)}(y)] \\ &= \Pr_{x,y \leftarrow \{0,1\}^k} [M^R(1^k x, y) = R(x, y)]. \end{aligned}$$

Here, each R_ϕ , and finally R , is chosen as a random function. The last equality holds from the fact that g is injective.

From this we see that when the machine M is given inputs $x, y \in \{0,1\}^k$ chosen randomly, and an oracle f_s with $s \in \{0,1\}^{2k}$ chosen randomly, it must query the oracle on (x, y) with overwhelming probability. By an averaging argument, there must exist a negligible function δ and strings $\{s_k\}_{k \in \mathbb{N}}$, each $s_k \in \{0,1\}^{2k}$, such that:

$$\forall k : \Pr_{x,y \leftarrow \{0,1\}^k} [M^{f_{s_k}}(1^k, x, y) \text{ queries its oracle on } (x, y)] \geq 1 - \delta(k).$$

When M queries its oracle on its given input, this query is either a type-1 or a type-2 query. Thus, there must exist additional $\{b_k\}_{k \in \mathbb{N}}$, each $b_k \in \{1, 2\}$, satisfying:

$$\forall k : \Pr_{x,y \leftarrow \{0,1\}^k} [(x, y) \in \mathcal{Q}_{b_k}[M, f_{s_k}; 1^k, x, y]] \geq \frac{1}{2} - \delta(k).$$

Importantly, even for a fixed s_k , the function f_{s_k} is a PRF and thus in the class \mathcal{C} for which the properties of M apply. So for each s_k , there is a corresponding simulator \mathcal{S}_k that takes input $1^k, f_{s_k}(x, y)$, and either x or y (depending on b_k), and whose output is indistinguishable from $\mathcal{Q}_{b_k}[M, f_{s_k}; 1^k, x, y]$.

Then we can use such a simulator to invert the PRG g with probability essentially half. The attack uses the non-uniform advice $\{(s_k, b_k, \mathcal{S}_k)\}_k$.

On input $(1^k, \beta)$: // where $\alpha \leftarrow \{0, 1\}^k$ and $\beta = f(\alpha)$

1. If $b_k = 1$: choose $x \leftarrow \{0, 1\}^k$ and run $Q \leftarrow \mathcal{S}_k(1^k, f(s_k \oplus g(x)), \beta), x$.
2. If $b_k = 2$: choose $y \leftarrow \{0, 1\}^k$ and run $Q \leftarrow \mathcal{S}_k(1^k, f(s_k \oplus \beta, g(y))), y$.
3. For each $(a, b) \in Q$: output a if $g(a) = \beta$; output b if $g(b) = \beta$.

By our assumption, Q is a polynomial-length list that, with probability essentially $1/2$, contains a value (a, b) such that $\beta \in \{g(a), g(b)\}$. Thus, we can output the preimage of β with probability essentially $1/2$. Since g is a PRG, and hence a one-way function, we have achieved a contradiction. Thus, the class of PRFs is not 2-hiding autoreducible.

Discussion and Interpretation. The proof considers only PRFs of a certain form. Let f be a fixed, Φ -RKA-secure PRF as above and g a fixed, length-doubling injective PRG. Then define $\widehat{\mathcal{C}}_{\text{PRF}}^{f,g} = \{f_s \mid s \in \{0, 1\}^*\}$, where $f_s(x, y) = f(s \oplus g(x), g(y))$. More precisely, we have shown that $\widehat{\mathcal{C}}_{\text{PRF}}^{f,g}$ ($\subseteq \mathcal{C}_{\text{PRF}}$) is not 2-hiding autoreducible.

Admittedly, the class $\widehat{\mathcal{C}}_{\text{PRF}}^{f,g}$ is not the most natural subclass of PRFs. The result shown here leaves open the possibility that some other class $\mathcal{C} \subseteq \mathcal{C}_{\text{PRF}}$ of PRFs is 2-hiding autoreducible (and not in a trivial sense, as when $|\mathcal{C}| < \infty$). For instance, let F^g denote the well-known GGM construction [18] applied to a PRG g . Is the class $\mathcal{C}_{\text{GGM}} = \{F^g \mid g \text{ is a PRG}\} \subseteq \mathcal{C}_{\text{PRF}}$ 2-hiding autoreducible?³

5 Results for Malicious Security

In this section we describe two constructions of FBB MPC protocols that achieve security against malicious adversaries.

Autoreducibility Criterion. Our first construction is similar in spirit to the one given in Section 4. Like that construction, it is based on a variant of autoreducibility.

Definition 5. Let \mathcal{C} denote a class of functions of two inputs. Let M be a PPT oracle machine and define $\mathcal{Q}[M, f; z]$ to be the random variable containing the sequence of oracle queries made during the computation $M^f(z)$. We say that \mathcal{C} is **1-hiding autoreducible** if there exists a PPT oracle machine M such that for all $f \in \mathcal{C}$:

1. For all x, y , we have that $\Pr[M^f(1^k, x, y) = f(x, y)]$ is overwhelming in k .
2. There exist PPT machine $\mathcal{S}_{f,1}$ and $\mathcal{S}_{f,2}$ such that for all x, y , the ensembles $\{\mathcal{S}_{f,1}(1^k, f(x, y), y)\}_k$, $\{\mathcal{S}_{f,2}(1^k, f(x, y), y)\}_k$, and $\{\mathcal{Q}[M, f; 1^k, x, y]\}_k$ are indistinguishable in k .

³ An alternative way to frame the question is as follows: define \mathcal{F}_{GGM} to be the oracle functionality which takes input x from Alice, y from Bob, and evaluates the GGM construction on seed x , input y , and taking the oracle g as the underlying PRG. Let \mathcal{C}_{PRG} denote the class of all PRGs. Is an FBB secure protocol possible for $\mathcal{F}_{\text{GGM}}^{\mathcal{C}_{\text{PRG}}}$?

In other words, M is able to use oracle access to f to determine $f(x, y)$, yet its oracle queries to f are “independent” of x and of y in some sense. A special case of Definition 5 is when the M ’s oracle queries are distributed uniformly (analogous to the definition of random self-reducibility, except defined with respect to a class of functions).

Theorem 2. *If \mathcal{C} is 1-hiding autoreducible, then $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ has a FBB, UC-secure (i.e., against malicious adversaries) protocol in the \mathcal{F}_{OT} -hybrid model.*

Proof (Proof sketch). The construction is quite similar to the one in the proof of Theorem 1. Both parties access an ideal functionality \mathcal{F}_M which carries out an execution of the machine M from the definition of 1-hiding autoreducibility. In this setting, however, \mathcal{F}_M gives output to *both* parties whenever M makes an oracle query. It then waits for responses from both Alice and Bob, and aborts if the two parties give different responses. Otherwise, it continues its simulation of M , using the parties’ response as the oracle response. As before, when the simulation of M finishes, \mathcal{F}_M gives the output to both parties.

The simulator for a corrupt Alice in the f -instantiated protocol is as follows. When Alice gives input (INIT, x) to \mathcal{F}_M , the simulator sends x to the ideal functionality and receives output $z = f(x, y)$. Then the simulator runs $(q_1, \dots, q_t) \leftarrow \mathcal{S}_{f,1}(1^k, z, x)$. For $i \in [t]$, the simulator gives output (QUERY, q_i) to Alice on behalf of \mathcal{F}_M , and aborts if Alice responds with anything but $(\text{RESP}, f(q_i))$. Recall that the simulator can depend arbitrarily on f and can therefore compute and verify $f(q_i)$. Finally, the simulator gives (OUT, z) to Alice and delivers the output in the ideal model. Soundness of this simulation follows from the definition of 1-hiding autoreducibility. The simulation for corrupt Bob is analogous.

A GMW-style Protocol Compiler. We describe another way to construct malicious-secure FBB protocols, similar in spirit to the well-known GMW compiler [19]. We use zero-knowledge proofs to “compile” a semi-honest-secure protocol into a malicious-secure one, preserving the relevant FBB property.

Theorem 3. *Let \mathcal{F} be an ideal functionality implemented as an oracle machine. If there exists a FBB protocol for $\mathcal{F}^{\mathcal{C}}$ secure against semi-honest adversaries in the \mathcal{F}_{OT} -hybrid model and a FBB protocol for $\mathcal{F}_{\text{FZK}}^{\mathcal{C}}$ (Section 3) that is UC-secure in the \mathcal{F}_{OT} -hybrid model, then there is an FBB protocol for $\mathcal{F}^{\mathcal{C}}$ that is UC-secure in the \mathcal{F}_{OT} -hybrid model.*

In other words, malicious security for the (possibly simpler) function $\mathcal{F}_{\text{FZK}}^{\mathcal{C}}$ can be leveraged to yield malicious security for $\mathcal{F}^{\mathcal{C}}$, while preserving the \mathcal{C} -FBB property.

Proof. The basic approach is to leverage the \mathcal{F}_{FZK} functionality to ensure consistency of each party’s oracle responses in the $\mathcal{F}^{\mathcal{C}}$ protocol. We first augment the $\mathcal{F}^{\mathcal{C}}$ protocol so that whenever one party is asked to make local oracle query q , the other party obtains a *commitment* of q . Then we augment $\mathcal{F}_{\text{FZK}}^{\mathcal{C}}$ to check

that the prover indeed provided the value q underlying the commitment, and also give the verifier a commitment to $f(q)$. Then both parties can return the commitment to $f(q)$ to the \mathcal{F}^C protocol to ensure that the query was answered consistently.

Let Com denote a statistically binding commitment scheme with non-interactive reveal phase. Let (π_A, π_B) be the FBB protocol for \mathcal{F}^C . Define the ideal functionality $\tilde{\mathcal{F}}_\pi$ to do the following:

1. Simulate π_A , π_B , and \mathcal{F}_{OT} in the natural way, with inputs of Alice being fed into π_A , outputs of π_A being given to Alice, and likewise for Bob with π_B .
2. Whenever π_A makes oracle query q , honestly generate a commitment $c \leftarrow \text{Com}(q)$ with decommitment value σ . Give $(\text{QUERY}, c, q, \sigma)$ to Alice and (QUERY, c) to Bob, and wait.
3. Upon receiving inputs $(\text{RESP}, c^*, \sigma^*)$ from Alice and (RESP, c^*) from Bob, ensure that π_A is currently waiting for an oracle response and that σ^* is a valid decommitment of c^* — say, to the value r^* . If so, then continue the simulation taking r^* as the oracle response for π_A .

The functionality has analogous behavior for Bob with respect to π_B . Let (ρ_P, ρ_V) be the FBB protocol for $\mathcal{F}_{\text{FZK}}^C$. Define the ideal functionality $\tilde{\mathcal{F}}_\rho$ to do the following:

1. Expect common input c . On input (INIT, σ) from the prover, ensure that σ is a valid decommitment of c , say, to the value q^* .
2. Simulate instances of ρ_P , ρ_V , and \mathcal{F}_{OT} in the natural way with input q^* to the sender.
3. Whenever ρ_P makes an oracle query q , give (QUERY, q) to prover, expect (RESP, r) from the prover, and continue simulating the components with r as the oracle answer for ρ_P .
4. When ρ_V generates output r^* for the verifier, honestly generate a commitment $c^* = \text{Com}(r^*)$ with decommitment value σ^* . Give $(\text{OUT}, c^*, \sigma^*)$ to the prover, and (OUT, c^*) to verifier.

Let $\tilde{\mathcal{F}}_{\text{FZK}}^C$ denote the functionality which does the following:

1. Expect common input c . On input σ from the prover, ensure that σ is a valid decommitment of c , say, to the value q^* .
2. Query the oracle $f \in \mathcal{C}$ on input q^* to obtain $r^* = f(q^*)$.
3. Honestly generate a commitment $c^* = \text{Com}(r^*)$ with decommitment value σ^* . Give (c^*, σ^*) to the prover, and c^* to verifier.

Then the protocol in the $\tilde{\mathcal{F}}_\rho$ -hybrid model (thus the \mathcal{F}_{OT} -hybrid model) which answers (QUERY, q) messages using the local oracle is a UC-secure, FBB protocol for $\tilde{\mathcal{F}}_{\text{FZK}}^C$.

Now the UC-secure FBB protocol for \mathcal{F}^C is as follows. Parties give their initial inputs to $\tilde{\mathcal{F}}_\pi$ and report outputs from $\tilde{\mathcal{F}}_\pi$. Say that $\tilde{\mathcal{F}}_\pi$ gives $(\text{QUERY}, c, q, \sigma)$ to Alice and (QUERY, c) to Bob. Then the parties invoke $\tilde{\mathcal{F}}_{\text{FZK}}^C$ on common input

c , with Alice acting as the prover providing input σ . Alice obtains (c^*, σ^*) and Bob obtains c^* . Then Alice gives $(\text{RESP}, c^*, \sigma^*)$ and Bob gives (RESP, c^*) to $\tilde{\mathcal{F}}_\pi$. Finally, when $\tilde{\mathcal{F}}_\pi$ gives output (from π_A or π_B), the appropriate party reports it as their own output.

The security of this protocol follows from the binding and hiding properties of the commitment scheme, as well as the security properties of ρ and π . Proof is deferred to the full version.

6 FBB Zero-Knowledge Proofs (and Relaxations)

If f is a deterministic function, then $\mathcal{F}_{\text{FZK}}^f$ (defined in Section 3) is essentially a zero-knowledge proof functionality for the preimage relation $R_f(x, w) = 1 \Leftrightarrow f(w) = x$.

Theorem 4. Define $\mathcal{C}_{\text{OWF}} = \{f \mid f \text{ is a OWF}\}$. If injective one-way functions exist, then there is no standalone-secure, FBB, honest-verifier witness-hiding protocol for $\mathcal{F}_{\text{FZK}}^{\mathcal{C}_{\text{OWF}}}$, even in the presence of an arbitrary trusted setup.

We emphasize that only plain standalone-secure security is required, and that the protocol is not assumed to be a proof (argument) of knowledge. In particular, the above theorem rules out essentially any interesting relaxation of zero-knowledge proofs for the class of relations in question.

Proof. Suppose such a protocol exists, and call its algorithms (P^f, V^f) . When the statement “ $\exists w : R_f(x, w)$ ” is being proven, the honest prover runs $P^f(1^n, w)$ and the honest verifier runs $V^f(1^n, x)$.

Let f be an injective, length-increasing OWF f . In an interaction involving an honest verifier with input x , let \mathcal{E} denote the event that his view contains a query to the oracle on a preimage of x . If $\Pr_w[\mathcal{E} \mid P^f(1^n, w) \rightleftharpoons V^f(1^n, f(w))]$ is non-negligible in n (over choice of random $w \leftarrow \{0, 1\}^n$), then we contradict the (honest verifier) witness-hiding property of the protocol.

So assume that this probability is negligible. Furthermore, in the interaction $P^f(1^n, w) \rightleftharpoons V^f(1^n, f(w))$, the verifier accepts with overwhelming probability due to the completeness property.

Then by an averaging argument, there exists a negligible function δ and strings w_1, w_2, \dots , with $|w_k| = k$, such that $\Pr[\mathcal{E} \mid P^f(1^n, w_n) \rightleftharpoons V^f(1^n, f(w_n))] \leq \delta(n)$. Now for each n , let z_n be a string not in the image of $f(\{0, 1\}^n)$, since f is length-increasing. Define:

$$f'_n(s) = \begin{cases} z_n & \text{if } s = w_n \\ f(s) & \text{else} \end{cases}.$$

Now f'_n is also a OWF and hence the security properties of the $\langle P, V \rangle$ protocol hold when instantiated with f'_n . Note that $R_f(f(w_n), w_n) = 1$ but $R_{f'_n}(f(w_n), w_n) = 0$, since f is injective.

Conditioned on V *not* querying its oracle on input w_n (an event which we assume happens with negligible probability), the outcomes $P^f(1^n, w_n) \rightleftarrows V^f(1^n, f(w_n))$ and $P^{f'}(1^n, w_n) \rightleftarrows V^{f'_n}(1^n, f(w_n))$ are identical. That is, the verifier accepts with overwhelming probability when instantiated with either f or f'_n . Then when the prover runs the honest protocol with oracle f and input w_n , it constitutes a violation of soundness against an honest verifier instantiated with f'_n . More specifically, in such an interaction the honest verifier is instantiated with oracle f'_n yet accepts a statement about $R_{f'_n}$ that is false.

References

1. Abadi, M., Feigenbaum, J., Kilian, J.: On hiding information from an oracle. *J. Comput. Syst. Sci.* 39(1), 21–50 (1989)
2. Allender, E.: New surprises from self-reducibility. In: Ferreira, F., Löwe, B., Mayordomo, E., Gomes, L.M. (eds.) CiE 2010, Abstract and handout booklet, pp. 1–5 (2010)
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: Ostrovsky, R. (ed.) FOCS, pp. 120–129. IEEE (2011)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (Im)possibility of Obfuscating Programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
5. Beaver, D., Feigenbaum, J.: Hiding Instances in Multioracle Queries. In: Choffrut, C., Lengauer, T. (eds.) STACS 1990. LNCS, vol. 415, pp. 37–48. Springer, Heidelberg (1990)
6. Beaver, D., Feigenbaum, J., Kilian, J., Rogaway, P.: Locally random reductions: Improvements and applications. *J. Cryptology* 10(1), 17–36 (1997)
7. Bellare, M., Cash, D.: Pseudorandom Functions and Permutations Provably Secure against Related-Key Attacks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 666–684. Springer, Heidelberg (2010)
8. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
9. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10. ACM (1988)
10. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003)
11. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
12. Brassard, G.: Cryptology - column 4: hiding information from oracles. *SIGACT News* 21(2), 5 (1990)
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. *Electronic Colloquium on Computational Complexity (ECCC)* TR01-016 (2001); Previous version “A unified framework for analyzing security of protocols” available at the ECCC archive TR01-016. Extended abstract in FOCS 2001 (2001)

14. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503. ACM (2002)
15. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO, pp. 199–203. Plenum Press, New York (1982)
16. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19. ACM (1988)
17. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
18. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
19. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229. ACM (1987)
20. Haitner, I., Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions of protocols for secure computation. SIAM J. Comput. 40(2), 225–266 (2011)
21. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC, pp. 44–61. ACM (1989)
22. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
23. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31. ACM (1988)
24. Kolesnikov, V., Schneider, T.: Improved Garbled Circuit: Free XOR Gates and Applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórssón, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
25. Lindell, Y.: Techniques for efficient secure two-party computation with malicious adversaries. Technical talk, The Check Point Institute Crypto and Security Day (2010)
26. Pass, R., Wee, H.: Black-Box Constructions of Two-Party Protocols from One-Way Functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009)
27. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation Is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
28. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of Reducibility between Cryptographic Primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
29. Selke, J.: Autoreducibility and friends: About measuring redundancy in sets. Master’s thesis, Gottfried-Wilhelm-Leibniz-Universität, Hannover (2008)
30. Yao, A.C.: How to generate and exchange secrets. In: FOCS, pp. 162–167. IEEE (1986)

Adaptively Secure Multi-Party Computation with Dishonest Majority

Sanjam Garg and Amit Sahai

University of California, Los Angeles, USA
`{sanjamg,sahai}@cs.ucla.edu`

Abstract. Adaptively secure multiparty computation is an essential and fundamental notion in cryptography. In this work we focus on the basic question of constructing a multiparty computation protocol secure against a *malicious, adaptive* adversary in the *stand-alone* setting without assuming an honest majority, in the plain model. It has been believed that this question can be resolved by composing known protocols from the literature. We show that in fact, this belief is fundamentally mistaken. In particular, we show:

- **Round inefficiency is unavoidable when using black-box simulation:** There does not exist any $o(\frac{n}{\log n})$ round protocol that adaptively securely realizes a (natural) n -party functionality with a black-box simulator. Note that most previously known protocols in the adaptive security setting relied on black-box simulators.
- **A constant round protocol using non-black-box simulation:** We construct a *constant round* adaptively secure multiparty computation protocol in a setting without *honest majority* that makes crucial use of non-black box techniques.

Taken together, these results give the first resolution to the question of adaptively secure multiparty computation protocols with a malicious dishonest majority in the plain model, open since the first formal treatment of adaptive security for multiparty computation in 1996.

1 Introduction

The notion of *secure computation* is central to cryptography. Introduced in the seminal works of [1, 2], secure multi-party computation (MPC) allows a group of (mutually) distrustful parties P_1, \dots, P_n , with private inputs x_1, \dots, x_n , to jointly compute any functionality f in such a manner that the honest parties obtain correct outputs and no group of malicious parties learns anything beyond their inputs and prescribed outputs. In this setting we can consider an adversary that can *adaptively* corrupt parties throughout the protocol execution depending on its view during the execution. Adaptively secure multiparty computation is an essential and fundamental notion in cryptography. We refer the reader to ([3], Section 1) for further discussion on the importance of considering adaptive adversaries.

Canetti, Feige, Goldreich and Naor [3] constructed the first adaptively secure MPC protocol in the standalone setting, assuming the presence of an honest

majority. Beaver constructed an adaptively secure zero-knowledge protocol [4] (see also [5]) and an adaptively secure OT protocol [6]. Similar results for general *two-party* computation were established in [7, 8]. Assuming a trusted common random string (CRS), Canetti, Lindell, Ostrovsky and Sahai [9] gave the first adaptively secure MPC protocol without honest majority in the two-party and the multi-party setting, in fact under an even strong notion of security called the UC security (which can be achieved only with a trusted setup). In this paper, we focus on the following *basic* question:

Is it possible to construct multiparty computation protocols in the standalone setting (without any trusted setup) secure against a malicious, adaptive adversary that may corrupt any number of parties?

Previous Work on This Question: Choi, Dachman-Soled, Malkin and Wee [10, 11] give a construction of an adaptively secure multi-party computation protocol when given access to an ideal commitment (more formally, in the commitment hybrid model). At the same time, many adaptively secure protocols for securely realizing the commitment functionality (e.g. [12]) are known. And we know that it is possible to compose protocols by the composition theorem of Canetti [13], which holds in the adaptive security setting.

Surprisingly, however, it turns out that a straightforward application of these results does not (in fact as we argue, it **cannot**) achieve adaptive security in the multiparty setting!¹ We stress that all the results stated in the previous paragraph (with proper formalization) are correct, and yet *still* the conclusion does not follow.

Adaptively Secure Composition: More than Meets the Eye. Somewhat surprisingly, a 2-party adaptively secure protocol *fails* to guarantee security when executed in the setting of n -parties, even if only two of the parties are *ever* talking to each other. (Thus, the 2-party adaptively secure commitment protocol of [12] is not necessarily adaptively secure in the n -party setting.) Indeed Canetti [13] (Theorem 10, Page 38) requires that in order to obtain an n -party adaptively secure protocol via the composition theorem, all the protocols being composed must be secure in the n -party setting to begin with. Nevertheless, this might seem surprising, and we demonstrate this issue by considering an example. We know that the vast majority of the simulators for MPC protocols are *black-box*. Now, consider an adaptively secure protocol in the 2-party case with a black-box simulator. Suppose that this 2-party protocol is executed in the setting of n parties out of which only two of them communicate. The black-box simulator for the 2-party protocol must rely on “rewinding” for the proof of security. However,

¹ Indeed, this composition seemed so “obvious” that in [11], Corollaries 2 and 3 claimed a result for adaptively secure multi-party computation in the plain model, and were given without proof. After seeing our work, the authors of [11] have corrected their paper to only refer to the two-party case in their corollaries. We stress that the corollaries of [11] do apply to the two-party setting, and that nothing in this paper should be taken to imply that any of the proofs given in [11] are incorrect.

in the adaptive n -party setting an adversary can also corrupt parties that *do not communicate* during the execution of the protocol. What if this happens during a rewinding? This case is never handled in the simulation for the 2-party case, and thus the proof of composition security breaks down. Indeed this seemingly small issue turns out to be a fundamental barrier to constructing adaptively secure MPC protocols. Not only does the proof break down, but as we show below, there are explicit impossibility results possible in the black-box setting. Thus, we show that in the setting without honest majority, we need to completely rethink the techniques used to construct adaptively secure multi-party computation protocols.

1.1 Our Results

We consider an asynchronous multi-party network² where the communication is open (i.e. all the communication between the parties is seen by the adversary) and delivery of messages is not guaranteed. (For simplicity, we assume that delivered messages are authenticated. This can be achieved using standard methods.) The two main results of the paper are:

Round inefficiency with a black-box simulation: There does not exist any $o(\frac{n}{\log n})$ round protocol that adaptively securely realizes a natural n -party functionality (more specifically an extension of the commitment functionality to the setting of n parties) with a black-box simulator. This result holds in the standalone setting in the plain model. We stress that all protocols that deal with adaptive security in the standalone model that we are aware of employ a black-box simulator. This implies that the techniques previously used to build adaptively secure multiparty protocols must incur a substantial efficiency loss.

A round efficient protocol with non-black box simulation: On the other hand, we show that non-black-box techniques can be used to circumvent the above described impossibility result. Assuming collision resistant hash functions, trapdoor permutations, augmented non-committing encryption [3, 9] and dense cryptosystems [14] we construct a *constant round* adaptively secure n -party protocol where the adversary is allowed to corrupt up to $n - 1$ parties in the *non-erasure* model. If security against corruption of all n parties is desired then our construction yields a protocol with round complexity that is proportional to the depth of the circuit being evaluated. Alternatively, in the setting where all n parties can be corrupted, we can get a constant-round protocol if data *erasures* are allowed. This result shows a new area where non-black-box techniques can allow us to overcome round efficiency barriers that would otherwise exist.

² The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

Discussion: The negative result leaves open the question of constructing adaptively secure protocols with black-box simulation, *but with poor round complexity*. We find this direction not very interesting in light of our positive result constructing round-efficient protocols using non black-box techniques. Nevertheless, we provide a sketch of a round-inefficient black-box construction in the full version, which is nearly tight with respect to our impossibility result.

On Erasures: Our positive results include round efficient protocols both in the setting of *erasures* and *non-erasures*. On the other hand our negative result holds even when parties are allowed to erase everything except their input. (Note that for our positive result with erasures, honest parties are certainly not required to erase their inputs.) From the earliest works on adaptive security [15] with erasures, it has been a major design goal to reduce the amount of erasures necessary. We refer the reader to ([13], Section 5.2) for a more general discussion on how trusted erasures may be a problematic assumption. Nevertheless, in light of our negative result we may also want to consider protocols that allow honest parties to erase their inputs during the protocol. We argue that it is particularly unreasonable to consider such protocols: Consider, for example, a setting where many hospitals are collaborating on some research involving their patient records. In order to do this research, they execute an MPC protocol, where each hospital’s input is its database of patient records. We do not expect any hospital to be willing to erase all of its patient records (even from backup facilities, as backup facilities could also be adaptively corrupted), even temporarily, just to enable an MPC computation for research purposes. Worse, any protocol in the dishonest majority setting that requires honest parties to erase its inputs would enable an adversary, simply by aborting the protocol, to cause all honest parties to lose all of their inputs *forever*. While the example of hospital patient records underscore how undesirable erasure of inputs can be, this issue would be quite problematic in most contexts. Thus, we do not consider protocols where inputs can be erased³. Recall, however, that we can achieve round-efficient adaptive security without requiring erasures at all using non-black-box techniques.

1.2 Our Techniques

The central idea for our impossibility result is to argue that a black-box simulator of an $o(\frac{n}{\log n})$ round protocol for an n -party functionality does not gain anything via “rewindings” in the adaptive setting. Now we elaborate on this. Consider an r round (such that r is $o(\frac{n}{\log n})$) protocol for an n -party functionality with a black-box simulator. Consider an adversary that behaves completely honestly in

³ It is not hard to see that if we were to allow erasure of inputs, then the following solution would be possible: The parties first non-malleably secret share their inputs among all parties. Subsequently, all parties erase everything except their own share (and the shares they receive from other parties). Finally, they run the protocol on the shares as inputs instead. However, we again stress that we find this solution highly unsatisfactory in light of the discussion above.

the protocol itself, however, after each round of the protocol it corrupts roughly $\omega(\log n)$ parties. Furthermore, the parties to be corrupted are chosen randomly (in fact pseudo-randomly based on the protocol messages so far) among the uncorrupted parties so far. On corruption of an honest party, the simulator is obliged to provide to the adversary the input of the party just corrupted. In its “main thread” execution with the adversary, to help the simulator in simulation, the simulator is also provided with these inputs. However, every time the simulator “rewinds” the adversary, the adversary will (with overwhelming probability) choose to corrupt at least one party that is not among the ones corrupted in the main thread. The simulator therefore will be unable to proceed in any “rewinding.” Note that the only additional power awarded to a *black-box* simulator is essentially the ability to “rewind” the adversary which is essentially useless in our context. We therefore conclude that no such simulator can exist.

As is clear from the impossibility result just described, the problem of round inefficiency will be inherent to any simulator that “rewinds.” In order to get around this problem, we turn to the non black-box simulation technique of Barak [16]. However, Barak’s protocols are far from being adaptively secure. To achieve adaptive security, we adapt and make use of a technique developed in the context of concurrently secure computation [17–19].

Technical Overview for the Construction of Our Constant Round Protocol: Now we give a detailed technical overview of our construction. We will start by giving a high level idea of the final protocol and then delving into the details of sub-protocol (along with specifics of constructions) that need to be built. Throughout the following description, we advise the reader to keep in mind that our goal is to construct a round efficient protocol and as is clear from the negative result stated above this cannot be done with a simulator that “rewinds.” Therefore we will restrict ourselves to a “straight-line” or a “non-rewinding” simulator.

- **Reducing the problem of adaptively secure MPC to generation of common random strings.** The starting point of our construction is the observation that an adaptively secure MPC protocol (Theorem 3, [20])⁴ for any functionality can be realized in OT-hybrid (oblivious transfer) model. Note that in this construction each OT call is made between two parties. Further note that for an OT call between two parties security is required only if at least one of the two parties is honest. Additionally, note that we can adaptively securely realize OT functionality in the CRS hybrid (common random string) model (e.g., using [9]). Therefore in order to construct an adaptively secure protocol it suffices for us to adaptively securely realize the CRS functionality between every pair of parties where the CRS generated by a pair of parties is required to be honestly generated only if at least one of the two parties is honest.
- **Generating a common random string between a pair of parties.** Now we are left with the goal of adaptively securely realizing CRS between every pair of parties. We start by giving intuition for a protocol that adaptively

⁴ We refer the reader to Remark 2 of [20] for discussion on variants of this result.

securely realizes CRS between two parties and then sketch the extension to the **setting of multiple parties**. We do this by constructing a **coin flipping protocol secure in the adaptive setting** in which the simulator can simulate in a straight-line manner. In order to construct such a coin flipping protocol our simulator will need the ability to *equivocate* on its commitments. In other words, we will need that our simulator can open its commitments to any value of its choice even after it has made those commitments. Looking ahead the simulator will also need the ability to *extract* (the reasons for which we see later) from the commitments made by the adversary. More specifically, we will need that the simulator can extract the values committed to by the adversary. Next we will first describe a mechanism that allows a straight-line simulator to equivocate on its commitments in the adaptive setting. Subsequently, we will see how equivocation can be used in setting up coin flipping (and the need of extractability in the process).

- **Equivocal commitment scheme in the adaptive setting.** We consider the *public-coin* zero-knowledge protocol⁵ of Barak [16]. Even though this protocol is secure against adaptive corruptions of the verifier, it is far from being adaptively secure if we were to consider adaptive corruption of the prover. We will modify Barak's protocol as follows. For every bit sent by the prover in the Barak's protocol, we will require that our prover instead sends a random string of appropriate (length of a pseudorandom bit commitment) length. Note that in this modified protocol no actual proof is given. Furthermore, all the messages sent by an honest prover and an honest verifier in this modified protocol are just random bits and thus adaptive corruption of parties participating in an execution of this modified protocol does not help the adversary in any way. However, a key idea is that we can define an NP-relation that accepts a transcript if only if there exist decommitments of the prover messages such the decommitted prover messages along with the verifier messages form an accepting transcript of an execution of the Barak's protocol. Roughly speaking our modified protocol has two properties, with respect to this NP-relation:

- No adaptively corrupted cheating prover interacting with an honest verifier in our modified protocol can output a witness for the transcript generated.
- Consider any execution in which the prover is honest. In this execution our simulator (simulating the prover) can internally use the simulator of Barak's protocol and always output a witness for the transcript generated.

We can reduce this transcript to a graph (can be constructed using an NP-reduction) that is Hamiltonian if and only if there exists a witness corresponding to the above NP-relation. Furthermore, given the witness we can also deduce the Hamiltonian cycle in the obtained graph. This graph can now be used to generate commitments such that a party with access to a

⁵ In a public-coin zero-knowledge protocol all messages of the verifier correspond to random bits (“coin flips”).

cycle in the graph can open them in any way. We refer the reader to Section 3 for more details on this.

Note that an execution of the modified Barak's protocol guarantees equivocability of commitments sent on behalf of only one of the two parties. Therefore we will have to set up *two* equivocal commitments. This can be easily achieved by execution modified Barak's protocol twice between the two parties switching the roles the two parties play in the two executions of the modified Barak's protocol.

- **Coin flipping protocol secure in the adaptive setting.** Next using equivocal commitments, we construct a coin flipping protocol between two parties A and B . One standard approach for constructing such a coin flipping protocol is to have the two-parties commit to random strings (via equivocal commitments) which they subsequently open one by one. The output of the protocol corresponds to the exclusive or of the two strings. Lets consider the case in which A opens first. The key technical problem that arises in this case is that if B is corrupted then the straight-line simulator (simulating A) without knowledge of the value committed to by B will not be able to force the output of the protocol to a value of its choice.

We solve this problem by doing two coin flips both of which roughly follow the same outline as above. The first coin flipping is done in-order to setup a public key of a public key encryption scheme (with pseudorandom public-keys and pseudorandom ciphertexts). In this protocol we require that B opens first and this allows the simulator to force the output of the protocol to a value of its choice (in a straight-line manner) as long as A is honest. Subsequently the parties execute a second coin flipping protocol in which we require that B (B opens later now), in addition to the commitment it sends, is required to send encryption of the randomness used in generating the commitment using the public key generated in the first coin flipping. This allows the simulator to extract the value committed by B (if B is corrupted) even before A needs to open its committed value and thereby allowing it to simulate in a straight line manner. However, in case B is honest then the simulator will have to explain the encryptions as if they were honestly generated. We achieve this in a way similar to [9].

- **Setting up multiple common random strings.** Additionally other well known issues relating to *non-malleability* arise in constructing of constant round protocols [21] because of the need to execute different protocol instances in *parallel*. We deal with issue using the *two-slot technique* of [17]. Concretely we consider Pass' [22] family of non-black-box zero knowledge protocols with strong *simulation soundness* properties, i.e., any one of these protocols continues to remain *sound* even when all the other protocols in the family are being simulated. We prove that modifying these protocols just like we modified the Barak's protocol above suffices for our purposes.

Roadmap. We start by providing our impossibility result for black-box simulation in Section 2. Next we recall some very basic notions and setup notation

in Section 3. Finally we provide the construction of our constant round protocol in Section 5 using sub-protocols constructed in Section 4.

2 Round Inefficiency with a Black-Box Simulation Is Unavoidable

In this section, we show the existence of a deterministic n -party functionality for which there does not exist any $o(\frac{n}{\log n})$ round adaptively secure protocol, with a black-box simulator.

Before proceeding to the formal proof, we first give some intuition behind our impossibility result. The central idea to our proof is to argue that a black-box simulator (say) \mathcal{S} of an $o(\frac{n}{\log n})$ round protocol does not gain any thing via “rewindings” in the adaptive setting. Informally speaking, this means that the simulator fails to get any useful information from any look-ahead thread and even in this setting it must still be able to extract the adversary’s input. However, a simulator must have some additional power over a real adversary, and the only additional power awarded to a *black-box* simulator is essentially the ability to rewind the adversary. We therefore conclude that black-box simulators cannot exist for any $o(\frac{n}{\log n})$ round protocol, as stated in Theorem 1 below.

Theorem 1. *There exist a deterministic n -party functionality for which there does not (assuming one way functions) exist any $o(\frac{n}{\log n})$ round adaptively secure protocol with respect to black-box simulators.*

Proof. We will organize our proof into two main parts.

1. First, we consider the commitment functionality F , where there are two special parties – the *committer* C and the *receiver* R , and $n - 2$ *dummy* parties. Let Π be any $o(\frac{n}{\log n})$ -round n -party protocol that adaptively securely realizes F with respect to a black-box simulator. Then, for large enough n , we first construct an adversary \mathcal{A} for Π , that corrupts C , such that *every* black-box simulator \mathcal{S} for Π gets full participation from the adversary in the “main thread,” but fails to get any “useful” information from the rewinds. Our adversary \mathcal{A} , has the inputs of dummy parties *hard-coded* inside itself and it acts in the following way. It starts by corrupting the committer C . It follows the honest committer strategy on behalf of C , except that after each round of Π it corrupts roughly $\omega(\log n)$ parties. Furthermore, the parties to be corrupted are chosen randomly (in fact pseudo-randomly based on the protocol messages so far) among the uncorrupted parties so far. On corruption of an honest party, the simulator is obliged to provide to the adversary the input of the party just corrupted. In its “main thread” execution with the adversary, to help the simulator in simulation, the simulator is also provided with these inputs. However, every time the simulator “rewinds” the adversary, the adversary will (with overwhelming probability) choose to corrupt at least one party that is not among the ones corrupted in the main

- thread. The simulator therefore will be unable to proceed in any “rewinding.” However, by security of the protocol such a simulator must still be able to extract the input of C . Proving this is in fact the crux of our proof.
2. Next, we consider *another* real-life adversary \mathcal{A}' , that corrupts all parties except C , uses the black-box simulator \mathcal{S} (constructed above) and actually succeeds in extracting the input of the honest *committer*. This contradicts the assumption that Π securely realizes F .

Combining the two parts, we conclude that for the n -party commitment functionality F (as described above), there does not exist any $o(\frac{n}{\log n})$ -round protocol that adaptively securely realizes F with respect to black-box simulators. We note that this is sufficient to prove Theorem 1. We give more details on both parts of the proof in the full-version.

3 Building Blocks for Our Constant Round Protocol

In this section we recall and define some very basic notions and setup notation. Let k denote a security parameter. We say that a function is *negligible* in the security parameter k if it is asymptotically smaller than the inverse of any fixed polynomial. Otherwise, the function is said to be *non-negligible* in k . We say that an event happens with *overwhelming* probability if it happens with a probability $p(k) = 1 - \nu(k)$ where $\nu(k)$ is a negligible function of k . In this section, we recall the definitions of basic primitives studied in this paper. We now discuss the main cryptographic primitives that we use in our construction.

Underlying Standard Commitment. The basic underlying commitment scheme Com is the standard non-interactive commitment scheme based on a one-way permutation f and a hard-core predicate b of f . That is, in order to commit to a bit σ , one computes $\text{Com}(\sigma) = \langle f(U_k), b(U_k) \oplus \sigma \rangle$, where U_k is the uniform distribution over $\{0, 1\}^k$. Note that Com is computationally secret, and produces pseudorandom commitments: that is, the distributions $\text{Com}(0)$, $\text{Com}(1)$, and U_{k+1} are computationally indistinguishable. Let the length of the commitment, for one bit message, generated by the pseudorandom commitment scheme be $\ell_C(k)$ ($k + 1$ in the above case). For simplicity of exposition, in the sequel, unless necessary, we will assume that random coins are an implicit input to the commitment function. Furthermore, we will sometimes abuse notation and use the same notation to generate commitments to strings, which can be thought of as a concatenation of commitments of individual bits.

The Modifier Graph Based Commitment Scheme IDCom_G . We use the notation of [19] and some of the text has been taken verbatim from there [19].

To commit to a 0, the sender picks a random permutation π of the nodes of G , and commits to the entries of the adjacency matrix of the permuted graph one by one, using Com . The sender also commits (using Com) to the permutation π . These values are sent to the receiver as $c = \text{IDCom}_G(0)$. To decommit, the sender decommits to π and decommits to every entry of the adjacency matrix. The receiver verifies that the graph it received is $\pi(G)$.

To commit to a 1, the sender chooses a randomly labeled q -cycle, and for all the entries in the adjacency matrix corresponding to edges on the q -cycle, it uses Com to commit to 1 values. For all the other entries, including the commitment to the permutation π , it simply produces random values from U_{k+1} (for which it does not know the decommitment!). These values are sent to the receiver as $c = \text{IDCom}_G(1)$. To decommit, the sender opens only the entries corresponding to the randomly chosen q -cycle in the adjacency matrix.

This commitment scheme has the property of being computationally secret, i.e. the distributions $\text{IDCom}_G(0)$ and $\text{IDCom}_G(1)$ are computationally indistinguishable for any graph G . Also, given the opening of any commitment to both a 0 and 1, one can extract a Hamiltonian cycle in G . Finally, as with the scheme of [23], given a Hamiltonian cycle in G , one can generate commitments to 0 and then open those commitments to both 0 and 1.

Furthermore, here if the simulator has knowledge of a Hamiltonian cycle in G , it can also produce a random tape for the sender explaining $c = \text{IDCom}_G(0)$ as a commitment to both 0 and 1. If, upon corruption of the sender, the simulator has to demonstrate that c is a commitment to 0 then all randomness is revealed. To demonstrate that c was generated as a commitment to 1, the simulator opens the commitments to the edges in the q -cycle and claims that all the unopened commitments are merely uniformly chosen strings (rather than commitments to the rest of G). This can be done since commitments produced by the underlying commitment scheme Com are pseudorandom.

In this setting as well, we will sometimes abuse notation and use the same notation to generate commitments to strings. In particular, we will use the notation $c = \text{IDCom}_G(m; r)$ to denote the function that generates a commitment to m using random coins r . Furthermore a commitment $c = \text{IDCom}_G(0^\kappa; r')$ to the zero string of length κ can be explained to any value m using the function $r = \text{IDOpen}_G(m, r', w)$, where w is a Hamiltonian cycle in the graph G .

Dense Cryptosystems. In our construction we will need an encryption scheme that has *pseudo-random public keys*. More specifically, we require that the public key is indistinguishable from a random string. Such an encryption scheme can be constructed from dense cryptosystems [14]. Furthermore, we will require that the scheme has pseudorandom ciphertexts. More formally:

Definition 1 (Encryption with pseudorandom ciphertexts). A public-key cryptosystem (G, E, D) has pseudorandom ciphertexts of length $\ell_E(k)$ if for all non-uniform polynomial time adversaries \mathcal{A} we have

$$\begin{aligned} & \Pr \left[(pk, sk) \leftarrow G(1^k) : \mathcal{A}^{E_{pk}(\cdot)}(pk) = 1 \right] \\ & \approx \Pr \left[(pk, sk) \leftarrow G(1^k) : \mathcal{A}^{R_{pk}(\cdot)}(pk) = 1 \right], \end{aligned} \quad (1)$$

where $R_{pk}(m)$ runs $c \leftarrow \{0, 1\}^{\ell_E(k)}$ and every time returns a fresh c . We require that the cryptosystem has errorless decryption.

Barak's Non-black Box Technique. We use the non black-box simulation technique of Pass [22] (which in turn builds on the work of Barak [16]). Consider

a “special” $\text{NTIME}(T(k))$ relation R_{Sim} as follows.⁶ Let $k \in \mathbb{N}$ and let $T : N \rightarrow N$ be a “nice” function that satisfies $T(k) = k^{\omega(1)}$. Let $\{\mathcal{H}_k\}_{h \in \{0,1\}^k}$ be hash function family where $h \in \mathcal{H}_k$ maps $\{0,1\}^*$ to $\{0,1\}^k$. Let the triple $\langle h, c, r \rangle$ be the input to R_{Sim} . Further, consider a witness that consists of a program Π , a string $y \in \{0,1\}^{|r|-k}$, and string s . Then $R_{Sim}(\langle h, c, r \rangle, \langle \Pi, s, y \rangle) = 1$ if and only if:

1. $c = \text{Com}(h(\Pi); s)$.
2. $\Pi(c, y) = r$ within $T(k)$ steps.

Witness Indistinguishable Universal Argument. The function $T(k)$ corresponding to the above describe relation R_{Sim} is super-polynomial in k . This implies that the language corresponding to R_{Sim} does not lie in NP (but rather in $\text{NTIME}(T(k))$). Such languages are beyond the scope of the “standard” witness indistinguishable proof systems (designed to handle NP-languages only), and will thus require the usage of a Witness Indistinguishable Universal Argument (WI-UARG) [24]. We note that the WI-UARG protocol of Barak and Goldreich [24] is public coin and the running time of the verifier in the protocol is bounded by a fixed polynomial.

4 Sub-protocols Used in Our Constant Round Protocol

In the construction of our final adaptively secure MPC protocol will use a *concurrently secure trapdoor generator protocol* $\langle P, V \rangle$ and a *coin flipping protocol* $\langle A, B \rangle$. In this section we will give a constructions of these protocols. Furthermore, we will prove special properties about these protocols that are useful for us in our final construction.

4.1 Trapdoor Generator Protocol

In this section we describe a *family of trapdoor generator protocols* $\langle P, V \rangle_i$ where $i \in \{1 \dots m\}$. $\langle P, V \rangle_i$ is a two party protocol between P_i and V_i and at the end of the protocol both parties output a Graph (let’s say G). Consider the setting in which one protocol instance of each of the protocols $\langle P, V \rangle_1, \langle P, V \rangle_2 \dots \langle P, V \rangle_m$ is being executed concurrently in between n parties – Q_1, \dots, Q_n with inputs $x_1 \dots x_n$.⁷ We stress that in these protocol executions each Q_i could

⁶ The relation presented is slightly oversimplified and will make Barak’s protocol work only when the hash function family is collision resistant against “slightly” super-polynomial sized circuits [16]. However, this can be modified to work assuming collision resistance against polynomial sized circuits only. It does not affect the analysis in this paper and we refer the reader to [24] for details.

⁷ As we will see later that we only need security in the setting of parallel composition. However, in this section we will stick with the notion of concurrent composition and argue security in this setting. From this it follows immediately that our protocol is also secure in the less demanding setting of parallel composition.

potentially be playing the role of multiple P_j 's and V_k 's where $j, k \in \{1 \dots m\}$. In this setting we consider an adversary \mathcal{A} that adaptively corrupts parties (an honest party reveals its input and random coins to the adversary on corruption).

However, for simplicity of exposition, we will model this instead as a setting of $n + 2m$ parties – Q_1, \dots, Q_n with inputs $x_1 \dots x_n$ and $P_1, \dots, P_m, V_1, \dots, V_m$ with no inputs. Furthermore, parties P_i and V_i execute an instance of the protocol $\langle P, V \rangle_i$. In this setting, we will consider an adversary that adaptively corrupts any of these parties. *We stress that any adversary in the original setting where each Q_i could potentially be playing the role of multiple P_j 's and V_k 's can always be used to construct an adversary in this setting.* This follows from the fact that in the original setting when an adversary corrupts a party Q_i it additionally corrupts multiple P_j 's and V_k 's. Analogously in this setting our adversary can continue to corrupt all the parties playing the roles of Q_i and P_j 's and V_k 's. Throughout the rest of this sub-section we will stick to this setting.

Very informally, assuming collision resistant hash functions, we will require that our protocol satisfies the following security properties:

1. For every such adversary \mathcal{A} that adaptively corrupts parties, there exists a non-black box simulator $\mathcal{S}_{\langle P, V \rangle}$ (that obtains the inputs of parties adaptively corrupted by \mathcal{A}) such that the view of the adversary \mathcal{A} in its interaction with honest parties and the view of the adversary \mathcal{A} in its interaction with $\mathcal{S}_{\langle P, V \rangle}$ are computationally indistinguishable.
2. For every $i \in [m]$ such that P_i is not corrupted, $\mathcal{S}_{\langle P, V \rangle}$ outputs a Hamiltonian cycle in the graph that the execution of $\langle P, V \rangle_i$ yields.
3. For every $i \in [m]$ such that V_i is not corrupted, \mathcal{A} cannot output a Hamiltonian cycle in the graph that parties P_i and V_i executing $\langle P, V \rangle_i$ output. Furthermore, we require that the \mathcal{A} cannot output a Hamiltonian cycle even when for every $i \in [m]$ such that P_i is not corrupted it is additionally provided with the Hamiltonian cycle in the graph that the execution of $\langle P, V \rangle_i$ yields.
4. Finally, since we are in the adaptive setting, on corruption, an honest party (or, the simulator on behalf of the honest party in the simulated setting) must provide its input and random coins to the adversary. We will require that all the above properties hold even when this additional communication happens with the adversary.

Next we build some notation that will be useful in the formal description of our protocol $\langle P, V \rangle_i$ (in Figure 1).

Shadow Version of WI-UARG. Recall that in the WI-UARG protocol V only sends random bits. Finally at the end of the protocol V outputs 1 or 0. We will modify the WI-UARG protocol into what we call the *shadow version of the WI-UARG* protocol. The prover in the shadow protocol, for every bit sent by the prover in the original protocol, sends a random string in $\{0, 1\}^{\ell_C(k)}$ (recall that $\ell_C(k)$ is the length of a pseudorandom commitment). Furthermore, the behavior of the verifier V remains unmodified. We will denote this modified protocol by

Com is a pseudo-random commitment scheme with output from $\{0, 1\}^{\ell_C(k)}$.^a We also use the “shadow version” of the 5-round public-coin WI-UARG protocol which we denote by sWI-UARG. Further let $\mu(k) = (m(k) \cdot 4k^3 + \ell(k) + k)$.

Setup : V_i sends $h \xleftarrow{\$} \mathcal{H}_k$ to P_i .

Slot 1 :

1. P_i sends a random string in $c_1 \xleftarrow{\$} \{0, 1\}^{k \cdot \ell_C(k)}$ to V_i .
2. V_i sends a challenge string $r_1 \xleftarrow{\$} \{0, 1\}^{i\mu(k)}$.

Slot 2 :

1. P_i sends a random string in $c_2 \xleftarrow{\$} \{0, 1\}^{k \cdot \ell_C(k)}$ to V_i .
2. V_i sends a challenge string $r_2 \xleftarrow{\$} \{0, 1\}^{(m+1-i)\mu(k)}$.

Main Proof Stage : P_i and V_i engage in the shadow version of the WI-UARG protocol^b in which P_i proves to V_i the following statement:

- there exists Π, s, y, b such that $R_{Sim}(\langle h, c_b, r_b \rangle, \langle \Pi, s, y \rangle) = 1$.

Output Stage : Let **transcript** be the transcript of the above execution. Let G be a graph (can be constructed using an NP-reduction) that is Hamiltonian if and only if $\exists w$ such that $R_{uarg}(\text{transcript}, w) = 1$. Both parties output G .

^a We use commitments based on one-way permutation just for simplicity of exposition. At the cost of a small complication, the one-message scheme could have been replaced by the Naor’s [25] 2-message commitment scheme, which can be based on “ordinary” one-way functions.

^b As already pointed, we advise the reader to keep in mind that both the honest prover and the honest verifier of the shadow version of the WI-UARG protocol just send random bits and that no real proof in an honest execution is ever given.

Fig. 1. $\langle P, V \rangle_i$ (the i^{th} protocol in the family of $m(k)$ protocols)

sWI-UARG. Consider an instance of execution of the sWI-UARG protocol with transcript **transcript**. Note that this transcript contains messages sent by the prover and the messages sent by the verifier. Further note that every $\ell_C(k)$ bit string sent by the prover could be interpreted (if they really are) as a commitment to a bit using the commitment scheme **Com**. Let w be the de-commitment information (if it exists) corresponding to all the $\ell_C(k)$ bit strings in **transcript** that are sent by the prover. Also let **transcript'** = **unshadow(transcript, w)**⁸ denote the transcript obtained by replacing every $\ell_C(k)$ bit string in **transcript** that is sent by P with the corresponding committed bit (as per **Com**). Let $R_{uarg}(\text{transcript}, w) = 1$ if and only if $V(\text{unshadow}(\text{transcript}, w)) = 1$.

We stress that in the shadow version of the WI-UARG protocol both the honest prover and the honest verifier just send random strings and that *no real proof is actually given*. However, we also consider a modification of the prover strategy of shadow version of the WI-UARG protocol called the *simulated shadow prover*. The simulated shadow prover additionally obtains a witness for

⁸ Note that the function **unshadow** is inefficient and is used just to define the NP-Relation.

the statement being proven and corresponding to every bit sent by the prover in the original WI-UARG protocol instead sends a commitment to that bit using the Com commitment scheme. Note that transcript transcript generated when the prover follows the simulated shadow prover strategy is such that there exists w such that $R_{uarg}(\text{transcript}, w) = 1$. Finally, note that the messages generated by a simulated shadow prover are computationally indistinguishable from the messages generated by an honest prover of the shadow version of the WI-UARG protocol. We will use this shadow prover strategy in our proof.

Common Parameters. All parties receive two parameters $m(k)$ and $\ell(k)$ as input. $m(k)$ corresponds to the size of the family of $\langle P, V \rangle$ protocols. In the adaptive setting, on corruption a party reveals its input to the adversary. We need a bound of this additional information sent to the adversary. This bound $\ell(k)$ corresponds to the sum of the lengths of inputs of parties Q_1, \dots, Q_n .

Discussion about the Protocol. Observe that the entire protocol as described in Figure 1 involves only random strings from honest P_i and honest V_i . Also note that the main proof stage involves an execution of the shadow version of the WI-UARG protocol. As already pointed out an honest execution of this protocol does not involve any actual proof being given. Therefore, the graph generated in an honest execution of $\langle P, V \rangle_i$ will essentially never be Hamiltonian. We provide full details on our simulator for the family of $\langle P, V \rangle$ protocols and the proofs of the security properties in the full version.

4.2 Coin-Flipping Protocol

Now we describe our coin flipping protocol. $\langle A, B \rangle$ is a protocol between two parties A and B . Both A and B in the $\langle A, B \rangle$ protocol get graphs G_1 and G_2 as common input and output a random string of length $\ell'(k)$. We assume that no PPT adversary can output a Hamiltonian cycle in G_1 if B is honest. Similarly, we assume that no PPT adversary can output a Hamiltonian cycle in G_2 if A is honest. Consider the setting in which an instance of the $\langle A, B \rangle$ protocol is being executed. In this setting we consider an adversary \mathcal{A} that adaptively corrupts parties (an honest party reveals its input and random coins to the adversary on corruption) and (assuming dense cryptosystems [14]) require that:

1. For every adaptive adversary \mathcal{A} , there exists a simulator $\mathcal{S}_{\langle A, B \rangle}$ which gets as input a Hamiltonian cycle in G_1 if A is honest (before the start of the protocol), a Hamiltonian cycle in G_2 if B is honest (before the start of the protocol) and a string crs (sampled from the uniform distribution) of length $\ell'(k)$. Furthermore, $\mathcal{S}_{\langle A, B \rangle}$ obtains the input of every party that \mathcal{A} corrupts. In this setting we require that the view of the adversary \mathcal{A} in its interaction with the honest parties and the view of the adversary \mathcal{A} in its interaction with $\mathcal{S}_{\langle A, B \rangle}$ are computationally indistinguishable.
2. The output of the protocol execution is crs as long as either A or B is not corrupted till the end of the protocol.

Our Protocol $\langle A, B \rangle$. $(\text{IDCom}, \text{IDOpen})$ is a graph based commitment scheme. And, (G, E, D) is an encryption scheme with pseudorandom ciphertexts and pseudo-random public keys (of length $\ell_1(k)$). Both parties get graphs G_1 and G_2 as common input.

1. A generates a commitment $c = \text{IDCom}_{G_1}(\alpha; r_1)$, where α is a random string in $\{0, 1\}^{\ell_1(k)}$ and r_1 are the random coins. It sends c to B .
2. B sends a random string $\beta \in \{0, 1\}^{\ell_1(k)}$ to A .
3. A sends (α, r_1) to B .
4. B aborts the protocol if $c \neq \text{IDCom}_{G_1}(\alpha; r_1)$.
5. Both parties set $pk := \alpha \oplus \beta$.
6. A generates a commitment $d = \text{IDCom}_{G_1}(\gamma; r_2)$, where γ is a random string in $\{0, 1\}^{\ell'(k)}$ and sends it to B .
7. B generates commitments $f_i = \text{IDCom}_{G_2}(\delta_i; s_i)$, where δ is a random string in $\{0, 1\}^{\ell'(k)}$ and δ_i is the i^{th} bit of δ . It also generates $e_{i, \delta_i} = E_{pk}(s_i; t_i)$ and $e_{i, 1-\delta_i}$ as a random string in $\{0, 1\}^{\ell_2(k)}$ (where $\ell_2(k)$ is the appropriate length). Finally it sends f_i , $e_{i,0}$ and $e_{i,1}$ for all $i \in [\ell'(k)]$ to A .
8. A sends (γ, r_2) to B .
9. B aborts if $d \neq \text{IDCom}_{G_1}(\gamma; r_2)$. Next, B sends δ_i, s_i, t_i for every $i \in [\ell'(k)]$ to A .
10. A aborts if for some $i \in [\ell'(k)]$, $f_i \neq \text{IDCom}_{G_2}(\delta_i; s_i)$ or $e_{i, \delta_i} \neq E_{pk}(s_i; t_i)$.
11. Both parties output $\gamma \oplus \delta$ as the output of the protocol.

Intuition behind the Proof. If B is honest before Step 9, then $\mathcal{S}_{\langle A, B \rangle}$ equivocates in the messages (sent on behalf of B) and thereby forcing the output of the protocol to a value of its choice. Now consider the case in which B is corrupted before Step 9. In this case we need to force the output of the protocol only if A is not corrupted. In this case the simulator $\mathcal{S}_{\langle A, B \rangle}$ will be able to force the value pk generated in Step 3 of the protocol to a value of its choice. Subsequently it can force the output of the protocol to a value of its choice by extracting the values committed by B in Step 7 and then later equivocating in Step 8. Further note that the simulation itself is straight line. However in proving indistinguishability of simulation from real interaction we do rewind the adversary.⁹ We provide full details on our simulator for the $\langle A, B \rangle$ protocols and the proof of the security properties in the full version.

5 Our Constant Round Protocol

Let f be any adaptively well-formed¹⁰ functionality. In this section we will give a constant round protocol Π that adaptively-securely realizes f . Let Q_1, \dots, Q_n be n parties that hold inputs x_1, \dots, x_n respectively. Let f be the function that they wish to evaluate on their inputs. Furthermore, let $\ell(k) = |x_1| + |x_2| \dots |x_n|$.

⁹ This is not a problem as rewinding is used only in the proof in order to reach a contradiction.

¹⁰ We require[9] that the functionality reveals its random input in case all parties are corrupted.

We start by describing a protocol that adaptively securely realizes the $\mathcal{F}_{n\text{-}crs}$ functionality (Figure 2). Note that whenever a party is corrupted then it reveals its input and random coins to the adversary. In our construction we use a *family of trapdoor generator protocols* $\langle P, V \rangle_i$ where $i \in \{1 \dots m\}$ (Section 4.1) and a *coin flipping protocol* $\langle A, B \rangle$ (Section 4.2). The protocol proceeds as follows.

1. **Trapdoor Creation Phase:** $Q_i \leftrightarrow Q_j$: For all $i, j \in [n]$, such that $i \neq j$, Q_i and Q_j engage in an execution of the protocol $\langle P, V \rangle_t$ (with common input n^2 and $\ell(k)$ where $t = i \cdot (n - 1) + j$), where Q_i plays the role of P_t and Q_j places the role of the V_t . Let $G_{i,j}$ be the output of the protocol. All these executions are done in parallel.
2. **Coin Flipping Phase:** $P_i \leftrightarrow P_j$: For all $i, j \in [n]$, such that $i < j$, Q_i and Q_j engage in an execution of the protocol $\langle A, B \rangle$, denoted as $\langle A, B \rangle^{i,j}$, where Q_i plays the role of A and Q_j plays the role of B with common input $G_{i,j}$ and $G_{j,i}$. Q_i and Q_j output the output of $\langle A, B \rangle^{i,j}$ as $\text{crs}_{i,j}$. All these executions are done in parallel.

Common input: Let Q_1, \dots, Q_n be n parties that hold inputs x_1, \dots, x_n respectively. Furthermore, let $\ell(k) = |x_1| + |x_2| \dots |x_n|$. $\mathcal{F}_{n\text{-}crs}$ sets up a list \mathcal{L} that is initially set to be empty. Let \mathcal{S} be the ideal world adversary and let A at any point be the set of corrupted parties.

1. On receiving a messages (crs, i, j) from party Q (including \mathcal{S}), $\mathcal{F}_{n\text{-}crs}$:
 - If $\exists(\text{crs}, i, j, \text{crs}_{i,j}) \in \mathcal{L}$: Sends $\text{crs}_{i,j}$ to Q .
 - If $(\text{crs}, i, j, \cdot) \notin \mathcal{L}$ and if at least one of Q_i or Q_j is not in A : Samples a random string $\text{crs}_{i,j} \in \{0, 1\}^{\ell'(k)}$, adds $(\text{crs}, i, j, \text{crs}_{i,j})$ to \mathcal{L} and sends $\text{crs}_{i,j}$ to Q .
 - If both $Q_i, Q_j \in A$: Obtain crs from \mathcal{S} and send the obtained crs to Q .
2. On receiving a message $(\text{corrupt}, Q_i)$ from \mathcal{A} , $\mathcal{F}_{n\text{-}crs}$ adds Q_i to A and sends x_i to \mathcal{S} .

Fig. 2. $\mathcal{F}_{n\text{-}crs}$

Theorem 2. *Assuming collision resistant hash functions and dense cryptosystems [14], the constant round protocol just described above adaptively securely evaluates $\mathcal{F}_{n\text{-}crs}$ (Figure 2).*

The formal proof of the above theorem appears in the full version.

Realizing All Functionalities. Now we elaborate on how we can construct an adaptive secure protocol for any functionality.

Theorem 3. *Assuming collision resistant hash functions, trapdoor permutations, augmented non-committing encryption and dense cryptosystems, for any $n \geq 2$, there exists an n -party constant round MPC protocol that is secure against any malicious adversary which may adaptively corrupt at most $n - 1$ parties*

Recall that we have already constructed a protocol that adaptively securely realizes $\mathcal{F}_{n\text{-}crs}$ ideal functionality. Therefore we are left with just constructing a protocol secure in the $\mathcal{F}_{n\text{-}crs}$ -hybrid model. This is implied by the following proposition.

Proposition 1. *Assuming trapdoor permutations and augmented non-committing encryption, for any $n \geq 2$, there exists an n -party constant round MPC protocol in the $\mathcal{F}_{n\text{-}crs}$ -hybrid model that is secure against any malicious adversary which may adaptively corrupt at most $n - 1$ parties.*

Remark on the above Proposition. Note that if security against corruption of all n parties is desired then a proposition (similar to the one above) that yields a protocol with round complexity that depends on the depth of the circuit being evaluated still holds. Additionally in this setting we can get a constant-round protocol if data *erasures* are allowed. We refer the reader to Remark 2 in [20] for discussion on this.

Proof Sketch. The proof of the above proposition is implicit in a number of previous works. For concreteness, we will describe one way of constructing such a protocol. Observe that the $\mathcal{F}_{n\text{-}crs}$ ideal functionality can be split into $\frac{n(n-1)}{2}$ ideal functionalities each generating a common random string for each pair of parties (each of these ideal functionalities works correctly as long as at least one of the two parties it is serving is honest). Next note that, using [9], given access to a common random string, we can construct an adaptively secure OT protocol. Using this protocol and applying the UC composition theorem [26] (Composing Different Setups, Page 61), multiple times, we can construct a protocol that achieves adaptively secure OT¹¹ between every pair of parties (as long as at least one of the two parties it is serving is honest). Finally, using these OT channels [20] we can adaptively securely realize any functionality.

Acknowledgements. Research supported in part from a DARPA/ONR PRO-CCEED award, NSF grants 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

We gratefully thank Abhishek Jain, Yuval Ishai, Manoj Prabhakaran, and Akshay Wadia for valuable discussions about this work. We would also like to thank Divya Gupta and the anonymous reviewers of CRYPTO 2012 for their comments on the previous drafts of this paper.

References

1. Yao, A.C.: How to generate and exchange secrets. In: Proc. 27th FOCS, pp. 162–167 (1986)
2. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)

¹¹ [9] assume trapdoor permutations and augmented non-committing encryption in their construction.

3. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC, pp. 639–648 (1996)
4. Beaver, D.: Adaptive zero knowledge and computational equivocation (extended abstract). In: STOC, pp. 629–638 (1996)
5. Lindell, Y., Zarosim, H.: Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *J. Cryptology* 24(4), 761–799 (2011)
6. Beaver, D.: Equivocable Oblivious Transfer. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 119–130. Springer, Heidelberg (1996)
7. Beaver, D.: Adaptively Secure Oblivious Transfer. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 300–314. Springer, Heidelberg (1998)
8. Katz, J., Ostrovsky, R.: Round-Optimal Secure Two-Party Computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
10. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved Non-committing Encryption with Applications to Adaptively Secure Protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
11. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
12. Pass, R., Wee, H.: Black-Box Constructions of Two-Party Protocols from One-Way Functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009)
13. Canetti, R.: Security and composition of multi-party cryptographic protocols. *Cryptology ePrint Archive*, Report 1998/018 (1998), <http://eprint.iacr.org/>
14. De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, SFCS 1992, pp. 427–436. IEEE Computer Society, Washington, DC (1992)
15. Beaver, D., Haber, S.: Cryptographic Protocols Provably Secure against Dynamic Adversaries. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 307–323. Springer, Heidelberg (1993)
16. Barak, B.: How to go beyond the black-box simulation barrier. In: Proc. 42nd FOCS, pp. 106–115 (2001)
17. Pass, R., Rosen, A.: Bounded-concurrent secure two-party computation in a constant number of rounds. In: FOCS, pp. 404–413 (2003)
18. Pass, R., Rosen, A.: New and improved constructions of nonmalleable cryptographic protocols. *SIAM J. Comput.* 38(2), 702–752 (2008)
19. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552 (2005)
20. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
21. Katz, J., Ostrovsky, R., Smith, A.: Round Efficiency of Multi-Party Computation with a Dishonest Majority. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 578–595. Springer, Heidelberg (2003)
22. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: Proc. 36th STOC, pp. 232–241 (2004)

23. Feige, U., Shamir, A.: Zero Knowledge Proofs of Knowledge in Two Rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, Heidelberg (1990)
24. Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM J. Comput. 38(5), 1661–1694 (2008)
25. Naor, M.: Bit commitment using pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991); Preliminary version In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 128–136. Springer, Heidelberg (1990)
26. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols (2000), <http://eprint.iacr.org/>

Collusion-Preserving Computation

Joël Alwen¹, Jonathan Katz^{2,*}, Ueli Maurer¹, and Vassilis Zikas^{2,**}

¹ ETH Zürich

{alwenj,maurer}@inf.ethz.ch

² University of Maryland

{jkatz,vzikas}@cs.umd.edu

Abstract. In *collusion-free* protocols, subliminal communication is impossible and parties are thus unable to communicate any information “beyond what the protocol allows.” Collusion-free protocols are interesting for several reasons, but have specifically attracted attention because they can be used to reduce trust in game-theoretic mechanisms. Collusion-free protocols are impossible to achieve (in general) when all parties are connected by point-to-point channels, but exist under certain physical assumptions (Lepinski et al., STOC 2005) or when parties are connected in specific network topologies (Alwen et al., Crypto 2008).

We provide a “clean-slate” definition of the stronger notion of collusion *preservation*. Our goals in revisiting the definition are:

- To give a definition with respect to *arbitrary* communication resources (including as special cases the communication models from prior work). We can then, in particular, better understand what types of resources enable collusion-preserving protocols.
- To construct protocols that allow no *additional* subliminal communication when parties *can* communicate via other means. (This property is *not* implied by collusion-freeness.)
- To support *composition*, so protocols can be designed in a modular fashion using sub-protocols run among subsets of the parties.

In addition to proposing the definition, we explore implications of our model and show a general feasibility result for collusion-preserving computation of arbitrary functionalities. We formalize a model for concurrently playing multiple extensive-form, mediated games while preserving many important equilibrium notions.

1 Introduction

Subliminal channels [28,29,30] in protocols allow parties to embed “disallowed” communication into protocol messages, without being detected. (For example, a party might communicate a bit b by sending a valid message with first bit equal to b .) The existence of subliminal channels is often problematic. In a large-scale distributed computation, for instance, subliminal channels could allow two

* Research supported in part by NSF grant #1111599.

** Supported in part by a fellowship from the Swiss National Science Foundation (Project No. PBEZP2-134445).

parties to coordinate their actions (i.e., to collude) even if they were not aware of each other in advance. In other settings, parties may be disallowed or otherwise unable to communicate “out-of-band,” and it would be undesirable if they could use the protocol itself to convey information.

More recently, subliminal channels have arisen as a concern in the context of cryptographic implementations of game-theoretic mechanisms. Here, informally, there is a game in which parties send their types/inputs to a trusted party which then computes an outcome/result. One might naturally want to replace the trusted party with a cryptographic protocol executed by the parties [13,15,8,23,24,25,21,1,2,19,20]. Using protocols for secure multi-party computation (e.g., [17]) can preserve Nash equilibria; however, such protocols do *not* suffice for implementing general equilibria precisely because they have subliminal channels and thus enable collusion in the real world even if such collusion is impossible in the original game.

This realization has motivated the investigation of *collusion-free* protocols that do not allow subliminal communication [23,24,25,21,19,5,3,20]. Collusion-free protocols for computing non-trivial functions are impossible when parties are connected by pairwise communication channels, and so researchers have turned to other communication models. Collusion-free computation of arbitrary functionalities is possible if parties have access to a semi-trusted “ballot box” and can communicate publicly via (physical) envelopes [24,21,19,20], or if parties are connected (via standard communication channels) to a semi-trusted entity in a “star network” topology [5,3].

1.1 A New Definition: Collusion Preservation

The works of Izmalkov et al. [24,21,19,20] and Alwen et al. [5,3] give incomparable definitions of collusion freeness, each tailored (to some extent) to the specific communication models under consideration. We revisit these definitions, and propose a stronger notion called *collusion preservation*. Intuitively, collusion-free protocols ensure that parties can communicate no more with the realizing protocol than using *only* the ideal functionality, whereas collusion-preserving protocols provide a stronger guarantee: the parties can communicate no more when running the protocol and using arbitrary fixed external channels, than they could using the ideal functionality and the same external channels. Our aim here is to provide a *clean, general-purpose* definition that better handles *composition*, both when collusion-preserving protocols are run as sub-routines within some larger protocol, as well as when collusion-preserving protocols are run concurrently with arbitrary other protocols (whether collusion-preserving or not). In what follows, we give an overview of our definition and expound further on the above points.

Overview of our definition. We follow the simulation-based definitional paradigm used by Alwen et al. [5,3]: In the real-world execution of a protocol, different adversaries can corrupt different parties. Importantly, these adversaries cannot communicate directly; instead, all parties are connected in a “star network”

topology with a semi-trusted mediator. (This is in contrast to usual cryptographic definitions, which assume a “monolithic” adversary who controls all corrupted parties and can coordinate their actions.) Two notions of stand-alone security [16] are then defined, depending on whether or not the mediator is honest:

1. **CONDITIONAL COLLUSION FREENESS:** When the mediator is honest, collusion freeness is required.
2. **FALLBACK SECURITY:** When the mediator is dishonest, we cannot hope for collusion freeness any more. Nevertheless a strong notion of security can be achieved; namely real/ideal adversaries are allowed to communicate arbitrarily, and the protocol is required to satisfy the standard (stand-alone) security definition [16].

We strengthen and extend the definition of collusion freeness in several ways. Firstly, rather than considering a specific “star network” topology with a special party (the mediator) in the center [5,3], or the specific physical assumptions of [24,21,19,20], we consider a general *resource* to which the parties have access in the real world. This resource is the only means of “legal” communication in the real world (though as we will see in a moment there may be other “illicit” means of communication available). In addition to being more general, our definition allows us to characterize the minimal properties of resources which achieve collusion-preserving computation. Secondly, we formulate our definitions in a universally composable (UC) [9] fashion, where there is an environment controlling the entire execution.¹ This has significant ramifications, since the environment *itself* can now act as a communication channel for the adversaries. If the environment chooses to allow no communication between the adversaries, then our definitions essentially “default” to the previous setting of collusion freeness. Crucially, however, if the environment allows the adversaries to communicate c bits of information “for free”, then a collusion-preserving protocol ensures that the adversaries cannot communicate more than c bits (on top of the communication allowed by the ideal functionality) by running the protocol in the presence of the stated resource. (We show below a simple counter-example demonstrating that collusion freeness does not imply collusion preservation.) Moreover, we prove a universal composition theorem, thereby improving upon the results of [5,3], which do not claim nor realize any form of composition, as well as the results of [24,21,19,20] which obtain only a limited sort of composition; see below.

Collusion preservation is stronger than collusion freeness. We motivate the need for a composable definition with an example: Consider a protocol π that is collusion-free in the mediated (star network) setting of [5,3]. We obtain a new protocol π' , identical to π except for the following two modifications to the mediator’s behavior (where λ is the security parameter):

1. The mediator takes a special message $m \in \{0,1\}^{2\lambda}$ from P_1 . In response, the mediator chooses a random $r \in \{0,1\}^\lambda$, sends it to P_1 , and stores (r, m) .

¹ Actually, we use the *generalized* UC (GUC) framework [10] as our starting point.

2. The mediator takes a special message $r' \in \{0,1\}^\lambda$ from P_2 . If the mediator has a stored tuple of the form (r', m) , it sends m to P_2 (and otherwise simply ignore r').

It is not hard to see that π' remains collusion-free: intuitively, this is because P_2 can guess $r' = r$ with only negligible probability.² However, π' is *not* collusion preserving. Specifically, if P_1 and P_2 have access to a λ -bit channel then they can use π' to communicate 2λ bits!

The above counter-example can be interpreted in several ways. One could imagine that π' is run in a setting in which P_1 and P_2 have access to a physical channel that only allows communication of λ bits. Alternately, the parties might be running π' while they are concurrently running some *other* protocol that is not collusion-free and enables the parties to (subliminally) communicate λ bits. Either way, the implication is the same: a collusion-free protocol may potentially allow additional communication than the corresponding ideal specification.

In the following we give an overview of the main results on the paper.

PROTOCOL COMPOSITION. The protocols of Izmalkov et al. [24,21,19,20] are collusion-free only when *at least one party running the protocol is honest*. The reason is that in their communication model parties have the *ability* to communicate arbitrary information (but their protocols guarantee that if an honest party is watching then it will detect any such communication). This limitation may not appear to be a problem, since one typically does not care to provide any guarantees once all parties are malicious. It becomes a problem, however, when collusion-free protocols are used as sub-routines within some larger protocol. Consider, for example, a collusion-free protocol Π for three parties P_1, P_2, P_3 in which each pair of parties runs some collusion-free sub-protocol π between themselves. If P_1 and P_2 are malicious, then π may provide no guarantees which means that they may now be able to communicate an unlimited amount of information; this could clearly be problematic with regard to the “outer protocol” Π .³

NECESSARY ASSUMPTIONS. The main criticism on the mediated model of [5,3] concerns the strength of the mediator as an assumption. In particular, an open question is whether or not one can reduce the power/complexity of the mediator without sacrificing collusion-freeness. Justifying the tightness of our model, we show that any recourse which allows for realizing arbitrary (well-formed) functionalities in a collusion preserving (or even collusion-free) manner satisfies three essential properties referred to as *isolation*, *independent randomness*, and *programmability*. Intuitively, the mediator is one of the simplest digital resources having all these properties, simultaneously. Note that the physical assumptions of [23,24,25,21] also implicitly satisfy these properties.

² In particular the simulators for π' can behave just as for π with the only modification that the simulator for P_1 responds with a random message r when it receives the special message m from its adversary.

³ Izmalkov et al. implicitly avoid this issue by having *every* party observe all sub-protocols that are run.

GENERAL FEASIBILITY WITH GUC FALBACK. Complementing and motivating our definitional work, we provide a completeness result for strong realization of a large class of functionalities. More concretely, for any functionality in this class we provide a protocol compiler and a particular resource which satisfies a universally composable version of the security definition from [3]: the compiled protocol provides Collusion Preserving (CP) security when executed with this particular resource, and, as a strong fallback, when executed with an arbitrary resource, it achieves GUC-type security, i.e., emulation by “monolithic” simulators.

SYNCHRONIZATION POLLUTION. The types of correlation ruled out by collusion-free protocols fall into two categories: the first is due to “*subliminal channels*” (i.e. means of communication not present in the ideal world) and the second is due to “*randomness pollution*” (i.e. publicly visible correlated randomness generated during the protocols execution). We observe and mitigate, a new type of correlation between split adversaries (called *synchronization pollution*) which is crucial to achieving concurrent (and thus universal) composability. Synchronization pollution arises for example, in settings where a multi-round (or, more generally, a *multi-stage*⁴) protocol is used to CP-realize a non-reactive functionality but the parties have no means, external to the protocol, of synchronizing their actions, e.g., there are no synchronized clocks. In fact, the problem can arise for any security notion with split adversaries, where the relative order of events occurring in different parties interfaces is observable. Intuitively, an environment witnessing the execution of such a protocol could keep track of these events *as they occur* (say by instructing the adversaries to corrupt all players, run the honest protocol and announce each event as it takes place). However, if the ideal functionality is used instead, a priori the simulators have no means for coordinating the simulation of the events in the correct order as the functionality is good only for a single round of communication and otherwise the simulators have no further means of synchronizing their outputs. In the full version we demonstrate how the above issue can lead to real attacks when protocols are executed concurrently.

We resolve this issue by considering ideal functionalities with a means to provide a (minimal) amount of synchronization, namely output-synchronization. That is for an interactive CP protocol, at the very least, simulators are allowed to coordinate when they produce the final output of the computation for the environment. Surprisingly we are able to show that this is the *only* additional synchronization required for good split-simulation in the ideal world. Intuitively this is because our protocol hides all other properties *and events* in the execution (such as at which stage of the protocol the execution is currently in) from adversaries even if they corrupt all parties and their ongoing views are combined by the environment. To the best of our knowledge, this is the first protocol to exhibit such behavior. For example in the case of the sequence of works [24,21,19,20] all protocols contain multiple publicly verifiable (i.e. visible) events. Therefore

⁴ Intuitively, by *multi-stage* we mean a protocol which results in at least one triplet of events (E_1, E_2, E_3) such that each is noticeable by at least one party and which can only occur in a fixed order $E_1 \rightarrow E_2 \rightarrow E_3$.

adversaries present *during* the execution of such protocols obtain significantly more than mere output synchronization. Moreover, to attain concurrent compositability the ideal worlds would have to be augmented with some means for simulators to have the same level of synchronicity afforded by those events (similar to the synchronization wrapper). On the other hand, in previous works dealing with the mediated model [5,3] the current stage of the protocol execution is not hidden from corrupt players. We remark that for both lines of work, the protocols remain sequentially composable because once the execution has completed, the simulators can generate appropriate transcripts atomically. The problem arises only if the transcripts are inspected by an on-line distinguisher.

IMPLICATIONS FOR GAME THEORY. We show how to use the results of the CP framework to adapt the traditional models of algorithmic game theory [27] and mechanism design to a more practical setting. First we generalize the standard stand-alone model of game play for (extensive form computational) games to a concurrent setting in which multiple games with different player sets using different mechanisms are being played in an arbitrarily interleaved manner. Next we introduce a strong notion of equivalence between sets of concurrent games which we call *isomorphic game sets*⁵. Intuitively a pair of isomorphic game sets have equivalent strategies in that they induce approximately the same outcome (and payoffs). Moreover for any strategy a player may use in one game set and any action taken in the course of that strategy it is easy for the player to compute the equivalent action(s) in the isomorphic game set regardless of which strategies are used by all other players involved and even of the strategy which dictated to play the original action. We provide evidence that this transformation between actions of isomorphic game sets preserves, in particular, many desirable notions of stability such as k -resilience [7], correlated equilibria [6], dominant strategies and t -immunity [2].

Next we apply our feasibility results for the CP framework to make a second important step in bringing the field of mechanism design closer to a practical setting. Leveraging the GUC fallback property we show how to significantly reduce the type of trust placed in a certain class of mechanisms. Traditionally game theorists place complete trust in mechanisms not only to enforce such properties as the isolation of players and fairness of output distribution, but also to compute outcomes correctly and preserve the players' privacy.⁶ We use the results from the cryptographic section of this work to show how to completely remove the latter two types of trust while still obtaining an isomorphic game set. Roughly speaking, this is done by replacing the original mechanism with a CP secure protocol running over a "less trusted" mechanism such that the resulting game is equivalent to the original even with respect to an arbitrary set of concurrently running games. The fact that the isomorphic strategies can be computed locally and automatically by each player without considering other

⁵ Our notion of isomorphic game sets is closely related to that of [26].

⁶ For example in a poker game it is implicitly assumed that the dealer deals from a uniform random deck and that they don't reveal the cards exchanged by one player to another.

players' strategies permits both mechanism designer and player to operate in the more traditional (presumably cleaner) "fully trusted" setting while actual game play occurs in the more practical setting with reduced trust.

In comparison to results of [18,19,20] which provide information theoretic-equivalence between games using an unconventional model of computation, the results in this paper provide only computational equivalence but use a standard computational model. However, their notion of composition is weaker in two ways. Conceptually it is not scalable but more concretely it seems to allow for only rather limited notion of concurrency. In particular protocols that implement a mechanism must be run atomically with respect to actions in any concurrent games. In contrast, the notion obtained in this paper is fully UC composable in the more traditional sense. On the other hand, while our protocols prevent signaling via aborts as in [18], they do not provide the full robustness to aborts of [19,20]. Finally, the amount of randomness in the public view of our protocols is limited to a single pre-computation round which can be run before types are distributed. From that point on there is no further "randomness pollution". This is similar to [22], better than [18] (where even executions of a protocol produce randomness pollution) but weaker than [19,20] which do not produce any randomness pollution at all.

Relation to abstract cryptography [26] and to UC with local adversaries [12]. Collusion-preserving computation can be described in the Abstract Cryptography (AC) framework by Maurer and Renner [26], but this is beyond the scope of this paper. Apart from being stated at an abstract level, two relevant aspects of AC in our context are that there is no notion of a central adversary (and simulators are local) and that all resources are modeled explicitly, which allows to capture the absence of resources (e.g. communication channels). In concurrent and independent work with ours, Canetti and Vald [12] also consider the question of extending the notion of collusion freeness (local adversaries/simulators) to the universally composable setting. Although many of our results can be proved also in their framework, the two models have considerable conceptual and formal differences. We refer to the full version of our work [4] for a discussion of the main differences between the two approaches.

2 Collusion-Preserving Computation

In this section we define our framework for investigating universally composable collusion freeness, namely *collusion-preserving* computation. On the highest level the idea is to combine the strong composable properties of the GUC framework of [10] with the model of split simulators along the lines of [3].

2.1 Preliminaries and Notation

We denote by $[n]$ the set $\{1, \dots, n\}$ (by convention $[0] = \emptyset$) and for a set $\mathcal{I} \subseteq [n]$ we denote by $\overline{\mathcal{I}}$ the set $[n] \setminus \mathcal{I}$. Similarly, for element $i \in [n]$ we write \overline{i} to denote the set $[n] \setminus \{i\}$. Using this notation we denote by $\mathsf{A}_{\mathcal{I}}$ a set of ITMs $\{\mathsf{A}_i\}_{i \in \mathcal{I}}$.

For input tuple $x_{\mathcal{I}} = \{x_i\}_{i \in \mathcal{I}}$ we write $\mathsf{A}_{\mathcal{I}}(x_{\mathcal{I}})$ to denote that for all $i \in \mathcal{I}$ the ITM A_i is run with input x_i (and a fresh uniform independent random tape). We assume a passing familiarity with the GUC framework and refer to the full version of this paper [4] for a description of the main features we use.

An intuitive description. We include an informal description of our CP framework and discuss its basics; a complete description can be found in [4]. Starting from the GUC model we make the following modifications:

SPLIT ADVERSARIES/SIMULATORS: Instead of a monolithic adversary/simulator we consider a set of n (independent) PPT adversaries $\mathsf{A}_{[n]} = \{\mathsf{A}_i : i \in [n]\}$, where A_i correspond to the adversary associated with the player i (and can corrupt at most this party). Moreover, we ask that for each $\mathsf{A}_i \in \mathsf{A}_{[n]}$ there exists an (independent) simulator Sim_i .

CORRUPTED-SET INDEPENDENCE: We also require that the simulators do not depend on each other. In other words the code of simulator Sim_i is the same for any set of adversaries $\mathsf{A}_{[n]}$ and $\mathsf{B}_{[n]}$ as long as $\mathsf{A}_i = \mathsf{B}_i$.

Resources, shared functionalities, and exclusive protocols. The main difference between a CP functionality \mathcal{R} and a GUC one is that besides the n interfaces to the (honest) parties it also has interfaces to *each* of the n adversaries $\mathsf{A}_1, \dots, \mathsf{A}_n$ (see Figure 1). In other words rather than n interfaces a CP functionality has $2n$ interfaces.⁷ Moreover,

similar to the GUC framework (but in contrast to plain UC) we distinguish between two types of functionalities: *resources* (i.e., functionalities that maintain state only with respect to a single instance of a protocol) which we denote with capital calligraphic font as in “ \mathcal{R} ” and *shared functionalities* (i.e., functionalities that can maintain state across protocol instances) which we denote with an additional over-line as in “ $\bar{\mathcal{G}}$ ”.

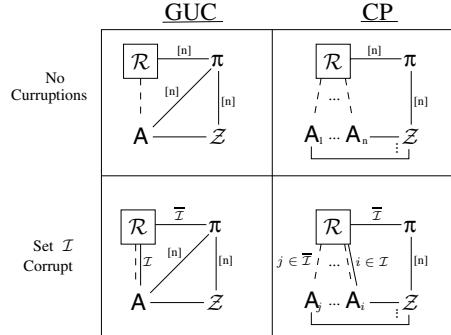


Fig. 1. Corruption of player set $\mathcal{I} \subseteq [n]$: GUC model vs. CP model. (Setup functionalities are left implicit.)

The \mathcal{R} -hybrid world. A CP execution in the \mathcal{R} -hybrid world is defined via a straightforward generalization to the analogous GUC execution. We denote the output of the environment \mathcal{Z} when witnessing an execution of protocol $\pi := \pi_{[n]}$

⁷ Intuitively, the reason for having n additional interfaces (one for each adversary) is that adversaries should not communicate with each other, therefore they cannot share the same interface; nevertheless, as in all composable frameworks, each of them should be able to communicate with the assumed resource, e.g., for scheduling delivery of messages to the corresponding party.

attacked by adversaries $\mathbf{A} := \mathbf{A}_{[n]}$ in the \mathcal{R} -hybrid model as $\text{CP-EXEC}_{\pi, \mathbf{A}, \mathcal{Z}}^{\mathcal{R}}$. Finally, we say a protocol π is \mathcal{R} -exclusive if it makes use of no other resources (shared or otherwise) than \mathcal{R} .

Bounding the number of calls to resources. A primary difference between how executions in a \mathcal{R} -hybrid world are modeled in the GUC and CP frameworks is that in CP parties can communicate with at most a *single* instance of \mathcal{R} . This is in contrast to all other UC like models where the \mathcal{R} -hybrid world is understood to mean that parties can make as many calls as they wish to \mathcal{R} instantiating a new copy for each new invocation of \mathcal{R} . Note that, for a composable notion with split adversaries fixing the number of instances of functionalities/resources available to adversaries is in fact *crucial* for capturing the desired intuition of collusion freeness. For example a primary motivation of this work is to provide a way for reducing trust on the mediators used in mechanism design by providing protocols which can be used to replace the interaction with the mediator. If we do not restrict the number of instances of the mechanism with which parties can interact then there is no meaningful way to capture a game which calls for only a single instance. Unless explicitly stated otherwise, in the present work, whenever we write a functionality we assume a single instance of it. For a longer discussion we refer to the full version [4].

Definition 1 (Collusion-Preserving Computation). Let $\bar{\mathcal{G}}$ be a setup, \mathcal{R} and \mathcal{F} be n -party resources, π be a $\{\bar{\mathcal{G}}, \mathcal{R}\}$ -exclusive protocol and ϕ be a $\{\bar{\mathcal{G}}, \mathcal{F}\}$ -exclusive protocol (both n -party protocols). Then we say that π collusion-preservingly (CP) emulates ϕ in the $\{\bar{\mathcal{G}}, \mathcal{R}\}$ -hybrid world, if there exists a collection of efficiently computable transformations $\text{Sim} = \text{Sim}_{[n]}$ mapping ITMs to ITMs such that for every set of adversaries $\mathbf{A} = \mathbf{A}_{[n]}$, and every PPT environment \mathcal{Z} the following holds: $\text{CP-EXEC}_{\pi, \mathbf{A}, \mathcal{Z}}^{\bar{\mathcal{G}}, \mathcal{R}} \approx \text{CP-EXEC}_{\phi, \text{Sim}(\mathbf{A}), \mathcal{Z}}^{\bar{\mathcal{G}}, \mathcal{F}}$.

Realization, reductions, and the “ \sqsubseteq ” notation. We use the following notation (for details see [4]). If for functionality \mathcal{F} , an \mathcal{R} -hybrid protocol π CP-emulates \mathcal{F} ⁸ then we say that π realizes \mathcal{F} (in the \mathcal{R} -hybrid world), and denote it by $\mathcal{F} \sqsubseteq_{\pi}^{\text{CP}} \mathcal{R}$, which can intuitively be read as “ \mathcal{F} CP-reduces to \mathcal{R} via protocol π ”. By omitting π in this notation we denote simply the existence of some protocol for which the relation holds. We also use “ \sqsubseteq^{GUC} ” to denote the analogous relation but for GUC-realization. To simplify notation and maintain consistency with previous UC-type works, whenever an explicit protocol for the honest players is missing in the CP-EXEC notation then it is implicitly assumed that they are running the dummy \mathcal{F} -hybrid protocol $\mathcal{D}^{\mathcal{F}}$ that forwards all its inputs from \mathcal{Z} to \mathcal{F} and vice-versa. For example we might write $\text{CP-EXEC}_{\mathbf{A}_{[n]}, \mathcal{Z}}^{\mathcal{F}}$ when the honest players are running $\mathcal{D}_{[n]}^{\mathcal{F}}$.

Composition theorem. As a main motivation for the CP model we put forth the goal of providing a formal and rigorous notion of composability for collusion-free security. We formalize a strong (universally) composable property of CP security

⁸ Formally we would write $\mathcal{D}^{\mathcal{F}}$ instead of \mathcal{F} .

in the following theorem. The proof can be found in the full version [4]. As part of the proof we provide a useful tool for proving CP security of protocols in the form of a much simplified security notion which proves to be almost as powerful yet far easier to work with.

Theorem 1 (Composition). *Let \mathcal{R} be an arbitrary resource and $\bar{\mathcal{G}}$ be a global setup (i.e. shared) functionality. Let ρ, π and ϕ be n -party protocols in the $\{\bar{\mathcal{G}}, \mathcal{R}\}$ -hybrid world such that π and ϕ are $\bar{\mathcal{G}}$ -subroutine respecting [10]. If π CP-emulates ϕ and ρ uses ϕ as a subroutine then $\rho^{\pi/\phi}$ CP-emulates ρ in the $\{\bar{\mathcal{G}}, \mathcal{R}\}$ -hybrid world.*

Relations to existing security notions. The weaker notion of collusion free computation [4,22] can be described as the special case of CP which assumes off-line environment, i.e., the environment does not interact with the adversaries during the computation. In the full version [4] we prove a pair of lemmas relating CP results to matching GUC results. The first formalizes the intuitive claim that CP security is at least as strong as GUC security via a lemma stating that CP realization essentially implies GUC realization. A bit more precisely, we describe a natural mapping of CP functionalities to analogous GUC functionalities $\mathcal{F} \mapsto [\mathcal{F}]$. Then we show that if a protocol CP realizes \mathcal{F} in the \mathcal{R} -hybrid world, then the same protocol executed in the analogous GUC $[\mathcal{R}]$ -hybrid world GUC realizes the analogous GUC functionality $[\mathcal{F}]$. The second establishes the other direction, i.e., translating GUC security statements to statements in CP, and is useful as a primary building block for our feasibility. To this end we define the \mathcal{R}_{ins} functionality which is an adaptation to the CP setting of the standard complete network of UC insecure channels. The main difference is that messages from say player P_i to P_j (as depicted in Figure 2) are first given to A_i and then A_j before it is delivered to P_j . Adversaries can modify (or not deliver) the message at will.

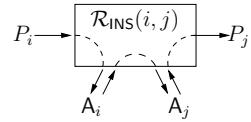


Fig. 2. Insecure Channel with split adversaries

3 Necessary Assumptions for Collusion Preservation

Having defined the CP framework and verified its composition properties, we turn to the next major goal of this work: to provide a resource with which we can (constructively) CP-realize as many functionalities as possible. Ideally we would like to obtain a CP-complete resource, namely one from which *any* reasonable functionality can be realized. Indeed, in the next section we describe just such a resource which we call the *mediator*. However, we must first justify the seemingly strong assumptions we will make when defining the mediator by showing their necessity. To this end, we demonstrate three necessary properties a given resource must have for it to be CP-complete. As corollaries to these results we rule out realizing large classes interesting functionalities using virtually all common communication resources such as fully connected networks and broadcast channels. Beyond this, due to their generality, we believe that given a target

ideal functionality \mathcal{F} (such as an auction mechanism or voting functionality), these results provide significant insights into the minimal assumptions about real world communication channels which can be used to CP realize \mathcal{F} . In the following we sketch these properties, and refer to the full version of this paper for a formal description using the language of our framework

Correlation is not free. Fundamentally, models with a monolithic adversary already allow *perfect* coordination between all interfaces connected to corrupt players. However, when adversaries are split (and a priori isolated) such coordination is not given to adversaries for free anymore. So the security requirements in executions where all players are corrupt are still non-trivial. Instead bounds may still be required on the amount of coordination between the behavior on different interfaces.⁹ By analyzing the implications of the security requirements in such settings we show the necessity of the following properties.

1. (ISOLATION) Consider a statement of the type $\mathcal{F} \sqsubseteq^{CP} \mathcal{R}$. Intuitively this holds only if \mathcal{R} can isolate (corrupt) players as much as \mathcal{F} . We formalize this by showing that (roughly speaking) for any amount of communication that adversaries can obtain from \mathcal{F} , it must be that \mathcal{R} does not allow any more communication. More specifically, we define an extremely weak ideal channel \mathcal{C} and show that if \mathcal{F} can be used to obtain \mathcal{C} by collaborating adversaries controlling *all* interfaces to \mathcal{F} , then \mathcal{R} cannot be used to obtain such a channel with more bandwidth. Given how weak \mathcal{C} is and how much power adversaries have over \mathcal{F} for obtaining it somehow, this result has some far reaching consequences with respect to standard communication models. Not only are they provably not CP-complete but in fact they can not be used to realize almost *any* interesting functionalities (other than themselves).
2. (RANDOMNESS) The second property states that any CP-complete resource must have its own internal randomness upon which its output depends. To prove this, we show that for any resource \mathcal{R} which can be used for CP realizing ideal coin flipping as well as key agreement, \mathcal{R} can not be deterministic.
3. (PROGRAMMABILITY) Finally we define the notion of a programmable resource and show that any CP-complete resource must be programmable. Intuitively a programmable resource can be thought of as being instantiated with a special parameter upon which its behavior depends in a non-trivial way. By non-trivial we mean that for at least one pair of possible values of the parameter the resulting pair of behaviors can not be CP-reduced to each other. Indeed the mediator resource in the next section is programmable and our protocol compiler not only outputs a protocol but also the parameters with which the mediator must be instantiated.

⁹ In a stand-alone setting one might ask why this is even an interesting case (for example the stand-alone notion of [24] explicitly rules it out). But for a composable security notion (for example with the application of modular protocol design in mind) it is vital to consider such executions. Moreover in the context of game theory where there is no such notion of “honest” behavior all players behave as adversaries in a cryptographic sense.

4 GUC Fallback Security

Without any further requirements CP security, as defined in Definition 1, can be easily achieved from an appropriate resource. Indeed, because the resource is completely trusted it could trivially be the functionality we are trying to compute. However, such trust is a rare commodity and so one might ask for a better solution. To that end we add a second property which we call “fallback security”. The goal is to capture what kind of security remains if the protocol is run not with the resource it was designed for but with an arbitrary (potentially malicious) resource instead. Note that the trivial solution provides essentially no fallback security at all. However, we will show, perhaps somewhat surprisingly, that in fact a very strong type of security can still be achieved; namely GUC-like realization.

Definition 2 (CP-realization with GUC fallback). *For setup $\bar{\mathcal{G}}$, functionalities \mathcal{F} and \mathcal{R} , we say that a protocol π CP-realizes a functionality \mathcal{F} with GUC fallback in the $\{\bar{\mathcal{G}}, \mathcal{R}\}$ -hybrid model if it has the following two properties:*

- CP SECURITY: π CP-realizes \mathcal{F} when using \mathcal{R} , i.e., $\{\bar{\mathcal{G}}, \mathcal{F}\} \sqsubseteq_{\pi}^{CP} \{\bar{\mathcal{G}}, \mathcal{R}\}$
- GUC FALBACK: For any efficient resource \mathcal{R}^* the protocol π still GUC-realizes \mathcal{F} , i.e., $\forall \mathcal{R}^* : \{\bar{\mathcal{G}}, \mathcal{R}_{ins}, \mathcal{F}\} \sqsubseteq_{\pi}^{GUC} \{\bar{\mathcal{G}}, \mathcal{R}^*\}$

Recall that the (G)UC plain model implicitly assumes $[\mathcal{R}_{ins}]$. Thus, by applying our CP-to-GUC translation from Section 2 (and omitting the redundant $[\mathcal{R}_{ins}]$ term) we have that GUC fallback security directly implies: $\{\bar{\mathcal{G}}, [\mathcal{F}]\} \sqsubseteq_{\pi}^{GUC} \{[\bar{\mathcal{G}}], [\mathcal{R}^*]\}$. Intuitively this means that π run with (even a single instance of) \mathcal{R}^* and arbitrarily coordinated adversaries still GUC realizes $[\mathcal{F}]$.

We note that as an alternative, by restricting the class of resources \mathcal{R}^* for which the fallback is desired one could, in turn, hope for weaker but still non-trivial fallback properties. This could reflect the real world settings where moderate guarantees about the behavior of the resource are given but it is still undesirable to completely trust the resource. In this sense the feasibility result in this work demonstrates that, at the very least, GUC fallback can be achieved even when no moderate guarantees of any type are made.

5 A General Feasibility Result

We are now ready to state and prove a general feasibility result. Roughly speaking we describe an (efficient) programmable resource $\{\mathsf{M}_{\mathcal{F}}\}_{\mathcal{F} \in \{0,1\}^*}$, called the *mediator*, parameterized by descriptions of functionalities, such that for any \mathcal{F} in a large class of functionality, we can design a protocol π (using setup $\bar{\mathcal{G}}$) which CP-realizes \mathcal{F} with GUC fallback in the $\{\bar{\mathcal{G}}, \mathsf{M}_{\mathcal{F}}\}$ -hybrid model.

Output-synchronized functionalities. In contrast to previous frameworks, because we consider split simulators the environment \mathcal{Z} has an additional means

for distinguishing between executions. Briefly, \mathcal{Z} can measure the amount of on-line synchronization that taking part in an execution provides to sets of adversaries. In contrast all actions taken by monolithic adversaries during an execution are already inherently perfectly synchronized so no such strategy exists in (G)UC frameworks. We address this by introducing the class of *output-synchronized* functionalities. For a arbitrary functionality \mathcal{F} we write $\widehat{\mathcal{F}}$ to denote the “output synchronized” version of \mathcal{F} . That is $\widehat{\mathcal{F}}$ consists of a wrapper (the *synchronizing shell*) and inner functionality \mathcal{F} . The synchronizing shell works as follows: On it’s first activation, $\widehat{\mathcal{F}}$ sends a request to one of the simulators (e.g. the one with the smallest ID) to acquire the index of the desired output round. Let R denote the response of this simulator (if no valid R is received then set $R := 1$). Subsequently, all inputs are forwarded to \mathcal{F} . Finally, outputs of \mathcal{F} are not immediately delivered to their recipient. Rather they are recorded are given only *upon request* from the recipient and only after R subsequent *complete rounds* have been observed (each output-request which is issued before that is answered by a default message \perp). By a complete round we mean a sequence of at least one activation from every player (or the corresponding simulator) in an arbitrary order. Intuitively this modification provides minimal synchronization between adversaries. For a more detailed discussion of output synchronization we refer to [4].

To formalize our main feasibility theorem we introduce the following terminology (for a more accurate description we refer to [4]): A functionality is *well-formed* [11] if its behavior is independent of the identities of the corrupted parties, and it is *aborting* if it accepts a special input **ABORT** from corrupted parties, in which case it outputs **ABORT** to all parties (note that this is the only type of functionalities we can compute while tolerating a corrupted majority). A protocol is said to be *setup off-line* if it precedes all other computation and communication by it’s only interaction with the setup $\bar{\mathcal{G}}^{10}$. We call a setup (i.e. a CP shared functionality) $\bar{\mathcal{G}}$ *GUC-AuthComplete* if in the $[\bar{\mathcal{G}}]$ -hybrid world:

1. There exists a setup off-line protocol which GUC realizes authenticated channels from insecure channels
2. Every well-formed functionality can be GUC securely realized (in the standard GUC model which assumes authenticated channels) by a setup off-line protocol.

We note that, assuming static adversaries,¹¹ one such setup is the Key-Registration with Knowledge (KRK) functionality of [10,14] when viewed as a CP shared functionality (we refer to the last paragraph of the current section for details). In particular the results of [14] imply Property (1) and the results of [10] imply Property (2). For the case of adaptive adversaries, one needs to use a setup which is stronger than KRK; as demonstrated in [14], a sufficient setup in this case is their so called Key-exchange functionality.

¹⁰ All known protocols (in particular the protocols of [10,14]) are of this form.

¹¹ A static adversary chooses the parties to corrupt at the beginning of the protocol.

Ideally, we would like to state our feasibility result for *any* (albeit efficient) functionality \mathcal{F} . More realistically we require that \mathcal{F} have the (standard) properties sketches above, i.e., be well-formed and aborting.

Theorem 2 (General Feasibility Theorem). *Let $\bar{\mathcal{G}}$ be a GUC-AuthComplete CP setup. Then there exists a programmable resource $M = \{M_{\mathcal{F}}\}$ such that for every well-formed aborting functionality \mathcal{F} there exists a protocol π which CP-realizes $\hat{\mathcal{F}}$ with GUC fallback in the $M_{\mathcal{F}}$ -hybrid model.*

We prove the theorem constructively, i.e., by describing an efficient compiler mapping a given aborting well-formed functionality \mathcal{F} to a protocol $CP(\pi)$ and parameters for resource M . Note that although we have assumed that $\hat{\mathcal{F}}$ is output-synchronized, the GUC property holds, for the same protocol $CP(\pi)$, even for the non-synchronized functionality, i.e., the one that results by removing from $\hat{\mathcal{F}}$ the synchronizing shell. The reason is that in the GUC-fallback setting the simulators can synchronize output generation by using the insecure channels R_{ins}^* . The proof of the theorem proceeds in two steps: (1) In the “bootstrapping” step (Lemma 1) we show how to obtain from a GUC-AuthComplete setup $\bar{\mathcal{G}}$, a setup off-line protocol π which CP realizes \mathcal{F} using insecure channels. (2) Then, in the “adding fallback” step (Lemma 2), we show how to compile π into protocol $CP(\pi)$ and resource $M_{\mathcal{F}}$ which CP realize $\hat{\mathcal{F}}$ with GUC fallback.

Lemma 1. *Let $\bar{\mathcal{G}}$ be a GUC-AuthComplete CP setup. Then for every well-formed aborting CP functionality \mathcal{F} there exist a setup off-line protocol π such that $\{\bar{\mathcal{G}}, R_{ins}, \mathcal{F}\} \sqsubseteq_{\pi}^{CP} \{\bar{\mathcal{G}}, R_{ins}\}$.*

The proof is simple and essentially verifies that the conditions for GUC-to-CP translation (Section 2) are met; for details we refer to [4]. The bulk of the work for proving Theorem 2 lies in proving the following lemma which states that if a protocol exists that CP realizes \mathcal{F} from insecure channels then there exists a protocol and resource which additionally have GUC fallback. Together with the previous lemma the theorem follows directly.

Lemma 2. *Let $\bar{\mathcal{G}}$ be a GUC-AuthComplete CP setup, let \mathcal{F} be a well-formed aborting functionality, and let π be a setup off-line protocol such that $\{\bar{\mathcal{G}}, R_{ins}, \mathcal{F}\} \sqsubseteq_{\pi}^{CP} \{\bar{\mathcal{G}}, R_{ins}\}$. Then there exists an efficient resource $M_{\mathcal{F}}$ (the “mediator”) and protocol $CP(\pi)$ such that $CP(\pi)$ CP realizes $\hat{\mathcal{F}}$ with GUC fallback in the $M_{\mathcal{F}}$ -hybrid model.*

Proof idea. The proof is constructive, i.e., we show how to construct $CP(\pi)$ and $M_{\mathcal{F}}$. As a starting point we begin with the ideas of the protocol of [3]. We simplify it both in terms of exposition (crystallizing the underlying technique of assisted emulation) and by removing the need for emulation of broadcast. Then we adapt the resulting protocol such that the (rather minimal) synchronization wrapper suffices for full simulation (an issue which does not arise in the stand-alone setting of [3]). This latter step is done by minimizing the amount of synchronization

the protocol affords players by adding explicit instructions for handling dummy steps. The result being that adversaries remain completely unaware of which round they are in even in an asynchronous environment where they may be activated many times (or just once) during any given round. On the highest level the idea is to have the mediator $M_{\mathcal{F}}$ emulate an execution π “in its head” such that the players are oblivious to everything but their input and output of π . Intuitively this guarantees the CP realization property of Definition 2.

To obtain the GUC fallback property: For each $i \in [n]$ the state of the emulated π_i is *shared* between P_i and $M_{\mathcal{F}}$ such that $M_{\mathcal{F}}$ can not alter it without the help of P_i yet both parties learn nothing about the actual value of the state. For this purpose we describe a pair of 2-party SFE’s run between P_i and $M_{\mathcal{F}}$ which allow for state of π_i to be updated as dictated by an honest execution of π . Intuitively it is the hiding of the states from $M_{\mathcal{F}}$ and the security of the SFE’s which ensure GUC fallback. While enjoying a significantly stronger security notion, our compiler is also conceptually simpler than the one in [3]. This stems from our assumption that the input protocol π operates over a *network of insecure channels* rather than the broadcast channel used in [3]. As a result, (1) our compiler does not need to worry about the parties authenticating their messages, as this is taken care of by π , and (2) we do not need specifically describe a “mediated” broadcast protocol as in [3]. We refer to the full version of this paper for a detailed description of the protocol construction and the proof of security.

5.1 A Concrete Instance

Thus far the results have been stated for an abstract GUC-AuthComplete setup. For a concrete instance we can use the Key Registration with Knowledge (KRK) setup of [10,14] (assuming a static adversary). Recall the KRK functionality (henceforth, GUC-KRK) allows the (monolithic) adversary to register and/or ask for the key of any corrupt player. We modify this to obtain the CP setup KRK such that the i^{th} adversary is allowed only to register and ask for the keys of the i^{th} player. One can verify that any protocol which is GUC secure in the original GUC-KRK hybrid world, is also GUC secure when using setup [KRK].¹² Moreover all protocols of [10,14] using GUC-KRK are setup off-line. In [14] the GUC-KRK is used to register public keys which allow for non-interactive key agreement. Such public/secret key pairs can easily be constructed based on the Decisional Diffie-Hellman (DDH) assumption. Therefore we obtain the following corollary.

Corollary 1. *If the DDH assumption holds, there exist (efficient) setup $\bar{\mathcal{G}}$ and a programmable resource $M = \{M_x\}_{x \in \{0,1\}^*}$ such that for every well-formed aborting functionality \mathcal{F} , there exists a protocol π which CP-realizes $\hat{\mathcal{F}}$ with GUC fallback in the $\{\mathcal{G}, M_{\mathcal{F}}\}$ -hybrid model.*

¹² The only difference is that when the (monolithic) adversary makes a registration or secret-key request for some party to [KRK], it needs to append the ID of this party to the message. But this has no effect on the protocol.

6 Implications for Mechanism Design

In this section we translate our results into the language of game theory and interpret them in terms of reducing trust in mechanisms for games played in a computational setting.

Concurrent games and reducing trust in mechanisms. On the highest level we describe a setting where multiple (rational) players are involved in several concurrent games. As a guiding principle for this concurrent model of game-play we hold that *while actions are performed locally to a given game, the intentions and results should be interpreted globally*. This is done to reflect the real world consideration that “winning” in one game may be unimportant to a given player if a concurrent game is “lost” (or more generally lost by a fellow player).¹³ Moreover strategies of concurrent games may depend on each other in a very real sense.¹⁴

A bit more formally, we generalize the powerful mediated games model of [21]. Here game play involves message exchanges between the players and a trusted mediator i.e. the mechanism for that game. In line with our guiding principle we define actions as being local to a particular game. For example placing a bid in one auction is an independent event from say casting a vote in some concurrent election. However, strategies (i.e. the decision how much to bid and for whom to vote) as well as utility functions (i.e. say the cost of losing the auction but winning the election) are defined over all games. This leads to a natural generalization of the equilibria notion(s) spanning concurrent games.

The goal of this section is to provide tools to simplify the task of mechanism design in a *concurrent computational world where trusted entities are a rare commodity*. To that end the tools we develop allow for the following:

- The mechanism may be described independently of the setting in which it will be used. In particular they are defined by a function mapping local actions to their local outputs.
- The mechanism may be designed and analyzed under the assumption it can be completely trusted.
- Yet the resulting (ideal) game can instead be played using a protocol and special network resource (in place of the mechanism) which can be implemented by computers such that:
 - The resource requires significantly less trust than the original mechanism. In particular it is only asked to enforce isolation. But it is not required to ensure that outcomes are computed correctly nor must it maintain any kind of privacy on behalf of the players. Instead these properties are obtained “for free”.

¹³ A telecom company that wins an auction for acquiring a patent for a new wireless technology may care less about this result if concurrently it lost the bid to use the wireless spectrum in which it was planning on implementing the technology.

¹⁴ A company bidding concurrently for multiple add-spaces may be willing to bid less in the first auction as its bid in the second auction increases.

- Nevertheless, using the resource is equivalent to using the original mechanism in a very strong game-theoretic sense. In other words it does not matter to *this or any concurrent* game whether the ideal local mechanism is used as described by the mechanism designer or the protocol and resource are used. To justify this claim we show that (in particular) many types of interesting (computational) equilibria are maintained between the two settings. Moreover, finding an equivalent strategy can be done locally by each player with no interaction or external help.

Overview of results. In light of space constraints we refer the interested reader to the full version [4] for a formal treatise of these results. There we formalize our model of concurrent games and define a notion of isomorphic game sets. We note that the equivalence is both powerful and constructive. An isomorphism comes equipped with a pair of efficient and locally computable function for each player. These functions map from actions in Γ to actions in G (and vice versa) such that any payoff induced by the actions remains essentially unchanged in the new game. In particular we show that these mappings preserve computational Nash Equilibria [27] (cNE). In other words (the actions of) any cNE of Γ is mapped to (the actions of) a cNE in G inducing essentially the same payoffs. We go on to sketch how to extend this result to the computational versions of k -resilient equilibria [7], correlated equilibria [6], t -immune equilibria [2] and dominant strategies.

Next, we show a central theorem tying the notion of CP realization to concurrent games.

Adversary-oblivious functionalities. There is an important subtlety which must be addressed for such an application of CP-realization to go through. We view rational parties as taking the role of the adversaries (rather than the honest parties) in the CP setting. More technically to use CP-realization in a game theoretic setting we wish to preserve *adversarial* power, not just that of the honest players. In one direction (from the real to the ideal) we are given such power-preservation essentially for free by the CP-realization notion. However, in the other direction, in general, CP-realizing protocols and resources only guarantee preservation of the honest interfaces. This means that the strategy of an ideal rational player (making non-trivial use of their adversarial interface) may not have an equivalent strategy in the real world. For this reason we make the following definition restricting the type of resources about which we can use CP-realization to make game theoretic equivalence statements.

Definition 3 (informal). *An n -player CP functionality (with $2n$ interfaces) is called adversary oblivious if the only permitted input/output on any adversarial interface can be locally simulated given access only to the corresponding honest interface.*

Put simply an adversary oblivious functionality provides no additional (non-trivial) power to an adversary controlling both interfaces compared to one

controlling only the honest player’s interface.¹⁵ Intuitively we require this restriction to guarantee that equilibria are preserved their stability when mapped to between game sets.

We can now (informally) state the central theorem tying CP-realization to mechanism design the proof of which can be found in the full version. In a nutshell it states that a resource which can be used to CP-realize another adversary oblivious resource can be used in its place even in a concurrent game theoretic context.

Theorem 3 (Replacing Mechanisms). *Let \mathcal{M} be an adversary oblivious CP-functionality, π be a protocol and \mathcal{R} be a CP-functionality such that $\mathcal{M} \sqsubseteq^{\text{CP}} \mathcal{R}$. Let Γ be a game set using mechanism \mathcal{M} . Then there exists an isomorphic game set G using \mathcal{R} in place of \mathcal{M} .*

Reducing trust. Finally, we apply the feasibility results for CP realization (Theorem 2) with GUC fallback to provide a concrete construction for reducing trust in concurrent used mechanisms. The GUC fallback satisfied by the construction in Theorem 2 is guaranteed regardless of the behavior of the resource it uses. In other words neither privacy of players actions, nor the correctness of the output computed by the mechanism rely on the honest behavior of the mechanism.

Since all players are rational our CP construction must tolerate full corruption. In this case the best we can hope to achieve is realization “with abort”. In other words we can only apply our construction to mechanisms which permit players to abort.

Well motivated games. We model mechanisms as having a special input \perp signaling the case when a player aborts the game or simply refuses to play. The construction of Theorem 2 permits simulating a mechanism which handles such actions by producing outcome \perp for all players. However, we opt instead to place the reasonable assumption on the utility profile of the game ruling out such behavior as irrational. In particular we call a game set Γ *well motivated* if for all $i \in [n]$ the expected utility of any outcome obtained with an abort is less than the utility of all outcomes obtained without an abort.

Corollary 2 (Replacing Mechanisms). *Let $\bar{\mathcal{G}}$ be a functionality satisfying the conditions of Theorem 2 and let M be the corresponding programmable resource. Let \mathcal{M} be an output synchronized mechanism¹⁶ and c be polynomial in the security parameter. Then for any well motivated game set Γ using (amongst others) mechanism \mathcal{M} there exists an isomorphic game set G using $M_{\mathcal{M}}$ in place of \mathcal{M} such that $M_{\mathcal{M}}$ is not assumed to enforce privacy or compute output correctly.*

¹⁵ Unlike an honest player, an adversary is not restricted to playing the honest (CP) protocol nor using the input (type) provided by the environment.

¹⁶ i.e. an adversary oblivious CP-functionality with the synchronization wrapper applied to it.

Human strategies. We observe that the mappings between strategies arising from the construction in Theorem 2 actually simply compose the original strategy with a function (namely the protocol ITM) which translates actions “on-the-fly”. In the context of humans playing games this has qualitative advantages over simply mapping from one strategy to the other. If actions are translated on the fly irrespective of the strategy used players need not even know their own strategy (let alone be aware of some efficient code implementing it) as long as they can always decide on a next action.

Acknowledgments. The authors thank Ran Canetti and Margarita Vald for their interesting discussions and helpful comments concerning the applications to mechanism design.

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: 25th ACM PODC, pp. 53–62. ACM Press (2006)
2. Abraham, I., Dolev, D., Halpern, J.Y.: Lower Bounds on Implementing Robust and Resilient Mediators. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 302–319. Springer, Heidelberg (2008)
3. Alwen, J., Katz, J., Lindell, Y., Persiano, G., Shelat, A., Visconti, I.: Collusion-Free Multiparty Computation in the Mediated Model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 524–540. Springer, Heidelberg (2009)
4. Alwen, J., Katz, J., Maurer, U., Zikas, V.: Collusion preserving computation. Cryptology ePrint Archive, Report 2011/443 (2011), <http://eprint.iacr.org/2011/433>
5. Alwen, J., Shelat, A., Visconti, I.: Collusion-Free Protocols in the Mediated Model. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 497–514. Springer, Heidelberg (2008)
6. Aumann, R.: Subjectivity and Correlation in Randomized Strategies. Journal of Math. Econ. 1, 67–96 (1974)
7. Aumann, R.J.: Acceptable points in general cooperative n-person games. In: Topics in Mathematical Economics and Game Theory Essays in Honor of Robert J Aumann, vol. 23, pp. 287–324 (1959)
8. Barany, I.: Fair distribution protocols, or how the players replace fortune. Mathematics of Operations Research 17, 327–340 (1992)
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE (2001), Full version at <http://eprint.iacr.org/2000/067/>
10. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Security with Global Setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
11. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
12. Canetti, R., Vald, M.: Universally composable security with local adversaries. Cryptology ePrint Archive, Report 2012/117 (2012), <http://eprint.iacr.org/2012/117>

13. Crawford, V., Sobel, J.: Strategic information transmission. *Econometrica* 50, 1431–1451 (1982)
14. Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and On-Line Deniability of Authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Heidelberg (2009)
15. Forges, F.: Universal mechanisms. *Econometrica* 58, 1342–1364 (1990)
16. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
17. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In: 19th ACM STOC, pp. 218–229. ACM Press (1987)
18. Izmalkov, S., Lepinski, M., Micali, S.: Rational Secure Computation and Ideal Mechanism Design. In: FOCS 2005: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 585–595. IEEE Computer Society, Washington, DC (2005)
19. Izmalkov, S., Lepinski, M., Micali, S.: Verifiably Secure Devices. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 273–301. Springer, Heidelberg (2008)
20. Izmalkov, S., Lepinski, M., Micali, S.: Perfect implementation. *Games and Economic Behavior* 71(1), 121–140 (2011), <http://hdl.handle.net/1721.1/50634>
21. Izmalkov, S., Micali, S., Lepinski, M.: Rational secure computation and ideal mechanism design. In: 46th FOCS, pp. 585–595. IEEE (2005), Full version available at <http://dspace.mit.edu/handle/1721.1/38208>
22. Lepinski, M., Micali, S., Shelat, A.: Collusion-Free Protocols. In: STOC 2005: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 543–552. ACM, New York (2005)
23. Lepinski, M., Micali, S., Peikert, C., Shelat, A.: Completely fair SFE and coalition-safe cheap talk. In: 23rd ACM PODC, pp. 1–10. ACM Press (2004)
24. Lepinski, M., Micali, S., Shelat, A.: Collusion-free protocols. In: 37th ACM STOC, pp. 543–552. ACM Press (2005)
25. Lepinski, M., Micali, S., Shelat, A.: Fair-Zero Knowledge. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 245–263. Springer, Heidelberg (2005)
26. Maurer, U., Renner, R.: Abstract cryptography. In: Innovations in Computer Science. Tsinghua University Press (2011)
27. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, New York (2007)
28. Simmons, G.J.: The prisoners' problem and the subliminal channel. In: Crypto 1983, pp. 51–67. Plenum Press (1984)
29. Simmons, G.J.: Cryptanalysis and protocol failures. *Communications of the ACM* 37(11), 56–65 (1994)
30. Simmons, G.J.: The History of Subliminal Channels. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 237–256. Springer, Heidelberg (1996)

Secret Sharing Schemes for Very Dense Graphs*

Amos Beimel¹, Oriol Farràs², and Yuval Mintz¹

¹ Ben Gurion University of the Negev, Be'er Sheva, Israel

² Universitat Rovira i Virgili, Tarragona, Spain

{beimel,yuvalmin}@cs.bgu.ac.il, oriol.farras@urv.cat

Abstract. A secret-sharing scheme realizes a graph if every two vertices connected by an edge can reconstruct the secret while every independent set in the graph does not get any information on the secret. Similar to secret-sharing schemes for general access structures, there are gaps between the known lower bounds and upper bounds on the share size for graphs. Motivated by the question of what makes a graph “hard” for secret-sharing schemes, we study very dense graphs, that is, graphs whose complement contains few edges. We show that if a graph with n vertices contains $\binom{n}{2} - n^{1+\beta}$ edges for some constant $0 \leq \beta < 1$, then there is a scheme realizing the graph with total share size of $\tilde{O}(n^{5/4+3\beta/4})$. This should be compared to $O(n^2/\log n)$ – the best upper bound known for general graphs. Thus, if a graph is “hard”, then the graph and its complement should have many edges. We generalize these results to nearly complete k -homogeneous access structures for a constant k . To complement our results, we prove lower bounds for secret-sharing schemes realizing very dense graphs, e.g., for linear secret-sharing schemes we prove a lower bound of $\Omega(n^{1+\beta/2})$.

1 Introduction

A secret-sharing scheme, introduced by [9,43,31], is a method by which a dealer, which holds a secret string, can distribute strings, called shares, to a set of participants, enabling only predefined subsets of participants to reconstruct the secret from their shares. The collection of predefined subsets authorized to reconstruct the secret is called the access structure. We consider perfect schemes, in which any unauthorized set of participants should learn nothing about the secret from their combined shares. Secret-sharing schemes are useful cryptographic building blocks, used in many secure protocols, e.g., multiparty computation [7,17,19], threshold cryptography [24], access control [39], attribute-based encryption [30,51], and oblivious transfer [44,50].

For a scheme to be efficient and be useful for the above mentioned applications, the size of the shares should be small (i.e., polynomial in the number of participants). There are access structures that have efficient schemes, e.g., the threshold access structure, in which the authorized sets are all sets containing at least ℓ participants (for some threshold ℓ) [9,43]. For every access structure

* This work was supported by ISF grant 938/09.

there exist secret-sharing schemes realizing it [31]. However, the best known schemes for general access structures, e.g., [8,45,13,34], are highly inefficient, that is, for most access structures the size of shares is $2^{O(n)}$, where n is the number of parties in the access structure. The best lower bound known on the total share size for an explicit or implicit access structure is $\Omega(n^2/\log n)$. Thus, there exists a large gap between the known upper and lower bounds. Bridging this gap is one of the most important questions in the study of secret-sharing schemes. We lack sufficient methods for proving lower bounds on the share size. Furthermore, we lack the sufficient understanding of which access structures are “hard”, that is, which access structures require large shares (if any). In contrast to general secret-sharing schemes, super-polynomial lower bounds are known for *linear* secret-sharing schemes, that is, for schemes where the shares are generated using a linear transformation. It was proved that there exists an explicit access structure such that the total share size of any linear secret-sharing scheme realizing it is $n^{\Omega(\log n)}$ [3,28,29]. Linear secret-sharing schemes are important as most known secret-sharing schemes are linear and many cryptographic applications require that the scheme is linear.

In this paper we consider a special family of access structures, in which all minimal authorized sets are of size 2. These access structures can be described by a graph, where each participant is represented by a vertex and each minimal authorized set is represented by an edge. Graph access structures are useful and interesting and have been studied in, e.g., [10,12,14,21,22,23,25,37,47,49]. Many of the results found for graph access structures, using graph theory, were later extended to apply to all access structure. This is illustrated by the next example.

Example 1. Blundo et al. [12] proved that the share size of graph access structures is either the size of the secret or at least 1.5 times larger than that size. This was generalized later to many other families of access structures. Martí-Farré and Padró [38] proved that the share size of every access structure that is not *matroidal* is at least 1.5 times larger than the size of the secret.

Other results on graph access structures have been extended to homogeneous access structures [36,41,46], which are access structures whose minimal authorized subsets are of the same size, and access structures described by simple hypergraphs [20,48].

Every graph access structure can be realized by a scheme in which the total share size is $O(n^2/\log n)$ [15,11,26]; this scheme is linear. The best lower bound for the total share size required to realize a graph access structure by a general secret-sharing scheme is $\Omega(n \log n)$ [25,10,21]. The best lower bound for the total share size required to realize a graph access structure by a linear secret-sharing scheme is $\Omega(n^{3/2})$ [6]. Although the gap between the lower and upper bounds is smaller than that of general access structures, studying this gap might reveal new insight that could be applied to the share size of general access structures.

Our Results. In this work we study a natural family of graphs – very dense graphs. These are graphs that have $\binom{n}{2} - \ell$ edges for $\ell \ll n^2$ (where n is the number of vertices in the graph). The motivation for this work is trying to

understand which graphs are “hard”, that is, which graphs require total share size of $\Omega(n^2/\text{polylog } n)$ (if any). For example, if a graph contains ℓ edges, then it can be realized by a trivial secret-sharing in which the total share size is 2ℓ times the size of the secret [31]. Thus, if there exists a “hard” graph then it has to have $\Omega(n^2/\text{polylog } n)$ edges. We are interested in the question if these “hard” graphs can be very dense. Our results show that this is not possible.

Our main result is that if a graph has $\binom{n}{2} - n^{1+\beta}$ edges for some $0 \leq \beta \leq 1$, then it can be realized by a secret-sharing scheme in which the total share size is $\tilde{O}(n^{5/4+3\beta/4})$,¹ this scheme is linear. In particular, if β is a constant smaller than 1, the total share size is $\ll n^2$, that is, these are not “hard” graphs as discussed above. Similarly, if $\beta < 1/3$, then the share size is $o(n^{3/2})$; thus, these graphs are easier than the graphs for which [6] proved their lower bounds for linear secret-sharing schemes. As a corollary of our main result we prove that if a graph has $\binom{n}{2} - \ell$ edges, where $\ell < n/2$, then it can be realized by a scheme in which the share size is $n + O(\ell^{5/4})$. Thus, if $\ell \ll n^{4/5}$, then the total share size is $n + o(n)$, which is optimal up to an additive factor of $o(n)$.

We extend the techniques used in this result to the study of two additional problems. First, we consider the following scenario: we start with a graph and remove few edges from it. The question is how much the share size of a secret-sharing scheme realizing the graph can grow as a result of the removed edges. If we add edges, then trivially the share size grows at most linearly in the number of added edges. We show that also when removing edges, the share size does not increase too much. We study this problem also for general access structures, considering the removal of minimal authorized subsets for any access structure. We show that for certain access structures the share size does not increase too much either. Second, we study the removal of ℓ minimal authorized subsets from k -out-of- n threshold access structures. We present a construction with total share size $\tilde{O}(\ell n)$ for $k \ll n$.

To complement our results, we prove lower bounds on the share size of secret-sharing schemes realizing very dense graphs. For graph access structures, the known lower bounds for general secret-sharing schemes [25,10,21] and linear secret-sharing schemes [6] use sparse graphs with $\theta(n \log n)$ edges and $\theta(n^{3/2})$ edges, respectively. Using the above lower bounds, we prove lower bounds of $\Omega(\beta n \log n)$ and $\Omega(n^{1/2+\beta/2})$ for general and linear secret-sharing schemes respectively for graphs with $\binom{n}{2} - n^{1+\beta}$ edges. In addition, we prove lower bounds of $n + \ell$ for graphs with $\binom{n}{2} - \ell$ edges, where $\ell < n/2$. Our lower bounds are not tight, however, they prove, as can be expected, that for linear secret-sharing schemes the total share size grows as a function of the number of excluded edges. The lower bounds for linear schemes are interesting as most known secret-sharing schemes, including the schemes constructed in this paper, are linear.

Techniques. Brickell and Davenport [14] proved that a connected graph has an ideal scheme (that is, a scheme in which the total share size is n times the size

¹ We use the \tilde{O} notation which ignores polylogarithmic factors.

of the secret) if and only if the graph is a complete multipartite graph.² To construct a scheme realizing a very dense graph, we cover the graph by complete multipartite graphs (in particular, cliques), that is, we construct a sequence of multipartite graphs G_1, G_2, \dots, G_r such that each graph G_i is a subgraph of G and each edge of G is an edge in at least one graph G_i . We next, for every i , share the secret independently using the ideal secret-sharing scheme realizing G_i . The total share size in the resulting scheme is the sum of the number of vertices in the graphs G_1, G_2, \dots, G_r . This idea of covering a graph was used in previous schemes, e.g., [11,12]. The contribution of this paper is how to find a “good” cover for every dense graph.

Our starting point is constructing a scheme for graphs in which every vertex is adjacent to nearly all other vertices, that is, graphs where the degree of every vertex in the complement graph is bounded by some $d \ll n$. We cover such graphs by equivalence graphs, that is, graphs which are union of disjoint cliques. Alon [1] proved, using a probabilistic proof, that every such graph can be covered by $O(d^2 \log n)$ equivalence graphs. We improve on this result, and prove, using a different probabilistic proof, that every such graph can be covered by $O(d \log n)$ equivalence graphs. The total share size of the resulting scheme is $\tilde{O}(dn)$.

We use the above scheme to realize very dense graphs. We first cover all vertices whose degree in the complement graph is “big”. There are not too many such vertices in the complement graph, and the share size in realizing each star (namely, a vertex and its adjacent edges) is at most n . Once we removed all edges adjacent to vertices whose degree is “big”, we use the cover by equivalence graphs to cover the remaining edges. To achieve a better scheme, we first remove vertices of high degree using stars, then use covers of bipartite graphs of [33] to further reduce the degree of the vertices in the complement graph, and finally use the cover by equivalence graphs.

Additional Related Work. Sun and Shieh [48] consider access structures that are defined by a *forbidden graph*, where each party is represented by a vertex, and 2 parties are an unauthorized set iff their vertices are connected by an edge. They give a construction which had an information ratio of $n/2$. In [48], every set of size 3 can reconstruct the secret. Our problem is much harder as every independent set in the graph is unauthorized.

2 Preliminaries

In this section we define secret-sharing schemes and provide some background material used in this work. We present a definition of secret-sharing as given in [18,5].

Definition 2. Let $P = \{p_1, \dots, p_n\}$ be a set of parties. A collection $\Gamma \subseteq 2^P$ is monotone if $B \in \Gamma$ and $B \subseteq C$ imply that $C \in \Gamma$. An access structure is a monotone collection $\Gamma \subseteq 2^P$ of non-empty subsets of P . Sets in Γ are called

² See Section 2 for the graph terminology used in the rest of this section.

authorized, and sets not in Γ are called unauthorized. The family of minimal authorized subsets is noted by $\min \Gamma$.

A distribution scheme $\Sigma = \langle \Pi, \mu \rangle$ with domain of secrets K is a pair, where μ is a probability distribution on some finite set R called the set of random strings and Π is a mapping from $K \times R$ to a set of n -tuples $K_1 \times K_2 \times \cdots \times K_n$, where K_j is called the domain of shares of p_j . A dealer distributes a secret $k \in K$ according to Σ by first sampling a random string $r \in R$ according to μ , computing a vector of shares $\Pi(k, r) = (s_1, \dots, s_n)$, and privately communicating each share s_j to party p_j . For a set $A \subseteq P$, we denote $\Pi(s, r)_A$ as the restriction of $\Pi(s, r)$ to its A -entries. The (normalized) total share size of a distribution scheme is $\sum_{1 \leq j \leq n} \log |K_j| / \log |K|$.

Definition 3 (Secret Sharing). Let K be a finite set of secrets, where $|K| \geq 2$. A distribution scheme $\langle \Pi, \mu \rangle$ with domain of secrets K is a secret-sharing scheme realizing an access structure Γ if the following two requirements hold:

CORRECTNESS. The secret k can be reconstructed by any authorized set of parties.

PRIVACY. Every unauthorized set cannot learn anything about the secret (in the information theoretic sense) from their shares.

In this work we mainly consider graph access structures. Let $G = (V, E)$ be an undirected graph. We consider the graph access structure, where the parties are the vertices of the graph and the minimal authorized sets are the edges. In other words, a set of vertices can reconstruct the secret iff it contains an edge. In the rest of the paper we will not distinguish between the graph and the access structure it describes and we will not distinguish between vertices and parties.

Next we define the graph terminology that we use throughout this paper. The *degree* of a graph is the maximum degree of vertices in a graph. A graph $G' = (V', E')$ is a *subgraph* of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. A k -partite graph $G = (V_1, \dots, V_k, E)$, where V_1, \dots, V_k are disjoint, is a graph whose vertices are $\cup_{i=1}^k V_k$ such that if $(u, v) \in E$, then there are $i \neq j$ such that $u \in V_i$ and $v \in V_j$ (that is, there are edges only between vertices in different parts). A k -partite graph is *complete* if it contains all edges between vertices in different parts. A graph is a *multipartite* graph if it is k -partite for some k . For example, a clique is a complete k -partite graph, where k is the number of vertices in the clique. A bipartite graph in which $|V_1| = 1$ is called a *star*; the vertex in V_1 is the *center* and the ones in V_2 are the *leaves*.

Brickell and Davenport [14] proved that a connected graph can be realized by an ideal scheme (that is, by a scheme with total share size n) iff the graph is a complete multipartite graph. As we use the ideal scheme for multipartite graphs we describe it below.

Theorem 4 ([14]). Let $G = (V_1, \dots, V_k, E)$ be a complete multipartite graph and $p > k$ be a prime. There is a linear secret-sharing realizing G where the domain of secrets and the domain of shares of each party are $\{0, \dots, p - 1\}$.

Proof. Let $s \in \{0, \dots, p-1\}$ be the secret. We first generate shares in Shamir's 2-out-of- k scheme [43] for the secret s . That is, we choose $a \in \{0, \dots, p-1\}$ at random with uniform distribution and we compute the share $s_i = a \cdot i + s \bmod p$ for $1 \leq i \leq k$. Next, we give s_i to all vertices in V_i . Two vertices from different parts, say V_i and V_j , can reconstruct the secret as follows: $s = (js_i - is_j)/(j-i)$ (where the arithmetic is in \mathbb{F}_p – the finite field with p elements). \square

Remark 5. The total share size in the above scheme is n . However, it requires that $p > k$. In the rest of the paper we assume that $p > n$, thus, we can realize every multipartite subgraph of a graph G with n vertices. This is a reasonable requirement that assumes that the number of bits in the secret is at least $\log n$. We will not mention the size of the secret in the rest of the paper and only consider the total share size of the scheme.

In the rest of the paper we will construct schemes, where we choose subgraphs of G which are multipartite, and share the secret s independently for each subgraph. The following is a well-known lemma.

Lemma 6. *Let $G = (V, E)$ be a graph and $G_1 = (V_1, E_1), \dots, G_r = (V_r, E_r)$ be subgraphs of G such that each G_i is a complete multipartite graph and $E = \cup_{i=1}^r E_i$ (that is, G_1, \dots, G_r cover G). Assume that we share a secret s independently for each G_i using the multipartite scheme. Then, the resulting scheme realizes G .*

Description of the Problem. In this work we study the problem of realizing a graph access structure, where the graph has few excluded edges. Specifically, let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = \binom{n}{2} - \ell$ for some $0 < \ell < \binom{n}{2}$. We consider the complement graph $\overline{G} = (V, \overline{E})$, where $e \in \overline{E}$ iff $e \notin E$. We call \overline{G} the *excluded graph* and call its edges the *excluded edges*. In the rest of the paper, the excluded graph \overline{G} is a sparse graph with $\ll \binom{n}{2}$ edges.

Example 7. Assume $\ell = 1$, that is, there is one excluded edge, say (v_{n-1}, v_n) . In this case, the graph can be realized by an ideal scheme as the graph is the complete $(n-1)$ -partite graph, where v_{n-1}, v_n are in the same part.

Example 8. Assume $\ell = 2$, and there are two adjacent excluded edges, say (v_{n-2}, v_n) and (v_{n-1}, v_n) . In this case, the graph G is not a complete multipartite graph, hence it cannot be realized by an ideal scheme [14]. However, it can be realized by a scheme in which each of the parties v_1, \dots, v_{n-3}, v_n gets a share whose size is the size of the secret and v_{n-2}, v_{n-1} get a share whose size is twice the size of the secret. The scheme is as follows: Generate shares according to the Shamir's 2-out-of- $(n-2)$ secret-sharing scheme, and give party v_i the i th share in Shamir's scheme for $1 \leq i \leq n-2$. In addition give to v_{n-1} and v_n the $(n-2)$ th share in Shamir's scheme. Using the above shares every pair of parties, except for pairs contained in $\{v_{n-2}, v_{n-1}, v_n\}$, can reconstruct the secret. As the only authorized pair in $\{v_{n-2}, v_{n-1}, v_n\}$ is (v_{n-2}, v_{n-1}) , we give them additional shares: we choose two random strings r_1 and r_2 whose exclusive-or is the secret, and give r_1 to v_{n-2} and r_2 to v_{n-1} . By [11], the total size of shares to realize G is at least $n+2$. That is, the above scheme is optimal.

3 Constructions for Bounded Degree Excluded Graphs

If the excluded graph contains few edges, then the average degree of its vertices is small. We first construct a scheme for graphs such that the degree of all vertices in its excluded graph is bounded by some d . In Section 4 we show how we can use this construction for any graph with few excluded edges.

The construction of a secret-sharing scheme for a graph G whose excluded graph \overline{G} has bounded degree uses a cover of G by cliques such that each vertex is contained in a relatively small number of cliques. This is useful as cliques have an ideal scheme. To construct this cover we use colorings of the excluded graph.

Definition 9. An equivalence graph is a vertex-disjoint union of cliques. An equivalence cover of $G = (V, E)$ is a collection of equivalence graphs $G_1 = (V, E_1), \dots, G_r = (V, E_r)$, each of them is a subgraph of G (that is, $E_i \subseteq E$), such that this collection covers all the edges of G (that is, $\cup_{1 \leq i \leq r} E_i = E$).

A coloring of a graph $\overline{G} = (V, \overline{E})$ with c colors is a mapping $\mu : V \rightarrow \{1, \dots, c\}$ such that $\mu(u) \neq \mu(v)$ for every $(u, v) \in \overline{E}$.

Lemma 10. Let $G = (V, E)$ be a graph such that the degree of every vertex in its excluded graph \overline{G} is at most d . Then there exists an equivalence cover of G with $r = 16d \ln n$ equivalence graphs.

Proof. An equivalence cover of G can be described by a coloring of \overline{G} and vice versa: given a coloring μ of \overline{G} we construct an equivalence graph $G' = (V, E')$, which is a subgraph of G , where two vertices in G' are connected if they are colored by the same color, that is, $E' = \{(u, v) : \mu(u) = \mu(v)\}$. For every color, the set of vertices colored by such color is an independent set in \overline{G} , hence a clique in G .

The existence of an equivalence cover of G of size r is proved by using the *probabilistic method* (see, e.g., [2]). We choose r random colorings μ_1, \dots, μ_r of \overline{G} with $4d$ colors. That is, each coloring is chosen independently with uniform distribution among all colorings of \overline{G} with $4d$ colors. For every coloring μ_i , we consider the equivalence graph G_i as described above. We next prove that with probability at least half G_1, \dots, G_r is an equivalence cover of G .

Let $(u, v) \in E$. We first fix i and compute the probability that u and v have the same color in the random coloring μ_i . Fix an arbitrary coloring of all vertices except for u and v . We prove that conditioned on this coloring, the probability that u and v are colored in the same color is at least $1/(8d)$: The number of colors not used by the neighbors of u and v is at least $2d$, thus, the probability that u is colored by such color is at least half, and the probability that in this case v is colored in the same color as u is at least $1/(4d)$. That is, with probability at least $1/(8d)$, the edge (u, v) is covered by the graph G_i .

The probability that an edge (u, v) is not covered by the r random equivalence graphs G_1, \dots, G_r is at most $(1 - 1/(8d))^r \leq e^{-r/8d} = 1/n^2$. Thus, the probability that there exists an edge $(u, v) \in E$ that is not covered by the r random equivalence graphs G_1, \dots, G_r is at most $\binom{n}{2}/n^2 < 1/2$. In particular, such cover with r equivalence graphs exists. \square

Remark 11. The existence of the equivalence cover in Lemma 10 is not constructive as we need to choose a random coloring of a graph of bounded degree. Such coloring can be chosen with nearly uniform distribution in polynomial time using a Markov process [32,42]. Given a collection of equivalence graphs, it is easy to check that for every edge $(u, v) \in E$ there is at least one graph in the collection that covers (u, v) . If this is not the case we repeat the process of choosing r random colorings until we find a good collection. The expected number of collections of colorings that have to be chosen before finding a good one is $O(1)$. Thus, we get a randomized polynomial-time algorithm to construct the equivalence cover.

Alon [1] observed that the size of the smallest equivalence cover of a graph G is smaller than the smallest clique cover of G . He further proved that if the degree of every vertex in \overline{G} is at most d , then G can be covered by $O(d^2 \ln n)$ cliques. We directly analyze the size of the smallest equivalence cover and get an equivalence cover of size $O(d \ln n)$. To the best of our knowledge such bound was not known prior to our work.

Lemma 12. *Let $G = (V, E)$ be a graph such that the maximum vertex degree in $\overline{G} = (V, \overline{E})$ is less or equal to d . Then, G can be realized by a secret-sharing scheme in which the total share size is $\tilde{O}(nd)$.*

Proof. Consider a collection of $r = 16d \ln n$ equivalence graphs that cover G (as guaranteed by Lemma 10). We realize the access structure of each equivalence graph G_i in the collection by an ideal scheme: For every clique C in G_i , generate shares in Shamir's 2-out-of- $|C|$ secret-sharing scheme, and distribute the shares among the parties of C .

For every excluded edge $(u, v) \notin E$, the vertices u and v are in different cliques in each G_i (as G_i is a subgraph of G). Thus, in the above scheme u and v do not get any information. On the other hand, every edge $(u, v) \in E$ is covered by at least one graph G_i , that is, u and v are in the clique in G_i , thus, u and v can reconstruct the secret. As in each graph G_i each party gets one share, the total share size of the resulting scheme is $nr = O(dn \ln n) = \tilde{O}(nd)$. \square

We can save a factor of $O(\ln n)$ by using an equivalence cover of size $O(d \ln n)$ such that each edge $(u, v) \in E$ is covered by $O(\ln n)$ graphs in the cover (the existence of such cover can be proved by the same arguments as in the proof of Lemma 10 using a Chernoff bound). We then use Stinson decomposition techniques [47], to construct a scheme with total share size $O(nd)$. The details will be explained in the full version of the paper.

3.1 Constructions for Bipartite Graphs with Bounded Degree

As a step in constructing a secret-sharing scheme realizing a graph with few excluded edges, we will need to realize certain bipartite graphs. In this section we show how to realize them using bipartite covers.

Definition 13 (Complete-bipartite cover and bipartite complement). Let $H = (U, V, E)$ be a bipartite graph. A complete-bipartite cover of $H = (U, V, E)$ is a collection of complete bipartite graphs $H_1 = (U_1, V_1, E_1), \dots, H_r = (U_r, V_r, E_r)$ (that is $E_i = U_i \times V_i$), each of them is a subgraph of H , such that this collection covers all the edges of H (that is, $\cup_{1 \leq i \leq r} E_i = E$).

The bipartite complement of a graph H is the bipartite graph $\overline{H} = (U, V, \overline{E})$, where $(u, v) \in \overline{E}$ iff $(u, v) \notin E$ for every $u \in U$ and $v \in V$.

Note that the bipartite complement of a bipartite graph is a bipartite graph and it differs from the complement of the bipartite graph.

Lemma 14 (Jukna [33, Theorem 1]). Let $H = (U, V, E)$ be a bipartite graph such that $|U| \leq |V|$ and the degree of every vertex in V in the bipartite complement graph \overline{H} is at most d . Then there exists a cover of H with $O(d \ln n)$ complete bipartite graphs, where $|V| = n$.

Lemma 15. Let $d < n$ and $H = (U, V, E)$ be a bipartite graph such that $|U| = k$, $|V| = n \geq k$, and the degree of every vertex in U in \overline{H} is at most d . Then, H can be realized by a secret-sharing scheme in which the total share size is $\tilde{O}(n + k^{3/2}d)$. If $k = (n/d)^{2/3}$, the total share size is $\tilde{O}(n)$.

Proof. Let $D = \{v \in V : \exists_{u \in U} \text{ such that } (u, v) \in \overline{E}\}$. As the degree of every vertex in U in \overline{H} is at most d , the size of D is at most dk . Furthermore, the complete bipartite graph $H_1 = (U, V \setminus D, U \times (V \setminus D))$ is a subgraph of H . We realize H_1 by an ideal scheme in which the total share size is less than $|U| + |V| = O(n)$.

Now, define $D_2 = \{v \in D : \text{The degree of } v \text{ in } \overline{H} \text{ is at least } \sqrt{k}\}$. As \overline{H} contains at most dk edges, $|D_2| \leq d\sqrt{k}$. Let $H_2 = (U, D_2, E \cap (U \times D_2))$. The number of edges in H_2 is less than $|U||D_2| \leq k^{3/2}d$, thus, we can realize H_2 by secret-sharing scheme in which the total share size is $O(k^{3/2}d)$.

Finally, let $V_3 = D \setminus D_2$ and $H_3 = (U, V_3, E \cap (U \times V_3))$. The degree of each vertex in V_3 in the graph \overline{H}_3 is at most \sqrt{k} , thus, by Lemma 14, H_3 can be covered by $r = O(\sqrt{k} \ln n)$ complete bipartite graphs. We realize each such bipartite graph by an ideal scheme in which the total share size is $|U| + |V_3| \leq k + kd = O(kd)$. Thus, we realize H_2 by a scheme in which the total share size is $O(rkd) = O(k^{3/2}d \ln n)$. As H_1 , H_2 , and H_3 cover H , we constructed a scheme realizing H in which the total share size is $\tilde{O}(n + k^{3/2}d)$. Taking $k = (n/d)^{2/3}$, the total share size is $\tilde{O}(n)$. \square

4 Constructions for Excluded Graph with Few Edges

We next show how to use the schemes of Lemma 12 and Lemma 15 to realize excluded graphs with $\ell = n^{1+\beta}$ edges, where $0 \leq \beta < 1$. We will start with a simple approach and then use more complicated constructions to achieve better upper bounds. We construct our scheme in steps, where in each step: (1) We choose a set of vertices $V' \subseteq V$. (2) We give shares to the parties in V' and the

rest of the parties, such that each edge adjacent to a party in V' can reconstruct the secret, and all other pairs of parties (i.e., unauthorized pairs containing parties in V' and all pairs not adjacent to V') get no information on the secret. (3) We remove the vertices in V' and all their adjacent edges from the graph. We repeat the following step until all vertices in \overline{G} have small degree and then use the equivalence covering scheme of Section 3 to realize the remaining graph. In this process we will ensure that the total share size remains relatively small. In the following, n will always refer to the number of vertices in the original graph.

Our first step is removing all vertices whose degree in \overline{G} is “high”.

Lemma 16. *Let G be a graph such that its excluded graph \overline{G} contains at most $n^{1+\beta}$ edges, where $0 \leq \beta < 1$. Then, for every $d < n$, we can give shares of size $O(n^{2+\beta}/d)$ and remove a set of vertices from G and all adjacent edges and obtain an induced subgraph G' of G such that \overline{G}' contains at most $n^{1+\beta}$ edges and the degree of \overline{G}' is at most d .*

Proof. We choose a vertex v whose degree in \overline{G} is greater than d and consider the star whose center is v and its leaves are all neighbors of v in G . We realize this star using an ideal scheme and remove v and its adjacent edges from G . The total share size in this step is at most n .

We choose another vertex whose degree in \overline{G} is greater than d and do the same until no vertices with degree greater than d exist in \overline{G} . As in the beginning there are $n^{1+\beta}$ edges in \overline{G} and in each step we remove at least d edges from \overline{G} , the number of steps is at most $n^{1+\beta}/d$. Thus, the total share size of the resulting scheme for the removed vertices is $O(nn^{1+\beta}/d)$. \square

We can combine the constructions of Lemma 16 and Lemma 12. That is, we choose some $d \leq n$, remove vertices with degree higher than d in \overline{G} , and then apply the equivalence cover construction to the remaining graph G , where the degree of \overline{G} is d . Thus, the total share size of the resulting scheme (including the scheme from of Lemma 12) is $\tilde{O}(n^{2+\beta}/d + dn)$. To minimize the share size we take $d = \sqrt{n^{1+\beta}}$ and get a scheme in which the total share size is $\tilde{O}(n^{1.5+\beta/2})$.

Using Lemma 16 we decrease the degrees of the vertices in \overline{G} . Instead of applying the construction of Lemma 12 to the resulting graph, we will apply some intermediate steps to further reduce the degree and only then use the construction of Lemma 12.

Lemma 17. *Let $\alpha' < \alpha \leq 1$ such that $\alpha \geq 0.25$ and $G = (V, E)$ be a graph such that the degree of \overline{G} is at most n^α and \overline{G} contains ℓ edges. Then, we can remove a set of vertices and all adjacent edges from the graph and obtain a graph G' such that the degree of \overline{G}' is at most $n^{\alpha'}$, the graph \overline{G}' contains $\ell - \ell'$ excluded edges for some $\ell' > 0$, and the total share size for the removed edges is $\tilde{O}(\ell'n^{1/3+2\alpha/3-\alpha'})$.*

Proof. Let $d = n^\alpha$ and $d' = n^{\alpha'}$. We remove the vertices of degree larger than d' in steps. In each step we choose an arbitrary set F of $k = (n/d)^{2/3}$ vertices of degree at least d' in \overline{G} (if the number of vertices of degree d' is smaller than k , then take the remaining vertices of degree d' and put them in F). Consider all

edges between vertices of F , there are less than $k^2 = n^{4/3}/d^{4/3} \leq n$ such edges (since $d \geq n^{1/4}$). Next consider the bipartite graph $H = (F, V \setminus F, E \cap (F \times (V \setminus F)))$. By Lemma 15, we can realize H with a scheme in which the total share size is $O(n)$. Thus, we can remove the vertices in F and all edges adjacent to them, and the total share size in the scheme for every step is $\tilde{O}(n)$.

Let ℓ' the total number of edges we removed from \overline{G} in these steps until the degree of \overline{G} is at most d' . As each vertex we remove has degree at least d' in \overline{G} , the number of vertices we remove is at most ℓ'/d' . In each step, except for the last, we remove a set F with $(n/d)^{2/3}$ vertices, thus, the number of sets we remove is at most $1 + \ell'/(d'(n/d)^{2/3}) = O(\ell' d^{2/3}/(d' n^{2/3}))$. As in each step the share size is $\tilde{O}(n)$, the total share size for the edges we removed from G is $\tilde{O}(\ell' n^{1/3} d^{2/3}/d') = \tilde{O}(\ell' n^{1/3+2\alpha/3-\alpha'})$. \square

We next show how to construct secret-sharing schemes for graphs with few excluded edges using the three building blocks presented so far: (1) initial degree reductions using stars, (2) $O(\log \log n)$ steps of degree reduction using complete bipartite graphs and stars, and (3) using the equivalence cover construction on the graph with reduced degree.

Theorem 18. *Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = \binom{n}{2} - n^{1+\beta}$ for some $0 \leq \beta < 1$. There exists a secret-sharing scheme realizing G with total share size $\tilde{O}(n^{5/4+3\beta/4})$.*

Proof. Let α_0 be a constant to be determined later. We first apply Lemma 16 with $d = n^{\alpha_0}$ and obtain a graph G such that the degree of \overline{G} is at most d . The total share size in this step is

$$O(n^{2+\beta}/d) = O(n^{2+\beta-\alpha_0}). \quad (1)$$

Next define $\alpha_i = (3 - 2(2/3)^i)\alpha_0 - 2 + 2(2/3)^i$ for $1 \leq i \leq \log \log n$. We choose these constants such that $2\alpha_i/3 - \alpha_{i+1} = 2/3 - \alpha_0$. We next repeatedly apply the degree reduction of Lemma 17; we apply it $\log \log n$ times. In the i th invocation of the lemma, where $0 \leq i < \log \log n$, we take $\alpha = \alpha_i$ and $\alpha' = \alpha_{i+1}$. The cost of each invocation is $\tilde{O}(\ell_i n^{1/3+2\alpha_i/3-\alpha_{i+1}}) = \tilde{O}(\ell_i n^{1-\alpha_0})$, where ℓ_i is the number of edges removed from \overline{G} in the i th invocation. As the number of edges removed in all invocations is at most $n^{1+\beta}$, the total share size in all these invocations is

$$\tilde{O}(n^{1+\beta} n^{1-\alpha_0}) = \tilde{O}(n^{2+\beta-\alpha_0}). \quad (2)$$

After the $\log \log n$ invocations of Lemma 17, the degree of each vertex in \overline{G} is at most $n^{\alpha_{\log \log n}} = O(n^{3\alpha_0-2})$. In the final stage we use Lemma 12 and realize the graph with total share size

$$\tilde{O}(nn^{3\alpha_0-2}) = \tilde{O}(n^{3\alpha_0-1}). \quad (3)$$

The total share of realizing G (by (1), (2), and (3)) is $O(n^{2+\beta-\alpha_0}) + \tilde{O}(n^{2+\beta-\alpha_0}) + \tilde{O}(n^{3\alpha_0-1})$. To minimize this expression, we require that $2 + \beta - \alpha_0 = 3\alpha_0 - 1$, thus, $\alpha_0 = 3/4 + \beta/4$ and the total share size in the scheme is $\tilde{O}(n^{5/4+3\beta/4})$. \square

It can be checked that the construction of the cover of G by multipartite graphs, as done in the above scheme, can be done by a probabilistic algorithm in expected polynomial time. Thus, the computation of the dealer and the parties in our scheme is efficient. In Theorem 18 we showed how to realize a graph where the number of excluded edges is small, however it is at least n . We next show how to realize graphs where the number of excluded edges is less than n .

Corollary 19. *Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = \binom{n}{2} - \ell$ for some $\ell < n/2$. There exists a secret-sharing scheme realizing G with total share size $n + \tilde{O}(\ell^{5/4})$.*

Proof. Let $V' \subseteq V$ be the set of vertices adjacent to excluded edges. As there are ℓ excluded edges, the size of V' is at most 2ℓ . Without loss of generality, let $V = \{v_1, \dots, v_n\}$ and $V' = \{v_t, \dots, v_n\}$ for some $t > n - 2\ell$. We first execute Shamir's 2-out-of- t secret-sharing scheme and give the share s_i to party v_i for $1 \leq i < t$, and give the share s_t to v_i for $t \leq i \leq n$.

Let V'' be such that $V' \subseteq V''$ and $|V''| = 2\ell$. Furthermore, let $G' = (V'', E')$ be the subgraph of G induced by V'' . The graph G' has at most $n' = 2\ell$ vertices and $\ell \leq n'$ excluded edges, thus, by Theorem 18 (with $\beta = 0$), it can be realized by a scheme in which the total share size is $\tilde{O}(\ell^{5/4})$. The total share size in realizing G is, therefore, $n + \tilde{O}(\ell^{5/4})$. \square

5 Constructions for Homogeneous Access Structures

In this section we extend the techniques used in the construction of graph secret-sharing schemes to the construction of schemes for homogeneous access structures, which are access structures whose minimal authorized subsets are of the same size. Every k -homogeneous access structure has a monotone formula of size $O(n^k / \log n)$ (see [52, Theorem 7.3]), thus, by [8], it can be realized by a secret-sharing scheme with total share size $O(n^k / \log n)$. Other upper bounds for hypergraphs are presented in [36, 41, 46, 48]; however they are useful for sparse access structures. In this section, we present constructions for dense homogeneous access structures for a constant k . We will describe these access structures by hypergraphs.

A *hypergraph* is a pair $H = (V, E)$ where V is a set of vertices and $E \subseteq 2^V \setminus \{\emptyset\}$ is the set of *hyperedges*. In this work we consider hypergraphs in which no hyperedge properly contains any other hyperedge. A hypergraph is *k -uniform* if $|e| = k$ for every $e \in E$. A k -uniform hypergraph is *complete* if $E = \binom{V}{k} = \{e \subseteq V : |e| = k\}$. For any k -uniform hypergraph we define the *complement* hypergraph $\bar{H} = (V, \bar{E})$, with $\bar{E} = \binom{V}{k} \setminus E$. Observe that there is a one-to-one correspondence between uniform hypergraphs and homogeneous access structures, and that complete hypergraphs are in correspondence with threshold access structures.

By analogy to graphs, we define an *equivalence* k -hypergraph as a vertex-disjoint union of complete k -uniform hypergraphs, and the equivalence cover of a k -uniform hypergraph $H = (V, E)$ as a collection of equivalence k -hypergraphs

$H_1 = (V, E_1), \dots, H_r = (V, E_r)$ with $E_i \subseteq E$ for $i = 1, \dots, r$ and $\cup_{1 \leq i \leq r} E_i = E$. A *weak* coloring with c colors of a hypergraph $H = (V, E)$ is a mapping $\mu : V \rightarrow \{1, \dots, c\}$ such that for every $e \in E$ there exist $u, v \in e$ with $\mu(u) \neq \mu(v)$.

Lemma 20. *Let $H = (V, E)$ be a k -uniform hypergraph such that the degree of every vertex in its excluded hypergraph is at most d . Then there exists an equivalence cover of H with $r = 2^k k^k d^{k-1} \ln n$ equivalence hypergraphs.*

The proof of this lemma is analogous to the proof of Lemma 10. In this case, the result is obtained by using r weak colorings of \overline{H} with $2kd$ colors.

Lemma 21. *Let $H = (V, E)$ be a k -uniform hypergraph such that the maximum vertex degree of $\overline{H} = (V, \overline{E})$ is less or equal to d . There exists a secret-sharing scheme realizing H in which the total share size is $\tilde{O}(2^k k^k d^{k-1} n)$.*

Proof. Take the equivalence cover of H of size $r = 2^k k^k d^{k-1} \ln n$ guaranteed by Lemma 20. Now we realize each equivalence hypergraph H_i in the collection by an ideal scheme: For every complete hypergraph C in H_i , generate shares in Shamir's k -out-of- $|C|$ secret-sharing scheme. Using arguments similar to the ones used in the proof of Lemma 12, this scheme realizes H and the total share size of the resulting scheme is $nr = \tilde{O}(2^k k^k d^{k-1} n)$. \square

In Theorem 23, we construct a secret-sharing scheme for every excluded hypergraph with few edges. For this purpose, we use a recursive argument based on the construction illustrated in the following example.

Example 22. Let $H = (V, E)$ be a hypergraph and let $v \in V$ be a vertex satisfying that $v \in e$ for every $e \in E$. Consider the hypergraph $H' = (V', E')$ with $V' = V \setminus \{v\}$ and $E' = \{e \setminus \{v\} : e \in E\}$. If there exists a secret-sharing scheme realizing H' with total share size r , then we can construct a scheme realizing H with total share size $r + 1$ as follows. In order to share a secret s , the dealer chooses at random s_1 and s_2 satisfying $s = s_1 + s_2$, sends s_1 to v , and shares s_2 among V' using the scheme realizing H' .

Theorem 23. *Let $H = (V, E)$ be a k -hypergraph with $|V| = n$ and $|E| = \binom{n}{k} - n^{1+\beta}$ for some $0 \leq \beta < k - 1$. There exists a secret-sharing scheme realizing H with total share size $\tilde{O}(2^k k^k n^{2+\beta})$.*

Proof. By induction on k , we prove that for every $H = (V, E)$ satisfying the hypothesis there exists a secret-sharing scheme with total share size $\tilde{O}(2^k k^k \ell^{1-\varepsilon_k} n)$, where $\ell = n^{1+\beta}$ and ε_k is defined by the equation $\varepsilon_{i+1} = \frac{\varepsilon_i}{i+\varepsilon_i}$ and $\varepsilon_1 = 1$. By Theorem 18 this property is satisfied for $k = 2$. Let $H = (V, E)$ be a k -hypergraph with $k > 2$. Define $d = \ell^{\frac{1}{k-1+\varepsilon_{k-1}}}$.

We choose a vertex v adjacent to $\ell_1 > d$ excluded hyperedges. By the hypothesis, there is a secret sharing scheme with total share size $\tilde{O}(2^{k-1}(k-1)^{k-1} \ell_i^{1-\varepsilon_{k-1}} n)$ for the $(k-1)$ -hypergraph $H' = (V', E')$, with $V' = V \setminus \{v\}$ and $E' = \{e \in \binom{V'}{k-1} : e \cup \{v\} \in E\}$. Following the Example 22, we construct a scheme for the sub-hypergraph determined by all hyperedges adjacent to v . Then

we remove v and its adjacent hyperedges from H . We choose another vertex v' adjacent to $\ell_2 > d$ excluded hyperedges and do the same until no vertices with degree greater than d in \overline{H} exist.

Since in the beginning there are ℓ excluded hyperedges, and in each step we remove $\ell_i > d$ hyperedges, the number of steps is at most ℓ/d . Thus, the total share size of the resulting scheme is $\tilde{O}(2^{k-1}(k-1)^{k-1}n \sum_{i=1}^{\ell/d} \ell_i^{1-\varepsilon_{k-1}})$. As $\sum_{i=1}^{\ell/d} \ell_i \leq \ell$, the above expression is maximized when $\ell_1 = \dots = \ell_{\ell/d} = d$, and the total share size of the scheme is $\tilde{O}(2^{k-1}(k-1)^{k-1}n\ell/d^{\varepsilon_{k-1}})$.

Finally, since the degree of \overline{H} at most d , we use Lemma 20 to construct a secret-sharing scheme realizing H with total share size $\tilde{O}(2^k k^k d^{k-1} n)$. \square

Corollary 24. *Let $H = (V, E)$ be a k -hypergraph with $|V| = n$ and $|E| = \binom{n}{k} - \ell$ for some $\ell k < n$. There exists a secret-sharing scheme realizing H with total share size $n + \tilde{O}(2^k k^{k+2} \ell^2)$.*

Proof. Define $W \subseteq V$ as the set of vertices of degree zero in \overline{H} . Since $\ell k < n$, $|W| > 0$. Consider the k -hypergraph $H' = (V, E')$ with $E' = \{e \in \binom{V}{k} : |e \cap W| \geq 1\}$. Observe that $H' \subseteq H$. By [40], there exists an ideal secret-sharing scheme realizing H' . Now it remains to find a secret-sharing scheme for $H \setminus H'$, a hypergraph defined on $V \setminus W$ whose complement has at most ℓk vertices and ℓ hyperedges. The proof is completed by using Theorem 23. \square

Remark 25. By [27], the scheme constructed in the first step of the proof of Corollary 24 can be constructed over any finite field \mathbb{F} with $|\mathbb{F}| > \binom{n+1}{k}$.

6 Removing Few Authorized Sets from Access Structures

Our main result (Theorem 18) shows that if we start with the complete graph and remove “few” edges, then the share size required to realize the new graph is not “too big”. We then generalize these results to complete homogeneous hypergraphs. In this section we address the effect of removing few authorized sets from an access structure. We first consider arbitrary graph access structures and then consider access structures where the minimal authorized sets are small and for each party we remove few sets containing the party (this generalizes the case where the complement graph has constant degree). Due to space constraints, the proofs in this section will appear in the full version of this paper.

Theorem 26. *Let $G = (V, E)$ and $G' = (V, E')$ be two graphs with $E' \subset E$, $|E \setminus E'| = \ell$, and $|V| = n$. Assume G can be realized by a scheme in which the total share size is m (clearly, $m \leq n^2$). If $\ell > m/n$, then G' can be realized by a scheme in which the total share size is $\tilde{O}(\sqrt{\ell mn})$. If $\ell \leq m/n$, then G' can be realized by a scheme in which the total share size is $m + 2\ell n \leq 3m$.*

In the interesting case in Theorem 26 when $\ell > m/n$, the total share size is $\tilde{O}(\sqrt{\ell mn})$. This is better than the trivial scheme giving shares of total size $O(n^2)$ only when ℓ is not too large, namely, $\ell \ll n^3/m$.

We next consider removing authorized sets from more general access structures. We say that access structure Γ is of *degree d* if for every $p \in P$ there are at most d subsets in $\min \Gamma$ containing p .

Theorem 27. *Let Γ_1 and Γ_2 be two access structures on P with $\min \Gamma_2 \subset \min \Gamma_1$ satisfying that $|A| \leq k$ for every $A \in \min \Gamma_1$. If Γ_2 is of degree d and there exists a scheme realizing Γ_1 with total share size m , then the access structure determined by $\min \Gamma_1 \setminus \min \Gamma_2$ can be realized by a secret-sharing scheme with total share size $\tilde{O}(2^k k^k d^{k-1} m)$.*

Observe that if $k \ll n$, the removal of minimal authorized subsets from an access structure does not increase so much the share size. Therefore, for $k \ll n$, access structures close to an access structure realized by an efficient scheme are not “hard”.

7 Lower Bounds for Very Dense Graphs

In this section we show lower bounds on the total share size for realizing very dense graphs. Recall that the best lower bound on the total share size for realizing a graph is $\Omega(n \log n)$ [25,10,21] and the best lower bound on the total share size for realizing a graph by a linear scheme is $\Omega(n^{3/2})$ [6]. However, these lower bounds use sparse graphs with $\Theta(n \log n)$ and $\Omega(n^{3/2})$ edges respectively. In this section we will show how to use these sparse graphs to prove lower bounds for very dense graphs. In particular, we show that there exists a graph with $n^{1+\beta}$ excluded edges such that in every linear secret-sharing realizing it, the total share size is $\Omega(n^{1+\beta/2})$ (for every $0 \leq \beta < 1$). This lower bound shows that the total share size grows as a function of β . However, there is still a gap between our upper and lower bounds. We start with a lower bound for graphs with less than n excluded edges.

Theorem 28. *For every n and every $2 < \ell < n/2$, there exists a graph with n vertices and ℓ excluded edges such that the total share size of every secret-sharing realizing it is at least $n + \ell$.*

Proof. We construct a graph $G = (V, E)$ with $n \geq 2\ell + 1$ vertices. We denote the vertices of the graph by $V = \{a, b_0, \dots, b_{\ell-1}, c_0, \dots, c_{\ell-1}, v_{2\ell+2}, \dots, v_n\}$. The graph G has all edges except for the following ℓ excluded edges: $\overline{E} = \{(a, c_i) : 0 \leq i \leq \ell - 1\}$.

For every $0 \leq i \leq \ell - 1$, consider the graph G restricted to the vertices $a, b_i, c_i, c_{(i+1) \bmod \ell}$. This graph has two excluded edges (a, c_i) and $(a, c_{(i+1) \bmod \ell})$. Blundo et al. [11] proved that in any secret-sharing realizing this graph, the sum of the sizes of the shares of b_i and c_i is at least 3 times the size of the secret. Thus, in any secret-sharing realizing G , the sum of the sizes of the shares of b_i and c_i is at least 3 times the size of the secret. By [35], the size of the share of each party in any secret-sharing realizing any graph with no isolated vertices is at least the size of the secret. Thus, the total share size in any secret-sharing realizing G is at least $n + \ell$. \square

Theorem 29. *For every β , where $0 \leq \beta < 1$, there exists a graph with n vertices and less than $n^{1+\beta}$ excluded edges, such that the total share size in any linear secret-sharing realizing it is $\Omega(n^{1+\beta/2})$.*

Proof. By [6], for every n there exists a graph with n vertices such that the total share size in any linear secret-sharing realizing it is $\Omega(n^{3/2})$. We use this graph to construct a dense graph $G = (V, E)$ with n vertices. We partition the vertices of G into $n^{1-\beta}$ disjoint sets of vertices $\mathcal{V}_1, \dots, \mathcal{V}_{n^{1-\beta}}$, where $|\mathcal{V}_i| = n^\beta$ for $1 \leq i \leq n^{1-\beta}$. We construct the edges as follows: First, for every 2 vertices u and v such that $u \in \mathcal{V}_i$ and $v \in \mathcal{V}_j$ for $i \neq j$, we add the edge (u, v) to E , i.e., there is an edge connecting every 2 vertices from different parts. Second, for every i (where $1 \leq i \leq n^{1-\beta}$), we construct a copy of the graph from [6] with n^β vertices among the vertices of \mathcal{V}_i . We denote this graph by $G_{\mathcal{V}_i}$.

Since all excluded edges in the above construction are between vertices in the same part, the number of excluded edges is at most $\binom{n^\beta}{2} n^{1-\beta} < n^{1+\beta}$. The total share size of any linear secret-sharing scheme realizing $G_{\mathcal{V}_i}$ (for $1 \leq i \leq n^{1-\beta}$) is $\Omega((n^\beta)^{3/2}) = \Omega(n^{3\beta/2})$. Thus, the total share size of any linear secret-sharing scheme realizing G is at least $\Omega(n^{1-\beta} n^{3\beta/2}) = \Omega(n^{1+\beta/2})$. \square

Theorem 30. *For every β , where $0 < \beta < 1$, there exists a graph with n vertices and less than $n^{1+\beta}$ excluded edges such that the share size of any secret-sharing scheme realizing it is $\Omega(\beta n \log n)$.*

Proof. We use the construction from the proof of Theorem 29, where for every $1 \leq i \leq n^{1-\beta}$ we set $G_{\mathcal{V}_i}$ to be a $\log n^\beta$ -dimensional cube. By [21], any secret-sharing scheme realizing $G_{\mathcal{V}_i}$ has a total share size of $\Omega(\beta n^\beta \log n)$. Thus, any secret-sharing scheme realizing G must have a total share size of $\Omega((n^{1-\beta}) \cdot \beta n^\beta \log n) = \Omega(\beta n \log n)$. \square

Acknowledgment. We thank Noga Alon and Stasys Jukna for discussions on equivalence covering, and Ilan Orlov for useful discussions and suggestions.

The second author's work is partly supported by the Spanish Government through projects TIN201127076-C03-01, 2010 CSD2007-00004, and by the Catalan Government through grant 2009 SGR 1135.

References

1. Alon, N.: Covering graphs by the minimum number of equivalence relations. Combinatorica 6(3), 201–206 (1986)
2. Alon, N., Spencer, J.H.: The Probabilistic Method, 3rd edn. John Wiley & Sons (2008)
3. Babai, L., Gál, A., Wigderson, A.: Superpolynomial lower bounds for monotone span programs. Combinatorica 19(3), 301–319 (1999)
4. Beimel, A.: Secret-Sharing Schemes: A Survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 11–46. Springer, Heidelberg (2011)

5. Beimel, A., Chor, B.: Universally ideal secret sharing schemes. *IEEE Trans. on Information Theory* 40(3), 786–794 (1994)
6. Beimel, A., Gál, A., Paterson, M.: Lower bounds for monotone span programs. *Computational Complexity* 6(1), 29–45 (1997)
7. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for noncryptographic fault-tolerant distributed computations. In: 20th STOC, pp. 1–10 (1988)
8. Benaloh, J., Leichter, J.: Generalized Secret Sharing and Monotone Functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–35. Springer, Heidelberg (1990)
9. Blakley, G.R.: Safeguarding cryptographic keys. In: 1979 AFIPS National Computer Conference, pp. 313–317 (1979)
10. Blundo, C., De Santis, A., de Simone, R., Vaccaro, U.: Tight bounds on the information rate of secret sharing schemes. *Des. Codes Cryptogr.* 11(2), 107–122 (1997)
11. Blundo, C., De Santis, A., Gargano, L., Vaccaro, U.: On the information rate of secret sharing schemes. *Theoretical Computer Science* 154(2), 283–306 (1996)
12. Blundo, C., De Santis, A., Stinson, D.R., Vaccaro, U.: Graph decomposition and secret sharing schemes. *J. of Cryptology* 8(1), 39–64 (1995)
13. Brickell, E.F.: Some ideal secret sharing schemes. *Journal of Combin. Math. and Combin. Comput.* 6, 105–113 (1989)
14. Brickell, E.F., Davenport, D.M.: On the classification of ideal secret sharing schemes. *J. of Cryptology* 4(73), 123–134 (1991)
15. Bublitz, S.: Decomposition of graphs and monotone formula size of homogeneous functions. *Acta Informatica* 23, 689–696 (1986)
16. Capocelli, R.M., De Santis, A., Gargano, L., Vaccaro, U.: On the size of shares for secret sharing schemes. *J. of Cryptology* 6(3), 157–168 (1993)
17. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: 20th STOC, pp. 11–19 (1988)
18. Chor, B., Kushilevitz, E.: Secret sharing over infinite domains. *J. of Cryptology* 6(2), 87–96 (1993)
19. Cramer, R., Damgård, I., Maurer, U.: General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
20. Di Crescenzo, G., Galdi, C.: Hypergraph decomposition and secret sharing. *Discrete Applied Mathematics* 157(5), 928–946 (2009)
21. Csirmaz, L.: Secret sharing schemes on graphs. *Cryptology ePrint Archive*, 2005/059 (2005)
22. Csirmaz, L.: An impossibility result on graph secret sharing. *Des. Codes Cryptog.* 53(3), 195–209 (2009)
23. Csirmaz, L., Tardos, G.: Secret sharing on trees: problem solved. *Cryptology ePrint Archive*, 2009/71 (2009)
24. Desmedt, Y., Frankel, Y.: Shared Generation of Authenticators and Signatures. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 457–469. Springer, Heidelberg (1992)
25. van Dijk, M.: On the information rate of perfect secret sharing schemes. *Des. Codes and Cryptog.* 6, 143–169 (1995)
26. Erdős, P., Pyber, L.: Covering a graph by complete bipartite graphs. *Discrete Mathematics* 170(1-3), 249–251 (1997)
27. Farràs, O., Martí-Farré, J., Padró, C.: Ideal multipartite secret sharing schemes. *J. of Cryptology* 25(1), 434–463 (2012)
28. Gál, A.: A characterization of span program size and improved lower bounds for monotone span programs. In: 30th STOC, pp. 429–437 (1998)

29. Gál, A., Pudlák, P.: Monotone complexity and the rank of matrices. *IPL* 87, 321–326 (2003)
30. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: 13th CCS, pp. 89–98 (2006)
31. Ito, M., Saito, A., Nishizeki, T.: Secret sharing schemes realizing general access structure. In: Globecom 1987, pp. 99–102 (1987)
32. Jerrum, M.: A very simple algorithm for estimating the number of k -colorings of a low-degree graph. *Random Structures & Algorithms* 7, 157–166 (1995)
33. Jukna, S.: On set intersection representations of graphs. *Journal of Graph Theory* 61, 55–75 (2009)
34. Karchmer, M., Wigderson, A.: On span programs. In: 8th Structure in Complexity Theory, pp. 102–111 (1993)
35. Karnin, E.D., Greene, J.W., Hellman, M.E.: On secret sharing systems. *IEEE Trans. on Information Theory* 29(1), 35–41 (1983)
36. Martí-Farré, J., Padró, C.: Secret sharing schemes on sparse homogeneous access structures with rank three. *Electr. J. Comb.* 11(1) (2004)
37. Martí-Farré, J., Padró, C.: Secret sharing schemes with three or four minimal qualified subsets. *Des. Codes Cryptog.* 34(1), 17–34 (2005)
38. Martí-Farré, J., Padró, C.: On secret sharing schemes, matroids and polymatroids. *Journal of Mathematical Cryptology* 4(2), 95–120 (2010)
39. Naor, M., Wool, A.: Access control and signatures via quorum secret sharing. In: 3rd CCS, pp. 157–167 (1996)
40. Padró, C., Sáez, G.: Secret sharing schemes with bipartite access structure. *IEEE Trans. on Information Theory* 46, 2596–2605 (2000)
41. Padró, C., Sáez, G.: Lower bounds on the information rate of secret sharing schemes with homogeneous access structure. *IPL* 83(6), 345–351 (2002)
42. Salas, J., Sokal, A.D.: Absence of phase transition for antiferromagnetic Potts models via the Dobrushin uniqueness theorem. *J. Statist. Phys.* 86, 551–579 (1997)
43. Shamir, A.: How to share a secret. *Communications of the ACM* 22, 612–613 (1979)
44. Shankar, B., Srinathan, K., Pandu Rangan, C.: Alternative Protocols for Generalized Oblivious Transfer. In: Rao, S., Chatterjee, M., Jayanti, P., Murthy, C.S.R., Saha, S.K. (eds.) ICDCN 2008. LNCS, vol. 4904, pp. 304–309. Springer, Heidelberg (2008)
45. Simmons, G.J., Jackson, W., Martin, K.M.: The geometry of shared secret schemes. *Bulletin of the ICA* 1, 71–88 (1991)
46. Stinson, D.R.: New General Lower Bounds on the Information Rate of Secret Sharing Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 168–182. Springer, Heidelberg (1993)
47. Stinson, D.R.: Decomposition construction for secret sharing schemes. *IEEE Trans. on Information Theory* 40(1), 118–125 (1994)
48. Sun, H., Shieh, S.: Secret sharing in graph-based prohibited structures. In: INFO-COM, pp. 718–724 (1997)
49. Sun, H.-M., Wang, H., Ku, B.-H., Pieprzyk, J.: Decomposition construction for secret sharing schemes with graph access structures in polynomial time. *SIAM J. Discret. Math.* 24, 617–638 (2010)
50. Tassa, T.: Generalized oblivious transfer by secret sharing. *Des. Codes Cryptog.* 58(1), 11–21 (2011)
51. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
52. Wegener, I.: The Complexity of Boolean Functions. Wiley-Teubner (1987)

Functional Encryption with Bounded Collusions via Multi-party Computation

Sergey Gorbunov^{1,*}, Vinod Vaikuntanathan^{1,**}, and Hoeteck Wee^{2,***}

¹ University of Toronto

² George Washington University

Abstract. We construct functional encryption schemes for polynomial-time computable functions secure against an a-priori bounded polynomial number of collusions. Our constructions require only semantically secure public-key encryption schemes and pseudorandom generators computable by small-depth circuits (known to be implied by most concrete intractability assumptions). For certain special cases such as predicate encryption schemes with public index, the construction requires only semantically secure encryption schemes.

Along the way, we show a “bootstrapping theorem” that builds a q -query functional encryption scheme for arbitrary functions starting from a q -query functional encryption scheme for *bounded-degree* functions. All our constructions rely heavily on techniques from secure multi-party computation and randomized encodings.

Our constructions are secure under a strong simulation-based definition of functional encryption.

1 Introduction

Traditional notions of public-key encryption provide *all-or-nothing* access to data: users who possess the secret key can recover the entire message from a ciphertext, whereas those who do not know the secret key learn nothing at all. While such “black-and-white” notions of encryption have served us well for the past thirty years and are indeed being widely used for secure communications and storage, it is time to move beyond. In particular, the advent of cloud computing and the resulting demand for privacy-preserving technologies requires that we come up with a much more fine-grained access control mechanism for encrypted data.

* Supported by NSERC Alexander Graham Bell Graduate Scholarship.

** Supported by an NSERC Discovery Grant and by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

*** Supported by NSF CAREER Award CNS-1237429.

Boneh, Sahai and Waters [BSW11] recently formalized the notion of functional encryption towards this end, building on and generalizing a number of previous constructs including (anonymous) identity-based encryption (IBE) [Sha84, BF01, Coc01, BW06], fuzzy IBE [SW05], attribute-based encryption (ABE) [GPSW06, LOS⁺10], and predicate encryption [KSW08, LOS⁺10].

Informally, a functional encryption scheme for a function $F(\cdot, \cdot)$ on two inputs – a “key” K and a “message” M – associates secret keys SK_K with every K and ciphertexts CT with every M . The owner of a secret key SK_K and a ciphertext CT of a message M should be able to obtain $F(K, M)$, but learn nothing else about the message M itself.

Constructions of functional encryption are known only for limited classes of functions (see [BF01, Coc01, SW05, GPSW06, KSW08, LOS⁺10] and others), leaving open a challenging open question: *Can we build functional encryption schemes for arbitrary (polynomial-time) functions?*

Much like its predecessors, functional encryption schemes are required to satisfy rather stringent security notions. In particular, a large part of the difficulty in constructing functional encryption schemes lies in the fact that we typically require security against adversaries who obtain secret keys for an unbounded number of inputs K_1, \dots, K_q .

In this work, we consider a relaxed notion of security where the adversary is given secret keys for an *a-priori bounded number of inputs* K_1, \dots, K_q of her choice (which can be made adaptively). This notion, which we call q -bounded security (or security against q collusions), is a natural relaxation of the strong definition above, and could be sufficient in a number of practical situations. In particular, one could envision scenarios where an authority releases secret keys for arbitrarily many inputs K , however the adversary can only form collusions of size at most q .

Our main result in this paper is a construction of q -bounded secure functional encryption schemes for *arbitrary polynomial-time functions* under *minimal cryptographic assumptions*.

The question of designing IBE schemes with bounded collusions has been considered in a number of works [DKXY02, CHH⁺07, GLW12]. The functional encryption setting presents us with a significantly more challenging landscape since (1) a secret key SK_K in functional encryption can be used to obtain (partial) information about many messages, as opposed to IBE where a secret key decrypts only ciphertexts for a single identity, and (2) the partial information is a result of a potentially complex computation on the key and the message together. Our constructions leverage interesting ideas from the study of (information-theoretic) multi-party computation (à la [BGW88, BMR90, DI05]) and randomized encodings [Yao86, IK00, AIK06].

1.1 Our Results

The main result of this work is the construction of q -query functional encryption schemes that:

1. handle arbitrary (polynomial-time computable) functions, and
2. are based on the existence of semantically secure public key encryption schemes, and pseudorandom generators (PRG) computable by polynomials of degree $\text{poly}(\kappa)$, where κ is the security parameter. The former is clearly a necessary assumption, and the latter is a relatively mild assumption which, in particular, is implied by most concrete intractability assumptions commonly used in cryptography, such as ones related to factoring, discrete logarithm, or lattice problems.

An important special case of functional encryption that we will be interested in is *predicate encryption with public index* (which is also called attribute-based encryption by some authors). This corresponds to a function F defined as:

$$F(K, (M, \mu)) = \begin{cases} (M, \mu) & \text{if } g(K, M) = \text{true} \\ (M, \perp) & \text{otherwise} \end{cases}$$

for some predicate g . In other words, the second input for F is divided into two parts – a “public index” M on which the computation takes place, and a secret “payload” μ which is conditionally released. For the special case of predicate encryption schemes with *public index*, our construction handles arbitrary polynomial-time functions while relying *solely* on the existence of semantically secure public-key encryption schemes, which is clearly the minimal necessary assumption. In particular, we do not need the “bounded-degree PRG” assumption for this construction.

In contrast, functional encryption schemes that handle an unbounded number of secret-key queries are known only for very limited classes of functions, the most general being inner product predicates [KSW08, LOS⁺10, OT10]. In particular, constructing an unbounded-query secure functional encryption scheme for general functions is considered a major open problem in this area [BSW11]. As for functional encryption schemes with public index (also referred to as “attribute-based encryption” by some authors) that handle an unbounded number of secret-key queries, there are a handful of constructions for polynomial-size formulas [GPSW06, OSW07], which themselves are a subclass of NC^1 circuits.

We will henceforth refer to a functional encryption scheme that supports arbitrary polynomial-time functions as a *general functional encryption* scheme. Summarizing this discussion, we show:

Theorem 1 (Main Theorem, Informal). *Let κ be a security parameter. Assuming the existence of semantically secure encryption schemes as well as PRGs computable by arithmetic circuits of degree- $\text{poly}(\kappa)$, for every $q = q(\kappa)$, there exists a general functional encryption scheme secure against q secret key queries.*

We have so far avoided discussing the issue of which security definition to use for functional encryption. Indeed, there are a number of different definitions in the literature, including both *indistinguishability* style and *simulation* style definitions. In a nutshell, we prove our constructions secure under a strong, adaptive simulation-based definition; see Section 1.3 for details.

1.2 Overview of Our Techniques

The starting point of our constructions is the fact, observed by Sahai and Seyalioglu [SS10], that *general* functional encryption schemes resilient against a *single* secret-key query can be readily constructed using the beautiful machinery of Yao’s “garbled circuits” [Yao86] (and in fact, more generally, from randomized encodings [IK00, AIK06]).¹ Building on this, our construction proceeds in two steps.

Construction 1: Functional Encryption for Bounded-Degree Functions. In the first step, we show how to construct a q -query functional encryption scheme for bounded-degree functions starting from any 1-query scheme (such as the one of [SS10]).

By bounded degree, we mean functions F such that for every K , the degree of the restriction $F(K, \cdot)$ is bounded a-priori by $D = D(\kappa)$ in the variables of M . This captures both arithmetic and boolean circuits of depth at most $\log D$ (with constant multiplicative fan-in). The complexity of our construction will be polynomial in both D and q , where q is the number of secret keys the adversary is allowed to see before he gets the challenge ciphertext. This construction assumes only the existence of semantically secure public-key encryption schemes. In addition, it also gives us for free a predicate encryption scheme with public index for *arbitrary polynomial-time* functions (with no a-priori bound on the degree).

The starting point of our construction is the BGW semi-honest MPC protocol without degree reduction (c.f. [DI05, Section 2.2]). Our main idea is to use the fact that this protocol is *completely non-interactive* when used to compute *bounded-degree* functions.

Suppose the encryptor holds input $M = (M_1, \dots, M_\ell)$, the decryptor holds input K , and the goal is for the decryptor to learn $F(K, M_1, \dots, M_\ell)$. The public/secret keys of the system consists of N independent public/secret keys for the 1-query scheme. To encrypt M , the encryptor first chooses ℓ random polynomials μ_1, \dots, μ_ℓ of degree t with constant terms M_1, \dots, M_ℓ respectively. In addition, she chooses a random polynomial ζ of degree Dt with constant term 0. (Here, t and N are parameters of the construction). Now, since $F(K, \cdot)$ has degree at most D , observe that

$$P(\cdot) := F(K, \mu_1(\cdot), \dots, \mu_\ell(\cdot)) + \zeta(\cdot)$$

is a univariate polynomial of degree at most Dt and whose constant term is $F(K, (M_1, \dots, M_\ell))$. The encryptor simply encrypts the shares $P(1), \dots, P(N)$ using the N public keys.

The key generation algorithm associates the receiver with a random subset $\Gamma \subseteq [N]$ of size $Dt + 1$ and generates secret keys for the public keys MPK_i

¹ We note that [SS10] is completely insecure for collusions of size two: in particular, given two secret keys $SK_{0\ell}$ and $SK_{1\ell}$, an adversary can derive the SK_K for any other K .

for $i \in \Gamma$. Using the underlying (1-query) FE scheme, the decryptor learns the evaluation of P on the points in Γ , which allows her to recover $F(K, M_1, \dots, M_\ell)$.

The key question now is: what happens when q of the decryptors collude? Let $\Gamma_1, \dots, \Gamma_q \subseteq [N]$ be the (uniformly random) sets chosen for each of the q secret key queries of the adversary. Whenever two of these sets intersect, the adversary obtains two distinct secret keys for the same public key in the underlying one-query FE scheme. More precisely, for every $j \in \Gamma_1 \cap \Gamma_2$, the adversary obtains two secret keys under the public key MPK_j . Since security of MPK_j is only guaranteed under a single adversarial query, we have to contend with the possibility that in this event, the adversary can potentially completely break the security of the public key MPK_j , and learn a share of the encrypted message M .

In particular, to guarantee security, we require that sets $\Gamma_1, \dots, \Gamma_q$ have *small pairwise intersections* which holds for a uniformly random choice of the sets under an appropriate choice of the parameters t and N . With small pairwise intersections, the adversary is guaranteed to learn at most t shares of the message M , which together reveal no information about M .

For technical reasons, we cannot establish security of the basic scheme. Informally, we need to rerandomize the polynomial P for each of the q queries. This can be done by having the encryptor hard-code additional randomness into the ciphertext. For more details, see Section 4.

To obtain a *predicate encryption scheme with public index*, we observe that the construction above satisfies a more general class of functions. In particular, if the input to the encryption algorithm is composed of a *public input* (that we do not wish to hide) and a *secret input* (that we do wish to hide), then the construction above only requires that the function F has small degree in the bits of the secret input. Informally, this is true because we do not care about hiding the public input, and thus, we will not secret share it in the construction above. Thus, the degree of the polynomial $P(\cdot)$ grows only with the degree of F in its secret inputs. The bottomline is that since predicate encryption schemes with *public index* deal with functions that have very low degree in the secret input (degree 1, in particular), our construction handles arbitrary predicates.

Construction 2: Bootstrapping Theorem for Functional Encryption. In the second step, we show a “bootstrapping theorem” for functional encryption schemes. In a nutshell, this shows how to generically convert a q -query secure functional encryption scheme for degree- D circuits into one that is q -query secure for arbitrary polynomial-time functions, assuming in addition the existence of a pseudo-random generator (PRG) that can be computed with circuits of degree $\text{poly}(\kappa)$. Such PRGs can be constructed based on most concrete intractability assumptions such as those related to factoring, discrete logarithms and lattices.

The main tool that enables our bootstrapping theorem is the notion of randomized encodings [Yao86, IK00, AIK06]. Instead of using the FE scheme to compute the (potentially complicated) function F , we use it to compute its randomized encoding \tilde{F} which is typically a much easier function to compute. In particular, secret keys are generated for K and the encryption algorithm for the bounded-degree scheme is used to encrypt the pair (M, R) , where R is a

uniformly random string. The rough intuition for security is that the randomized encoding $\tilde{F}(K, M; R)$ reveals “no more information than” $F(K, M)$ itself and thus, this transformation does not adversely affect the security of the scheme.

Unfortunately, intuitions can be misleading and so is this one. Note that in the q -query setting, the adversary obtains not just a single randomized encoding, but q of them, namely $\tilde{F}(K_1, M; R), \dots, \tilde{F}(K_q, M; R)$. Furthermore, since all these encodings use *the same randomness* R , the regular notion of security of randomized encodings does not apply *as-is*. We solve this issue by hard-coding a large number of random strings (proportional to q) in the ciphertext and using a cover-free set construction, ensuring that the adversary learns q randomized encodings with independently chosen randomness. See Section 5 for more details.

Putting this construction together with a randomized encoding scheme for polynomial-time computable functions (which follows from Yao’s garbled circuits [Yao86, AIK06]) whose complexity is essentially the complexity of computing a PRG, we get our final FE scheme.

As a bonus, we show a completely different way to bootstrap q -query FE schemes for small functions into a q -query FE scheme for polynomial-time functions, using a fully homomorphic encryption scheme [Gen09, BV11]. We defer the details to the full version.

1.3 Definitions of Functional Encryption

Our constructions are shown secure under a strong simulation-based definition, in both the adaptive and non-adaptive sense. The non-adaptive variant requires the adversary to make all its secret key queries before receiving the challenge ciphertext whereas in the adaptive variant, there is no such restriction. Although the adaptive variant is clearly stronger, Boneh, Sahai and Waters [BSW11] recently showed that it is also impossible to achieve, even for very simple functionalities (related to IBE). We observe that the BSW impossibility result holds only if the adversary obtains an unbounded number of ciphertexts (essentially because of a related lower bound for non-committing encryption schemes with unbounded messages). Faced with this state of affairs, we show our constructions are shown secure in the non-adaptive sense, as well as in the adaptive sense with a bounded number of messages. Due to lack of space, we deal with the non-adaptive setting in this paper, postponing a discussion of adaptive security to the full version.

1.4 A Perspective: Bounded-Use Garbled Circuits

The reason why the construction of Sahai and Seyalioglu only achieves security against collusions of size 1 is intimately related to the fact that Yao’s garbled circuits become completely insecure when used more than once. Our constructions may be viewed as a stateless variant of Yao’s garbled circuit that can be reused for some a-priori bounded number of executions. Fix a two-party functionality $F(K, M)$. Specifically, we can view the ciphertext as encoding computation of $F(\cdot, M)$ on some fixed value M , such that we can “delegate” computation on

q different inputs K_1, \dots, K_q without leaking any information about M beyond $F(K_1, M), \dots, F(K_q, M)$.

Organization of the Paper. After describing a simulation-based definition of functional encryption in Section 2, we describe Construction 1 for bounded-degree circuits in Section 4 and Construction 2 for bootstrapping in Section 5. For completeness, we have also included the construction of a 1-query functional encryption in the appendix.

2 Functional Encryption against Bounded Collusions

A functional encryption scheme for a family of functions $F = \{F_\kappa : \mathcal{K}_\kappa \times \mathcal{M}_\kappa \rightarrow \{0, 1\}\}_{\kappa \in \mathbb{N}}$ is a four-tuple of algorithms $(\text{FE}.\text{Setup}, \text{FE}.\text{Keygen}, \text{FE}.\text{Enc}, \text{FE}.\text{Dec})$ where:

- $\text{FE}.\text{Setup}(1^\kappa)$ generates a pair of keys – a master public key MPK and a master secret key MSK ;
- $\text{FE}.\text{Keygen}(\text{MSK}, K)$ takes as input $K \in \mathcal{K}_\kappa$ and generates a secret key SK_K ;
- $\text{FE}.\text{Enc}(\text{MPK}, M)$ takes as input $M \in \mathcal{M}_\kappa$ and generates a ciphertext CT ; and
- given SK_K and CT , $\text{FE}.\text{Dec}$ outputs $y \in \{0, 1\}$.

We require that for all but a negligible fraction of $(\text{MPK}, \text{MSK}) \leftarrow \text{FE}.\text{Setup}(1^\kappa)$, for all $SK_K \in \text{FE}.\text{Keygen}(\text{MSK}, K)$ and all $\text{CT} \in \text{FE}.\text{Enc}(\text{MPK}, M)$, the decryption algorithm $\text{FE}.\text{Dec}(SK_K, \text{CT})$ outputs $y = F_\kappa(K, M)$.

We now describe simulation-based definitions for functional encryption with *bounded* collusions, largely based on the recent works of Boneh, Sahai and Waters [BSW11] and O’Neill [O’N10]. We then go on to discuss relations between various flavors of these definitions, with details deferred to the full version.

Definition 1 (q -NA-SIM- and q -AD-SIM- Security). Let \mathcal{FE} be a functional encryption scheme for a family of functions $F = \{F_\kappa : \mathcal{K}_\kappa \times \mathcal{M}_\kappa\}_{\kappa \in \mathbb{N}}$. For every p.p.t. adversary $A = (A_1, A_2)$ and a p.p.t. simulator $S = (S_1, S_2)$, consider the following two experiments.

We distinguish between two cases of the experiment:

1. The adaptive case, where:

- the oracle $\mathcal{O}(\text{MSK}, \cdot) = \text{FE}.\text{Keygen}(\text{MSK}, \cdot)$ and
- the oracle $\mathcal{O}'(\text{MSK}, st', \cdot)$ is the second stage of the simulator, namely $S_2^{F(\cdot, M)}(\text{MSK}, st', \cdot)$.

The simulator algorithm S_2 is stateful in that after each invocation, it updates the state st' which is carried over to its next invocation. We call a simulator algorithm $S = (S_1, S_2)$ admissible if, on each input K , S_2 makes just a single query to its oracle $F(\cdot, M)$ on K itself.

$\text{Exp}_{\mathcal{F}\mathcal{E},A}^{\text{real}}(1^\kappa)$:	$\text{Exp}_{\mathcal{F}\mathcal{E},S}^{\text{ideal}}(1^\kappa)$:
1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE}.\text{Setup}(1^\kappa)$	1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE}.\text{Setup}(1^\kappa)$
2: $(M, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$	2: $(M, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$ ► Let (K_1, \dots, K_q) be A_1 's oracle queries ► Let SK_i be the oracle reply to K_i ► Let $\mathcal{V} := \{F_\kappa(K_i, M), K_i, \text{SK}_i\}$.
3: $\boxed{\text{CT} \leftarrow \text{FE}.\text{Enc}(\text{MPK}, M)}$	3: $\boxed{(\text{CT}, st') \leftarrow S_1(\text{MPK}, \mathcal{V}, 1^{ \mathcal{M} })}$
4: $\boxed{\alpha \leftarrow A_2^{\mathcal{O}(\text{MSK}, \cdot)}(\text{MPK}, \text{CT}, st)}$	4: $\boxed{\alpha \leftarrow A_2^{\mathcal{O}'(\text{MSK}, st', \cdot)}(\text{MPK}, \text{CT}, st)}$
5: $\text{Output } (\alpha, M)$	5: $\text{Output } (\alpha, M)$

The functional encryption scheme $\mathcal{F}\mathcal{E}$ is then said to be q -query simulation-secure for one message against adaptive adversaries (q -AD-SIM-secure, for short) if there is an admissible p.p.t. simulator $S = (S_1, S_2)$ such that for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most q queries, the following two distributions are computationally indistinguishable:

$$\left\{ \text{Exp}_{\mathcal{F}\mathcal{E},A}^{\text{real}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{Exp}_{\mathcal{F}\mathcal{E},S}^{\text{ideal}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}}$$

2. The non-adaptive case, where the oracles $\mathcal{O}(\text{MSK}, \cdot)$ and $\mathcal{O}'(\text{MSK}, st, \cdot)$ are both the “empty oracles” that return nothing: the functional encryption scheme $\mathcal{F}\mathcal{E}$ is then said to be q -query simulation-secure for one message against non-adaptive adversaries (q -NA-SIM-secure, for short) if there is a p.p.t. simulator $S = (S_1, \perp)$ such that for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most q queries, the two distributions above are computationally indistinguishable.

Intuitively, our security definition states that any information that the adversary is able to learn from the ciphertext and secret keys, can be obtained by a simulator from the secret keys and the outputs of the function alone. A number of remarks on this definition are in order.

1. In the non-adaptive definition, the only difference between the real and ideal experiments is in how the ciphertext is generated – in the real experiment, the ciphertext is an encryption of M , whereas in the ideal experiment, the simulator generates a simulated ciphertext given (the secret keys $\text{SK}_{K_1}, \dots, \text{SK}_{K_q}, K_1, \dots, K_q$ and) the output values $F(K_1, M), \dots, F(K_q, M)$. In the adaptive definition, we additionally allow the simulator to “program” the answers to the post-ciphertext secret-key queries.
2. Even if the adversary does not request any secret keys, he learns the length of M and therefore, the simulator should be given this information to be on even ground with the adversary. This also ensures that the definition properly generalizes (regular) public-key encryption.

3. We remark that our definitions imply (and are stronger than) those of presented in the work of Boneh, Sahai and Waters [BSW11]. We defer a discussion of this and other definitional implications to the full version.

Why focus on this definition? First, as mentioned above, our definition is at least as strong as the definition presented in [BSW11]. In addition, in the full version of this paper, we show the following relations between the definitions:

1. *Relations between simulation and indistinguishability:* We show that a *single* message simulation definition implies *single* message indistinguishability definition for both non-adaptive and adaptive worlds.
2. *Relations between single and many messages (simulation):* We show that a *single* message non-adaptive simulation implies *many* messages non-adaptive simulation definition. However, we cannot hope to achieve the same implication for adaptive world due to the impossibility results presented in [BSW11].
3. *Relations between single and many messages (indistinguishability):* Finally, we show that a *single* message indistinguishability implies *many* message indistinguishability definition in both the adaptive and non-adaptive worlds.

As a result of these definitional implications, we focus on proving that our constructions are secure under the *single* message simulation definitions for both adaptive and non-adaptive worlds.

3 Preliminaries and Standard Cryptographic Definitions

3.1 Shamir's Secret Sharing

We assume familiarity with Shamir's secret-sharing scheme [Sha79] which works as follows: Let \mathbb{F} be a finite field and let $\mathbf{x} = (x_1, \dots, x_n)$ be a vector of any distinct non-zero elements of \mathbb{F} , where $n < |\mathbb{F}|$. Shamir's t -out-of- n secret-sharing scheme works as follows:

- To share a secret $M \in \mathbb{F}$, the sharing algorithm $\text{SS.Share}_{t,n}(M)$ chooses a random univariate polynomial $\mu(x)$ of degree t with constant coefficient M . The n shares are $\mu(x_1), \dots, \mu(x_n)$. Note that any t or fewer shares look uniformly random.
- The reconstruction algorithm SS.Reconstruct takes as input $t + 1$ shares and uses Lagrange interpolation to find a unique degree- t polynomial $\mu(\cdot)$ that passes through the share points. Finally, it computes $\mu(0)$ to recover the secret.

An important property of this scheme is that it permits computation on the shares, a feature used in many multi-party computation protocols starting from [BGW88]. In particular, adding shares gives us $\mu_1(i) + \mu_2(i) = (\mu_1 + \mu_2)(i)$ meaning that that sharing scheme is additively homomorphic. Multiplying shares gives us $\mu_1(i)\mu_2(i) = (\mu_1\mu_2)(i)$ meaning that the scheme is also multiplicatively

homomorphic (where $\mu_1\mu_2$ denotes the product of the polynomials). The main catch is that the degree of the polynomial increases with the number of multiplications, requires more shares to recover the answer post multiplication. In other words, the scheme per se is multiplicatively homomorphic for a bounded number of multiplications (but an arbitrary number of additions).

3.2 Decomposable Randomized Encoding

Let \mathcal{C} be a circuit that takes inputs $K \in \{0,1\}^\ell, M \in \{0,1\}^n$ and outputs $\mathcal{C}(K, M) \in \{0,1\}^m$. A decomposable randomized encoding scheme \mathcal{RE} consists of two algorithms ($\text{RE.Encode}, \text{RE.Decode}$) satisfying the following properties:

- 1. Decomposable Encoding.** $\text{RE.Encode}(1^\kappa, \mathcal{C}, M)$: A p.p.t. algorithm takes as inputs a security parameter, a description of a circuit \mathcal{C} , an input M and outputs a randomized encoding:

$(\tilde{\mathcal{C}}_1(\cdot, M; R), \dots, \tilde{\mathcal{C}}_\ell(\cdot, M; R))$ for $i \in [\ell]$, where $\tilde{\mathcal{C}}_i(\cdot, M; R)$ depends only on K_i

- 2. Decoding.** $\text{RE.Decode}((\tilde{Y}_i)_{i=1}^\ell)$: On input an encoding of a circuit $\tilde{Y}_i = \tilde{\mathcal{C}}_i(K_i, M; R)$ for some $K = (K_1, \dots, K_\ell)$ output $\mathcal{C}(K, M)$.
- 3. Semantic Security.** We say decomposable randomized encoding \mathcal{RE} is secure if there exists a p.p.t. simulator RE.Sim , such that for every p.p.t. adversary A , every circuit \mathcal{C} , and inputs $K = (K_1, \dots, K_\ell)$ and M , the outputs of the following two distributions are computationally indistinguishable:

$$\left\{ \tilde{\mathcal{C}}_i(K_i, M; R) \right\}_{i=1}^\ell : \tilde{\mathcal{C}}_i(\cdot, M; R) \leftarrow \text{RE.Encode}(1^\kappa, \mathcal{C}, M) \right\} \approx_c \left\{ \tilde{\mathcal{C}}_i(K_i, M; R) \right\}_{i=1}^\ell \leftarrow \text{RE.Sim}(1^\kappa, \mathcal{C}, \mathcal{C}(K, M)) \right\}$$

Note that such a randomized encoding for arbitrary polynomial-size circuits follows from Yao's garbled circuit construction [Yao86, AIK06].

4 A Construction for Bounded-Degree Functions

In this section, we construct a functional encryption scheme for functions that can be computed by circuits of bounded degree (see below for a precise definition), secure against an a-priori bounded number of non-adaptive secret key queries. Our construction will rely on any semantically secure public-key encryption scheme.

The Class of Functions. We consider the class of deterministic functions that computes a bounded-degree polynomial over the message space for some fixed degree bound D . That is, the message space $\mathcal{M} = \mathbb{F}^\ell$ is an ℓ -tuple of field elements, and for every key $K \in \mathcal{K}$, $F(K, \cdot)$ is an ℓ -variate polynomial over \mathbb{F}

of total degree at most D (in the second input). This captures both arithmetic and boolean circuits of depth at most $\log D$ (with constant multiplicative fan-in). Specifically, to handle boolean circuits, we let \mathbb{F} be a sufficiently large field extension of \mathbb{F}_2 . The complexity of our construction will be polynomial in both D and q , where q is the number of secret keys the adversary is allowed to see before he gets the challenge ciphertext.

Building Block. The main result of this section shows how to construct a functional encryption scheme for degree- D functions secure against q (non-adaptive) secret-key queries, starting from one that is secure against a *single non-adaptive secret-key query*. Sahai and Seyalioglu [SS10] have shown that FE schemes (for general polynomial-time functions) secure against a single query can be readily constructed using Yao’s garbled circuits [Yao86].

4.1 Our Construction

Let F be a functionality with circuits of degree $D = D(\kappa)$ in its second input (namely, the message M), and let $q = q(\kappa)$ be a bound on the number of secret key queries. Our scheme is associated with additional parameters $S = S(\kappa)$, $N = N(\kappa)$, $t = t(\kappa)$ and $v = v(\kappa)$ (for an instantiation of the parameters, see Section 4.2).

We start by defining a new functionality G as follows:

$$G((K, \Delta), (M, Z_1, \dots, Z_S)) := F(K, M) + \sum_{i \in \Delta} Z_i \quad (1)$$

where $\Delta \subseteq [S]$ and $Z_1, \dots, Z_S \in \mathbb{F}$.

Let $(\text{OneQFE}.\text{Setup}, \text{OneQFE}.\text{Keygen}, \text{OneQFE}.\text{Enc}, \text{OneQFE}.\text{Dec})$ be a functional encryption scheme for G secure against a *single* secret key query. Our q -query secure scheme $\mathcal{BDFE} = (\text{BdFE}.\text{Setup}, \text{BdFE}.\text{Keygen}, \text{BdFE}.\text{Enc}, \text{BdFE}.\text{Dec})$ for F works as follows:

- **Setup** $\text{BdFE}.\text{Setup}(1^\kappa)$: Run the one-query setup algorithm N times to generate independent master public-key/secret-key pairs

$$(\text{MPK}_i, \text{MSK}_i) \leftarrow \text{OneQFE}.\text{Setup}(1^\kappa) \quad \text{for } i = 1, \dots, N$$

Output $(\text{MPK}_i)_{i=1}^N$ as the master public key and $(\text{MSK}_i)_{i=1}^N$ as the master secret key.

- **Key Generation** $\text{BdFE}.\text{Keygen}(\text{MSK}, K)$: On input the master secret key MSK and a key $K \in \mathcal{K}$ for the functionality,
 1. Choose a uniformly random set $\Gamma \subseteq [N]$ of size $tD + 1$;
 2. Choose a uniformly random set $\Delta \subseteq [S]$ of size v ;
 3. Generate the secret keys

$$\text{SK}_{K, \Delta, i} \leftarrow \text{OneQFE}.\text{Keygen}(\text{MSK}_i, (K, \Delta)) \quad \text{for every } i \in \Gamma$$

Output as secret key $\text{SK}_K := (\Gamma, \Delta, (\text{SK}_{K,\Delta,i})_{i \in \Gamma})$.

- **Encryption** $\text{BdFE}.\text{Enc}(\text{MPK}, M)$: On input the master public key $\text{MPK} = (\text{MPK}_i)_{i=1}^N$ and a message $M = (M_1, \dots, M_\ell) \in \mathcal{M}$:

1. For $i = 1, 2, \dots, \ell$, pick a random degree t polynomial $\mu_i(\cdot)$ whose constant term is M_i .
2. For $i = 1, 2, \dots, S$, pick a random degree Dt polynomial $\zeta_i(\cdot)$ whose constant term is 0.
3. Run the one-query encryption algorithm $\text{OneQFE}.\text{Enc}$ N times to produce ciphertexts

$$\text{CT}_i \leftarrow \text{OneQFE}.\text{Enc}(\text{MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \zeta_1(i), \dots, \zeta_S(i)))$$

for $i = 1 \dots N$.

Output $(\text{CT}_i)_{i=1}^N$ as the ciphertext.

- **Decryption** $\text{BdFE}.\text{Dec}(\text{SK}_K, \text{CT})$: On input a secret key SK_K and a ciphertext CT , do the following:

1. Parse $\text{SK}_K == (\Gamma, \Delta, (\text{SK}_{K,\Delta,i})_{i \in \Gamma})$ and $\text{CT} = (\text{CT}_i)_{i=1}^N$.
2. Compute a degree Dt polynomial $\eta(\cdot)$ such that

$$\eta(i) = \text{OneQFE}.\text{Dec}(\text{SK}_{K,\Delta,i}, \text{CT}_i)$$

for all $i \in \Gamma$.

3. Output $\eta(0)$.

We first show that the scheme above is correct. By correctness of the underlying single-query FE, we have that for all $i \in \Gamma$,

$$\begin{aligned} \eta(i) &= G((K, \Delta), (\mu_1(i), \dots, \mu_\ell(i)), \zeta_1(i), \dots, \zeta_S(i)) \\ &= F(K, (\mu_1(i), \dots, \mu_\ell(i))) + \sum_{a \in \Delta} \zeta_a(i) \end{aligned}$$

Since $|\Gamma| \geq Dt + 1$, this means that η is equal to the degree Dt polynomial

$$\eta(\cdot) = F(K, (\mu_1(\cdot), \dots, \mu_\ell(\cdot))) + \sum_{a \in \Delta} \zeta_a(\cdot)$$

Hence, $\eta(0) = F(K, (M_1, \dots, M_\ell)) = F(K, M)$.

4.2 Setting the Parameters

We show how to set the parameters $S = S(\kappa)$, $N = N(\kappa)$ and $t = t(\kappa)$. These parameters govern the choice of the sets Γ and Δ during the key generation algorithm, and are required to satisfy the following two conditions:

Small Pairwise Intersections. Let $\Gamma_1, \dots, \Gamma_q \subseteq [N]$ be the (uniformly random) sets chosen for each of the q secret key queries of the adversary. Whenever two

of these sets intersect, the adversary obtains two distinct secret keys for the underlying one-query secure FE scheme. More precisely, for every $j \in \Gamma_1 \cap \Gamma_2$, the adversary obtains two secret keys under the public key MPK_j . Since security of MPK_j is only guaranteed under a single adversarial query, we have to contend with the possibility that in this event, the adversary can potentially completely break the security of the public key MPK_j . In particular, for every such j , the adversary potentially learns a share of the encrypted message M .

Thus, to guarantee security, we require that the union of the pairwise intersections of $\Gamma_1, \dots, \Gamma_q$ is small. In particular, we require that $\left| \bigcup_{i \neq j} (\Gamma_i \cap \Gamma_j) \right| \leq t$. This ensures that the adversary learns at most t shares of the message M , which together reveal no information about M .

A simple probabilistic argument shows that this is true (with probability $1 - 2^{-\Omega(t/q^2)}$) as long as $q^2 \cdot (Dt/N)^2 \cdot N \leq t/10$. In other words, we will set $t(\kappa) = \Theta(q^2\kappa)$ and $N(\kappa) = \Theta(D^2q^2t)$ which satisfies the above constraint with probability $1 - 2^{-\Omega(\kappa)}$.

Cover-Freeness. Let $\Delta_1, \dots, \Delta_q \subseteq [S]$ be the (uniformly random) sets chosen for each of the q secret key queries of the adversary. The security proof relies on the condition that the polynomials $\sum_{a \in \Delta_j} \zeta_a(\cdot)$ are uniformly random and independent which is true if the collection of sets $\Delta_1, \dots, \Delta_q$ is cover-free. That is, for every $i \in [q]$: $\Delta_i \setminus \left(\bigcup_{j \neq i} \Delta_j \right) \neq \phi$.

A simple probabilistic argument shows that this is true (with probability $1 - 2^{-\Omega(q^2v^2/S)}$) as long as $q^2v^2/S \leq v/100$. In other words, we will set $v(\kappa) = \Theta(\kappa)$ and $S(\kappa) = \Theta(vq^2)$ which satisfies the above constraint with probability $1 - 2^{-\Omega(\kappa)}$.

We remark that in our construction, multiple secret key queries for the same $K \in \mathcal{K}$ result in different secret keys SK_K , essentially because of the different random choices of the sets Δ and Γ . Using a pseudorandom function (applied to K), it is possible to ensure that multiple secret key queries for the same K result in the same answer.

4.3 Proof of Security

Overview. We prove that \mathcal{BDFE} is q -NA-SIM-secure. Recall that the simulator gets as input all of the following values: 1) the public key: $\text{MPK} = (\text{MPK}_1, \dots, \text{MPK}_N)$; 2) the queries and outputs of F : (K_1, \dots, K_q) , and all the outputs $F(K_1, M), \dots, F(K_q, M)$; 3) the corresponding secret keys: $\text{SK}_1, \dots, \text{SK}_q$, which determine the sets $\Gamma_1, \dots, \Gamma_q, \Delta_1, \dots, \Delta_q$.

We describe our strategy for simulating the ciphertext $\text{CT} = (\text{CT}_1, \dots, \text{CT}_N)$. Let $\mathcal{I} := \bigcup_{j \neq j'} (\Gamma_j \cap \Gamma_{j'})$. We will consider two cases:

- $i \notin \mathcal{I}$: Here, we issue at most one secret key corresponding to $(\text{MPK}_i, \text{MSK}_i)$; this is because at most one of the sets $\Gamma_1, \dots, \Gamma_q$ contains i . Therefore,

we may appeal to the security of the underlying one-query FE scheme. Specifically, we simulate CT_i computationally using the simulator for the underlying one-query FE scheme.

- $i \in \mathcal{I}$: Here, we may issue more than one secret key corresponding to $(\text{MPK}_i, \text{MSK}_i)$; therefore, we can no longer rely on the security of the underlying one-query FE scheme. Instead, we rely on the statistical security of the underlying MPC protocol and the fact that $|\mathcal{I}| \leq t$. Specifically, we simulate CT_i statistically as in an honestly generated ciphertext.

We refer the reader to the full version for the formal proof of security.

5 A Bootstrapping Theorem for Functional Encryption

In this section, we show a “bootstrapping-type” theorem for functional encryption (FE). In a nutshell, this shows how to take a q -query functional encryption scheme for “bounded degree” circuits, and transform them into a q -query functional encryption scheme for arbitrary polynomial-size circuits. The transformation relies on the existence of a pseudorandom generator (PRG) that stretches the seed by a constant factor, and which can be computed by circuits of degree $\text{poly}(\kappa)$. This is a relatively mild assumption, and in particular, is implied by most concrete intractability assumptions commonly used in cryptography, such as ones related to factoring, discrete logarithm, or lattice problems.

In a high-level the idea is this: Suppose we wish to construct an FE scheme for a polynomial-size circuit $F(K, M)$, and let $\tilde{F}(K, M; R)$ denote a randomized encoding of F that is computable by a constant-depth circuit with respect to the inputs M and R . By [AIK06, Theorem 4.14], we know that assuming the existence of a pseudo-random generator in $\oplus\text{L}/\text{poly}$, such a randomized encoding exists for every polynomial-size circuit F .

Consider the function G that works in the following way:

$$G((K, \Delta), (M, R_1, \dots, R_S)) := \tilde{F}\left(K, M; \bigoplus_{a \in \Delta} R_a\right)$$

Observe the following:

- Since $G(K, \cdot, \cdot)$ is computable by a constant-depth circuit, then $G((K, \Delta), \cdot)$ is computable by a constant-degree polynomial. Using the result from the previous scheme, we have a q -NA-SIM-secure FE scheme for G .
- Given a functional encryption scheme for G , it is easy to construct one for F . Decryption works by first recovering the output of G and then applying the decoder for the randomized encoding.
- Informally, 1-AD-SIM-security follows from the fact that the ciphertext together with the secret key reveals only the output of $G(K, M)$, which in turn reveals no more information than $F(K, M)$. More formally, given $F(K, M)$, we can simulate $G(K, M)$ and then the ciphertext, using first the simulator for the randomized encoding and then that for the underlying FE scheme.

- The role of the subset Δ is similar to that in the preceding construction — to “rerandomize” the randomness used in G , which is necessary to achieve q -AD-SIM-security.

Functional Encryption Scheme for F . Let $(\text{BdFE}.\text{Setup}, \text{BdFE}.\text{Keygen}, \text{BdFE}.\text{Enc}, \text{BdFE}.\text{Dec})$ be a q -AD-SIM-secure scheme for G , with a simulator $\text{BdFE}.\text{Sim}$. We construct an encryption scheme $(\text{FE}.\text{Setup}, \text{FE}.\text{Keygen}, \text{FE}.\text{Enc}, \text{FE}.\text{Dec})$ for F works as follows (that takes parameters S, v as before).

- **Setup** $\text{FE}.\text{Setup}(1^\kappa)$: Run the bounded FE setup algorithm to generate a master public-key/secret-key pair $(\text{MPK}, \text{MSK}) \leftarrow \text{BdFE}.\text{Setup}(1^\kappa)$.
- **Key Generation** $\text{FE}.\text{Keygen}(\text{MSK}, K)$: On input the master secret key MSK and a key $K \in \mathcal{K}$ for the functionality F , do the following:
 1. Choose a uniformly random set $\Delta \subseteq [S]$ of size v ;
 2. Generate the secret key $\text{SK}_{K, \Delta} \leftarrow \text{BdFE}.\text{Keygen}(\text{MSK}, (K, \Delta))$. for the functionality G .
 Output as secret key $\text{SK}_K := (\Delta, \text{SK}_{K, \Delta})$.
- **Encryption** $\text{FE}.\text{Enc}(\text{MPK}, M)$: On input the master public key MPK and a message $M \in \mathcal{M}$, do the following:
 1. For $i = 1, 2, \dots, S$, choose uniformly random $R_i \xleftarrow{\$} \{0, 1\}^r$.
 2. Run the bounded degree encryption algorithm $\text{BdFE}.\text{Enc}$ to produce a ciphertext $\text{CT} \leftarrow \text{BdFE}.\text{Enc}(\text{MPK}, (M, R_1, \dots, R_S))$.
 Output CT as the ciphertext.
- **Decryption** $\text{FE}.\text{Dec}(\text{SK}_K, \text{CT})$: On input a secret key SK_K and a ciphertext CT ,
 - Run the bounded FE decryption algorithm to get $\tilde{Y} \leftarrow \text{BdFE}.\text{Dec}(\text{SK}_K, \text{CT})$.
 - Run the randomized encoding decoder on \tilde{Y} to get the output $Y \leftarrow \text{RE}.\text{Decode}(\tilde{Y})$.

Correctness and Security. We first show correctness of the scheme \mathcal{FE} . Given a secret key SK_K and a ciphertext $\text{CT} \leftarrow \text{FE}.\text{Enc}(\text{MPK}, M)$, the decryption algorithm computes

$$\tilde{Y} = \text{BdFE}.\text{Dec}(\text{SK}_K, \text{CT}) = G((K, \Delta), (M, R_1, \dots, R_S)) = \tilde{F}(K, M; \bigoplus_{a \in \Delta} R_a)$$

Of course, running $\text{RE}.\text{Decode}$ on this should return $Y = F(K, M)$, by the correctness of the randomized encoding scheme.

The security of the scheme follows in a straightforward way from the security of \mathcal{BDFE} , and that of the randomized encoding.

Bootstrapping for Unbounded Queries. Although the transformation above assumes the knowledge of q (the bound on the number of secret key queries of the adversary), we can generalize it to work for unbounded queries as follows. Essentially, the idea is to generate fresh (computational) randomness for each randomized encoding using a pseudo-random function.

In particular, let $\{\text{prf}_S\}_{S \in \{0,1\}^\kappa}$ be a family of weak pseudo-random functions. Consider a function G that works in the following way:

$$G((K, R), (M, S)) := \tilde{F}\left(K, M; \text{prf}_S(R)\right)$$

Then, essentially the same construction as above works as a way to bootstrap an FE scheme for arbitrary circuits from FE schemes for circuits that can compute the weak PRF followed by the randomized encoding. Assuming the existence of weak PRFs and PRGs that can be computed by circuits of degree $\text{poly}(\kappa)$, we then obtain functional encryption schemes for arbitrary circuits.

References

- [AIK06] Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Computational Complexity 15(2), 115–162 (2006)
- [BF01] Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
- [BGW88] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 1–10. ACM, New York (1988)
- [BMR90] Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: STOC, pp. 503–513 (1990)
- [BSW11] Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
- [BV11] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2011)
- [BW06] Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
- [CFGN96] Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multiparty computation. In: STOC, pp. 639–648 (1996), Longer version at <http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-682.pdf>
- [CHH⁺07] Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-Secure Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)
- [Coc01] Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
- [DI05] Damgård, I., Ishai, Y.: Constant-Round Multiparty Computation Using a Black-Box Pseudorandom Generator. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 378–394. Springer, Heidelberg (2005)
- [DKXY02] Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
- [DN00] Damgård, I., Nielsen, J.B.: Improved Non-committing Encryption Schemes Based on a General Complexity Assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)

- [Gen09] Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), <http://crypto.stanford.edu/craig>
- [GLW12] Goldwasser, S., Lewko, A., Wilson, D.A.: Bounded-Collusion IBE from Key Homomorphism. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 564–581. Springer, Heidelberg (2012)
- [GPSW06] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
- [IK00] Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: FOCS, pp. 294–304 (2000)
- [KO04] Katz, J., Ostrovsky, R.: Round-Optimal Secure Two-Party Computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
- [KSW08] Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
- [LOS⁺10] Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
- [O’N10] O’Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010), <http://eprint.iacr.org/>
- [OSW07] Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM Conference on Computer and Communications Security, pp. 195–203 (2007)
- [OT10] Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
- [Sha79] Shamir, A.: How to share a secret. Commun. ACM 22, 612–613 (1979)
- [Sha84] Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
- [SS10] Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: ACM Conference on Computer and Communications Security, pp. 463–472 (2010)
- [SW05] Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
- [Yao86] Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)

A One-Query General Functional Encryption from Randomized Encodings

We describe a construction of a one-query functional encryption scheme that is essentially from Sahai and Seyalioglu [SS10]. They proved the construction secure in the 1-NA-SIM sense; we observe that their “bootstrapping” construction works

for 1-AD-SIM. Let F be arbitrary polynomial-size functionality. We construct the scheme \mathcal{ONEQFE} for F as follows.

Our starting point is a “brute-force construction” first presented by Boneh et al. [BSW11, Section 4.1]. We call this \mathcal{BFFE} . Essentially, Boneh et al. presented an NA-SIM-secure scheme for any functionality where the key space has polynomial size, starting from any semantically secure public-key encryption scheme. (They only claimed indistinguishability-based security, but it clearly satisfies simulation-based security too.) For simplicity, we just use their construction for the key-space $\mathcal{K} = \{0, 1\}$. In addition, we observe that the scheme can be made AD-SIM-secure (for bounded message spaces) by replacing the underlying encryption scheme by an appropriate “non-committing type” scheme; the details are deferred to the full version.

In a high-level the idea is this: suppose we wish to construct an FE scheme for a polynomial-size circuit $F(K, M)$, and let $\tilde{F}(K, M; R)$ denote a randomized encoding of F where for every M, R , $\tilde{F}(\cdot, M; R)$ has small locality; specifically, every output bit of $\tilde{F}(K, M; R)$ depends only on one input bit of K . Assume the key has length λ . Then, we can write

$$\tilde{F}(K, M; R) = (\tilde{F}_1(K_1, M; R), \dots, \tilde{F}_\lambda(K_\lambda, M; R))$$

where $\tilde{F}_i(\cdot, M; R)$ depends only on K_i , the i th bit of K .

- **Setup** $\text{FE}.\text{Setup}(1^\kappa)$: Run the brute-force setup algorithm λ times to generate independent master public-key/secret-key pairs

$$(\text{MPK}_i, \text{MSK}_i) \leftarrow \text{BFFE}.\text{Setup}(1^\kappa) \quad \text{for } \tilde{F}_i \text{ and } i = 1, \dots, \lambda$$

Output $(\text{MPK}_i)_{i=1}^\lambda$ as the master public key and $(\text{MSK}_i)_{i=1}^\lambda$ as the master secret key.

- **Key Generation** $\text{FE}.\text{Keygen}(\text{MSK}, K)$: On input the master secret key MSK and a key $K \in \mathcal{K}$ for the functionality, pick

$$\text{SK}_{K,i} \leftarrow \text{BFFE}.\text{Keygen}(\text{MSK}_i, K_i) \quad \text{for } i = 1, \dots, \lambda$$

Output as secret key $\text{SK}_K := ((\text{SK}_{K,i})_{i \in [\lambda]})$.

- **Encryption** $\text{FE}.\text{Enc}(\text{MPK}, M)$: On input the master public key MPK and a message $M \in \mathcal{M}$, compute

$$\text{CT}_i \leftarrow \text{BFFE}.\text{Enc}(\text{MPK}_i, M) \quad \text{for } i = 1, \dots, \lambda$$

Output $(\text{CT}_i)_{i=1}^\lambda$ as the ciphertext.

- **Decryption** $\text{FE}.\text{Dec}(\text{SK}_K, \text{CT})$: On input a secret key $\text{SK}_K = (\text{SK}_{K,i})_{i \in [\lambda]}$ and a ciphertext $\text{CT} = (\text{CT}_i)_{i=1}^\lambda$, do the following:

1. Compute $\tilde{Y}_i = \text{BFFE}.\text{Dec}(\text{MSK}_i, \text{CT}_i)$ for $i = 1, \dots, \lambda$;
2. Run the decoder to get $Y \leftarrow \text{RE}.\text{Decode}(\tilde{Y}_1, \dots, \tilde{Y}_\lambda)$.

Output Y .

New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques

Allison Lewko* and Brent Waters**

The University of Texas at Austin
`{alewko,bwaters}@cs.utexas.edu`

Abstract. We develop a new methodology for utilizing the prior techniques to prove selective security for functional encryption systems as a direct ingredient in devising proofs of full security. This deepens the relationship between the selective and full security models and provides a path for transferring the best qualities of selectively secure systems to fully secure systems. In particular, we present a Ciphertext-Policy Attribute-Based Encryption scheme that is proven fully secure while matching the efficiency of the state of the art selectively secure systems.

1 Introduction

Functional encryption presents a vision for public key cryptosystems that provide a strong combination of flexibility, efficiency, and security. In a functional encryption scheme, ciphertexts are associated with descriptive values x , secret keys are associated with descriptive values y , and a function $f(x, y)$ determines what a user with a key for value y should learn from a ciphertext with value x . One well-studied example of functional encryption is attribute-based encryption (ABE), first introduced in [30], in which ciphertexts and keys are associated with access policies over attributes and subsets of attributes. A key will decrypt a ciphertext if and only if the associated set of attributes satisfies the associated access policy. There are two types of ABE systems: Ciphertext-Policy ABE (CP-ABE), where ciphertexts are associated with access policies and keys are associated with sets of attributes, and Key-Policy ABE (KP-ABE), where keys

* Supported by a Microsoft Research PhD Fellowship.

** Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

are associated with access policies and ciphertexts are associated with sets of attributes.

To achieve desired flexibility, one strives to construct ABE systems for suitably expressive types of access policies over many attributes. Current constructions allow boolean formulas or linear secret sharing schemes as access policies. This high level of flexibility means that keys and ciphertexts have rich structure, and there is a very large space of possible access policies and attribute sets. This presents a challenge to proving security, since a suitable notion of security in this setting must enforce collusion resistance, meaning that several users should not be able to decrypt a message that none of them are individually authorized to read. Hence a security proof must consider an attacker who can collect many different keys, just not a single one that is authorized to decrypt the ciphertext.

This requires security reductions to balance two competing goals: the simulator must be powerful enough to provide the attacker with the many keys that it adaptively requests, but it must also lack some critical knowledge that it can gain from the attacker's success. The first security proofs in the standard model for ABE systems (e.g. [19, 30, 34]) followed a very natural paradigm for balancing these two goals known as partitioning. This proof technique was previously used in the context of identity-based encryption [6, 7, 9, 11, 32]. In a partitioning proof, the simulator sets up the system so that the space of all possible secret keys is partitioned into two pieces: keys that the simulator can make and those that it cannot. To ensure that the keys the attacker requests all fall in the set of keys the simulator can produce and that any key capable of decrypting the challenge ciphertext falls in the opposite set, the prior works [19, 30, 34] had to rely on a weaker security model known as *selective security*. In the selective security model, the attacker must declare up front what the challenge ciphertext will be, *before* seeing the public parameters.

This notion of selective security is quite useful as an intermediary step, but is rather unsatisfying as an end goal. In the setting of identity-based encryption, the need for selectivity was overcome by arranging for the simulator to “guess” a partition and abort when the attacker violated its constraints [32]. However, the richer structure of attribute-based systems appears to doom this approach to incur exponential loss, since one must guess a partition that respects the partial ordering induced by the powers allocated to the individual keys.

Dual System Encryption. With the goal of moving beyond the constraints of the partitioning paradigm, Waters introduced the dual system encryption methodology [33]. In a dual system security proof, the simulator is always prepared to make *any* key and *any* challenge ciphertext. The high level idea of the methodology is as follows. There are two types of keys and ciphertexts: normal and semi-functional. A key will decrypt a ciphertext properly unless *both* the key and the ciphertext are semi-functional, in which case decryption will fail with all but negligible probability. The normal keys and ciphertexts are used in the real system, while the semi-functional objects are gradually introduced in the hybrid security proof - first the ciphertext is changed from normal to semi-functional, and then the secret keys given to the attacker are changed from normal to

semi-functional one by one. Ultimately, we arrive at a security game in which the simulator only has to produce semi-functional objects and security can be proved directly.

The most critical step of the hybrid proof is when a key turns semi-functional: at this point, we must leverage the fact that the key is not authorized to decrypt the (now semi-functional) challenge ciphertext in order to argue that the attacker cannot detect the change in the key. However, since we are not imposing a partition on the simulator, there is no constraint preventing the simulator itself from creating a key that is authorized to decrypt and testing the nature of the key for itself by attempting to decrypt the semi-functional ciphertext. In the first application of dual system encryption to ABE [22], this paradox was averted by ensuring that the simulator could only produce a key that would be *correlated* with the semi-functional ciphertext so that decryption would succeed in the simulator's view, regardless of the presence or absence of semi-functionality. This correlation between a semi-functional key and semi-functional ciphertext was called *nominal semi-functionality*. It was argued that this correlation was hidden information-theoretically from the attacker, who cannot request keys authorized to decrypt the challenge ciphertext. This provided the first proof of full security for an ABE scheme in the standard model.

The One-Use Restriction. The information-theoretic argument in [22] required a one-use restriction on attributes in access formulas/LSSS matrices, meaning that a single attribute could only be used once in a policy. This can be extended to a system which allows reuse of attributes by setting a fixed bound M on the maximum number of times an attribute may be used and having separate parameters for each use. This scales the size of the public parameters by M , as well as the size of secret keys for CP-ABE systems¹. This approach incurs a very significant loss in efficiency, and has been inherited by all fully secure schemes to date ([24, 28] employ the same technique). This loss in efficiency is costly enough to limit the potential applications of fully secure schemes. As an example, the recent work of [2] building verifiable computation schemes from KP-ABE systems only produces meaningful results when one starts with a KP-ABE scheme that can be proven secure *without* incurring the blowup of this encoding technique.

Our work eliminates this efficiency loss and allows unrestricted use of attributes while still proving full security in the standard model. Our main observation is motivated by the intuition that the information-theoretic step of the prior dual system proof is ceding too much ground to the attacker, since a computational argument would suffice. In fact, we are able to resurrect the earlier selective proof techniques inside the framework of dual system encryption in order to retake ground and obtain a wholly computational proof of full security.

Our Techniques. Dual system encryption is typically implemented by designing a “semi-functional space” where semi-functional components of keys and ciphertexts will behave like a parallel copy of the normal components of the system,

¹ For KP-ABE systems, the ciphertext size instead will grow multiplicatively with M .

except divorced from the public parameters. This provides a mechanism allowing for *delayed parameters* in the semi-functional space, meaning that relevant variables can be defined later in the simulation instead of needing to be fixed in the setup phase. The hybrid structure of a dual system encryption argument is implemented by additionally providing a mechanism for *key isolation*, meaning that some or all of the semi-functional parameters will only be relevant to the distribution of a single semi-functional key at a time.

In combination, these two mechanisms mean that the semi-functional space has its own fresh parameters that can be decided on the fly by the simulator when they become relevant, and they are only relevant for the semi-functional ciphertext and a single semi-functional key. Previous dual system encryption arguments have used the isolated use of these delayed semi-functional parameters as a source of entropy in the attacker's view to make an information-theoretic argument. We observe that these mechanisms can also be used to implement prior techniques for selective security proofs, without needing to impose the selective restriction on the attacker.

To explain this more precisely, we consider the critical step in the hybrid security proof when a particular key becomes semi-functional. We conceptualize the unpublished semi-functional parameters as being defined belatedly when the simulator first issues either the key in question or the semi-functional ciphertext. For concreteness, we consider a CP-ABE system. If the ciphertext is issued first, then the simulator learns the challenge policy *before* defining the delayed semi-functional parameters - this is closely analogous to the setting of selective security for a CP-ABE system. If the key is issued first, then the simulator learns the relevant set of attributes *before* defining the delayed semi-functional parameters, and this is closely analogous to the setting of selective security for a KP-ABE system. This provides us an opportunity to combine the techniques used to prove selective security for both CP-ABE and KP-ABE systems with the dual system encryption methodology in order to obtain a new proof of full security that maintains the efficiency of selectively secure systems.

Our Results. Since our approach utilizes selective techniques for both CP-ABE and KP-ABE in order to prove full security for either kind of system, we inherit the kinds of complexity assumptions needed to prove selective security in both settings. The KP-ABE scheme of [19] is proven selectively secure under the decisional bilinear Diffie-Hellman assumption, and so we are able to rely on the relatively simple 3-party Diffie-Hellman assumption for this part of our proof. The most efficient selectively secure CP-ABE scheme that is known is provided in [34], and it is proven secure under a q -based assumption (meaning that the number of terms in the assumption is parameterized by a value q that depends on the behavior of the attacker). Hence we inherit the need to rely on a q -based assumption in our security proof as well.

The dual system encryption methodology has previously been implemented both in prime order bilinear groups (e.g. in [28, 33]) and in composite order bilinear groups (e.g. in [22, 23]). The two settings provide different but roughly interchangeable mechanisms for executing delayed parameters and key isolation,

and our techniques are compatible with either setting. We first present a CP-ABE construction and security proof in composite order groups, relying on a few instances of the general subgroup decision assumption to execute the delayed parameters and key isolation mechanisms. In the full version, we also present an analogous CP-ABE construction and security proof in prime order groups, relying on the decisional linear assumption for these functions. To translate our construction from the composite order setting to the prime order setting, we employ the dual pairing vector space framework developed in [26–28], along with the relevant observations in [21]. The formal statements of our complexity assumptions for the composite order setting can be found in Section 2.1, while those for the prime order setting can be found in the full version. Though we present only CP-ABE schemes in this work, we expect that our techniques are equally applicable to the KP-ABE setting.

We view our work as providing a new view of the relationship between the selective and full security models, as we illustrate a methodology for using techniques in the selective context as direct building blocks for a full security proof. We suspect that any new improvements in selectively secure systems may now translate to improvements in the full security setting. In particular, a new proof of selective security for an efficient CP-ABE system relying on a static (non q-based) assumption could likely be combined with our techniques to yield a fully secure scheme of comparable efficiency under similar assumptions. This remains an important open problem.

Other Related Work. The roots of attribute-based encryption trace back to identity-based encryption (IBE), which was first conceived by Shamir [31] and then constructed by Boneh and Franklin [9] and Cocks [14]. This concept was extended to the notion of hierarchical identity-based encryption (HIBE) by Horwitz and Lynn [20], and this was first constructed by Gentry and Silverberg [17]. Subsequent constructions of IBE and HIBE can be found in [1, 6–8, 11, 12, 15, 16, 23, 25]. There have been several prior constructions of attribute-based encryption which have been shown to be selectively secure in the standard model [13, 18, 19, 29, 30, 34] or proven secure in the generic group model [5] (this is a heuristic model intended to capture an attacker who can only access group operations in a black-box fashion).

2 Preliminaries

Here we present the relevant background on composite order bilinear groups and state the complexity assumptions we use in this context. We also give background on LSSS access structures. Further background on CP-ABE systems and their formal security definition can be found in the full version.

2.1 Composite Order Bilinear Groups and Complexity Assumptions

We will first construct our system in composite order bilinear groups, which were introduced in [10]. We let \mathcal{G} denote a group generator - an algorithm which takes

a security parameter λ as input and outputs a description of a bilinear group G . We define \mathcal{G} 's output as (N, G, G_T, e) , where $N = p_1 p_2 p_3$ is a product of three distinct primes, G and G_T are cyclic groups of order N , and $e : G^2 \rightarrow G_T$ is a map such that:

1. (Bilinear) $\forall g, f \in G, a, b \in \mathbb{Z}_N, e(g^a, f^b) = e(g, f)^{ab}$
2. (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order N in G_T .

We refer to G as the *source group* and G_T as the *target group*. We assume that the group operations in G and G_T and the map e are computable in polynomial time with respect to λ , and the group descriptions of G and G_T include a generator of each group. We let G_{p_1} , G_{p_2} , and G_{p_3} denote the subgroups of order p_1 , p_2 , and p_3 in G respectively. We note that these subgroups are “orthogonal” to each other under the bilinear map e : if $f_i \in G_{p_i}$ and $f_j \in G_{p_j}$ for $i \neq j$, then $e(f_i, f_j)$ is the identity element in G_T . If g_1 generates G_{p_1} , g_2 generates G_{p_2} , and g_3 generates G_{p_3} , then every element f of G can be expressed as $g_1^{c_1} g_2^{c_2} g_3^{c_3}$ for some values $c_1, c_2, c_3 \in \mathbb{Z}_N$. We will refer to $g_1^{c_1}$ as the “ G_{p_1} part of f ”, for example.

We now present the complexity assumptions we will use in composite order bilinear groups. We use the notation $X \xleftarrow{R} S$ to express that X is chosen uniformly randomly from the finite set S . We will consider groups G whose orders are products of three distinct primes. For any non-empty set $Z \subseteq \{1, 2, 3\}$, there is a corresponding subgroup of G of order $\prod_{i \in Z} p_i$. We denote this subgroup by G_Z . Our first assumption has been previously used in [22, 23], for example, and holds in the generic group model:

Assumption 1. Given a group generator \mathcal{G} , we define the following distribution:

$$\begin{aligned} \mathbb{G} &= (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, \\ g_1 &\xleftarrow{R} G_{p_1}, g_2, X_2, Y_2 \xleftarrow{R} G_{p_2}, g_3 \xleftarrow{R} G_{p_3}, \alpha, s \xleftarrow{R} \mathbb{Z}_N, \\ D &= (\mathbb{G}, g_1, g_2, g_3, g_1^\alpha X_2, g_1^s Y_2), T_0 = e(g_1, g_1)^{\alpha s}, T_1 \xleftarrow{R} G_T. \end{aligned}$$

We define the advantage of an algorithm \mathcal{A} in breaking this assumption to be:

$$Adv_{\mathcal{G}, \mathcal{A}}^1(\lambda) := |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|.$$

We say that \mathcal{G} satisfies Assumption 1 if $Adv_{\mathcal{G}, \mathcal{A}}^1(\lambda)$ is a negligible function of λ for any PPT algorithm \mathcal{A} .

We next define the General Subgroup Decision Assumption for composite order bilinear groups with three prime subgroups. This was first defined in [4] more generally for groups with an arbitrary number of prime order subgroups, but three will be sufficient for our purposes. We will only use a few specific instances of this assumption, but we prefer to state its full generality here for conciseness. We note that for our prime order construction, Assumption 1 and all instances of the General Subgroup Decision Assumption will be replaced by the Decisional Linear Assumption.

The General Subgroup Decision Assumption. We let \mathcal{G} denote a group generator and $Z_0, Z_1, Z_2, \dots, Z_k$ denote a collection of non-empty subsets of $\{1, 2, 3\}$ where each Z_i for $i \geq 2$ satisfies either $Z_0 \cap Z_i \neq \emptyset \neq Z_1 \cap Z_i$ or $Z_0 \cap Z_i = \emptyset = Z_1 \cap Z_i$. We define the following distribution:

$$\begin{aligned} \mathbb{G} &= (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, \\ g_{Z_2} &\xleftarrow{R} G_{Z_2}, \dots, g_{Z_k} \xleftarrow{R} G_{Z_k} \\ D &= (\mathbb{G}, g_{Z_2}, \dots, g_{Z_k}), T_0 \xleftarrow{R} G_{Z_0}, T_1 \xleftarrow{R} G_{Z_1}. \end{aligned}$$

Fixing the collection of sets Z_0, \dots, Z_k , we define the advantage of an algorithm \mathcal{A} in breaking this assumption to be:

$$Adv_{\mathcal{G}, \mathcal{A}}^{SD}(\lambda) := |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|.$$

We say that \mathcal{G} satisfies the General Subgroup Decision Assumption if $Adv_{\mathcal{G}, \mathcal{A}}^{SD}(\lambda)$ is a negligible function of λ for any PPT algorithm \mathcal{A} and any suitable collection of subsets Z_0, \dots, Z_k . This can be thought of as a family of assumptions, parameterized by the choice of the sets Z_0, \dots, Z_k . All of these individual assumptions hold in the generic group model, assuming it is hard to find a non-trivial factor of N . We will assume that $\frac{1}{p_i}$ is negligible in the security parameter for each prime factor p_i of N . In particular, this means we may assume (ignoring only negligible probability events) that when an element is randomly chosen from a subgroup of G , it is in fact a generator of that subgroup.

We next introduce an assumption that we call The Three Party Diffie-Hellman Assumption in a Subgroup. This is a close relative of the standard Decisional Bilinear Diffie-Hellman Assumption, but it has a challenge term remaining in the source group and takes place in a prime order subgroup of a composite order bilinear group. These adjustments from the usual DBDH assumption allow us to use our assumption in the semi-functional space for a particular key - without affecting the normal space or the other keys.

The Three Party Diffie-Hellman Assumption in a Subgroup. Given a group generator \mathcal{G} , we define the following distribution:

$$\begin{aligned} \mathbb{G} &= (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, \\ g_1 &\xleftarrow{R} G_{p_1}, g_2 \xleftarrow{R} G_{p_2}, g_3 \xleftarrow{R} G_{p_3}, x, y, z \xleftarrow{R} \mathbb{Z}_N, \\ D &= (\mathbb{G}, g_1, g_2, g_3, g_2^x, g_2^y, g_2^z), T_0 = g_2^{xyz}, T_1 \xleftarrow{R} G_{p_2}. \end{aligned}$$

We define the advantage of an algorithm \mathcal{A} in breaking this assumption to be:

$$Adv_{\mathcal{G}, \mathcal{A}}^{3DH}(\lambda) := |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|.$$

We say that \mathcal{G} satisfies The Three Party Diffie-Hellman Assumption if $Adv_{\mathcal{G}, \mathcal{A}}^{3DH}(\lambda)$ is a negligible function of λ for any PPT algorithm \mathcal{A} .

We next introduce a q -based assumption that we call The Source Group q -Parallel BDHE Assumption in a Subgroup. This is a close relative of The Decisional q -parallel Bilinear Diffie-Hellman Exponent Assumption introduced in [34], except that its challenge term remains in the source group and it takes place in a prime order subgroup of a composite order bilinear group. In the full version, we prove that the prime order variant of this assumption holds in the generic group model (the proof for this version follows analogously). Below, we use the notation $[q]$, for example, to denote the set $\{1, 2, \dots, q\}$.

The Source Group q -Parallel BDHE Assumption in a Subgroup. Given a group generator \mathcal{G} and a positive integer q , we define the following distribution:

$$\begin{aligned} \mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) &\xleftarrow{R} \mathcal{G}, \\ g_1 &\xleftarrow{R} G_{p_1}, g_2 \xleftarrow{R} G_{p_2}, g_3 \xleftarrow{R} G_{p_3}, c, d, f, b_1, \dots, b_q \xleftarrow{R} \mathbb{Z}_N, \end{aligned}$$

The adversary will be given:

$$\begin{aligned} D = (\mathbb{G}, g_1, g_3, g_2, g_2^f, g_2^{df}, g_2^c, g_2^{c^2}, \dots, g_2^{c^q}, g_2^{c^{q+2}}, \dots, g_2^{c^{2q}}, \\ g_2^{c^i/b_j} \forall i \in [2q] \setminus \{q+1\}, j \in [q], \\ g_2^{dfb_j} \forall j \in [q], g_2^{dfc^i b_{j'}/b_j} \forall i \in [q], j, j' \in [q] \text{ s.t. } j \neq j'). \end{aligned}$$

We additionally define

$$T_0 = g_2^{dc^{q+1}}, T_1 \xleftarrow{R} G_{p_2}.$$

We define the advantage of an algorithm \mathcal{A} in breaking this assumption to be:

$$Adv_{\mathcal{G}, \mathcal{A}}^q(\lambda) := |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|.$$

We say that \mathcal{G} satisfies The Source Group q -Parallel BDHE Assumption in a Subgroup if $Adv_{\mathcal{G}, \mathcal{A}}^q$ is a negligible function of λ for any PPT algorithm \mathcal{A} .

2.2 Access Structures

Definition 1. (*Access Structure [3]*) Let $\{P_1, \dots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \dots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) \mathbb{A} of non-empty subsets of $\{P_1, \dots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, \dots, P_n\}} \setminus \{\emptyset\}$. The sets in \mathbb{A} are called the authorized sets, and the sets not in \mathbb{A} are called the unauthorized sets.

In our setting, attributes will play the role of parties and we will consider only monotone access structures. One can (inefficiently) realize general access structures with our techniques by having the negation of an attribute be a separate attribute (so the total number of attributes doubles).

Linear Secret-Sharing Schemes Our construction will employ linear secret-sharing schemes (LSSS). We use the following definition adapted from [3].

Definition 2. (*Linear Secret-Sharing Schemes (LSSS)*) A secret sharing scheme Π over a set of parties \mathcal{P} is called linear (over \mathbb{Z}_p) if

1. The shares for each party form a vector over \mathbb{Z}_p .
2. There exists a matrix A called the share-generating matrix for Π . The matrix A has ℓ rows and n columns. For all $j = 1, \dots, \ell$, the j^{th} row of A is labeled by a party $\rho(j)$ (ρ is a function from $\{1, \dots, \ell\}$ to \mathcal{P}). When we consider the column vector $v = (s, r_2, \dots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \dots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Av is the vector of ℓ shares of the secret s according to Π . The share $(Av)_j$ belongs to party $\rho(j)$.

We note the *linear reconstruction* property: we suppose that Π is an LSSS for access structure \mathbb{A} . We let S denote an authorized set, and define $I \subseteq \{1, \dots, \ell\}$ as $I = \{j | \rho(j) \in S\}$. Then the vector $(1, 0, \dots, 0)$ is in the span of rows of A indexed by I , and there exist constants $\{\omega_j \in \mathbb{Z}_p\}_{j \in I}$ such that, for any valid shares $\{\lambda_j\}$ of a secret s according to Π , we have: $\sum_{j \in I} \omega_j \lambda_j = s$. These constants $\{\omega_j\}$ can be found in time polynomial in the size of the share-generating matrix A [3]. For unauthorized sets, no such constants $\{\omega_j\}$ exist. For our composite order group construction, we will employ LSSS matrices over \mathbb{Z}_N , where N is a product of three distinct primes.

3 CP-ABE Construction

We now present our CP-ABE scheme in composite order groups. This closely resembles the selectively secure CP-ABE scheme in [34], but with a one extra group element for each key and ciphertext. This extra group element is helpful in performing a cancellation during our security proof (when we are dealing with Phase II queries). We note that the freshly random exponent t for each key serves to prevent collusion, since it “ties” together the user’s attributes. Our main system resides in the G_{p_1} subgroup, while the G_{p_2} subgroup is reserved as the semi-functional space, and the G_{p_3} subgroup provides additional randomness on keys that helps to isolate keys in the hybrid argument. We assume that messages to be encrypted as elements of the target group G_T . The notation $[\ell]$ denotes the set $\{1, \dots, \ell\}$.

Setup(λ, \mathcal{U}) \rightarrow PP, MSK The setup algorithm chooses a bilinear group G of order $N = p_1 p_2 p_3$ (3 distinct primes). We let G_{p_i} denote the subgroup of order p_i in G . It then chooses random exponents $\alpha, a, \kappa \in \mathbb{Z}_N$, and a random group element $g \in G_{p_1}$. For each attribute $i \in \mathcal{U}$, it chooses a random value $h_i \in \mathbb{Z}_N$. The public parameters PP are $N, g, g^a, g^\kappa, e(g, g)^\alpha, H_i = g^{h_i} \forall i$. The master secret key MSK additionally contains g^α and a generator g_3 of G_{p_3} .

KeyGen(MSK, S , PP) \rightarrow SK The key generation algorithm chooses random exponents $t, u \in \mathbb{Z}_N$, and random elements $R, R', R'', \{R_i\}_{i \in S} \in G_{p_3}$ (this can be done by raising a generator of G_{p_3} to random exponents modulo N). The secret key is: S , $K = g^\alpha g^{at} g^{\kappa u} R$, $K' = g^u R'$, $K'' = g^t R''$, $K_i = H_i^t R_i \forall i \in S$.

Encrypt((A, ρ), PP, M) \rightarrow CT For A an $\ell \times n$ matrix and ρ a map from each row A_j of A to an attribute $\rho(j)$, the encryption algorithm chooses a random vector $v \in \mathbb{Z}_N^n$, denoted $v = (s, v_2, \dots, v_n)$. For each row A_j of A , it chooses a random $r_j \in \mathbb{Z}_N$. The ciphertext is (we also include (A, ρ) in the ciphertext, though we do not write it below):

$$C_0 = Me(g, g)^{\alpha s}, \quad C = g^s, \quad C' = (g^\kappa)^s, \quad C_j = (g^a)^{A_j \cdot v} H_{\rho(j)}^{-r_j}, \quad D_j = g^{r_j} \quad \forall j \in [\ell].$$

Decrypt(CT, PP, SK) $\rightarrow M$ For a secret key corresponding to an authorized set S , the decryption algorithm computes constants $\omega_j \in \mathbb{Z}_N$ such that $\sum_{\rho(j) \in S} \omega_j A_j = (1, 0, \dots, 0)$. It then computes:

$$e(C, K) e(C', K')^{-1} / \prod_{\rho(j) \in S} (e(C_j, K'') e(D_j, K_{\rho(j)}))^{\omega_j} = e(g, g)^{\alpha s}.$$

Then M can be recovered as $C_0/e(g, g)^{\alpha s}$.

Correctness We observe that $e(C, K) e(C', K')^{-1} = e(g, g)^{\alpha s} e(g, g)^{sat}$. For each j , $e(C_j, K'') e(D_j, K_{\rho(j)}) = e(g, g)^{at A_j \cdot v}$, so we have:

$$\prod_{\rho(j) \in S} (e(C_j, K'') e(D_j, K_{\rho(j)}))^{\omega_j} = e(g, g)^{at \sum_{\rho(j) \in S} \omega_j A_j \cdot v} = e(g, g)^{sat}.$$

4 Security Proof

We now prove the following theorem:

Theorem 1. *Under Assumption 1, the general subgroup decision assumption, the three party Diffie-Hellman assumption in a subgroup, and the source group q -parallel BDHE assumption in a subgroup defined in Section 2.1, our CP-ABE scheme defined in Section 3 is fully secure.*

Our security proof is obtained via a hybrid argument over a sequence of games. We let Game_{real} denote the real security game in the standard definition of full security for CP-ABE schemes (see the full version for a complete description). To describe the rest of the games, we must first define semi-functional keys and ciphertexts. We let g_2 denote a fixed generator of the subgroup G_{p_2} .

Semi-functional Keys. To produce a semi-functional key for an attribute set S , one first calls the normal key generation algorithm to produce a normal key consisting of $K, K', K'', \{K_i\}_{i \in S}$. One then chooses a random element $W \in G_{p_2}$ and forms the semi-functional key as: $KW, K', K'', \{K_i\}_{i \in S}$. In other words, all of the elements remain unchanged except for K , which is multiplied by a random element of G_{p_2} .

Semi-functional Ciphertexts. To produce a semi-functional ciphertext for an LSSS matrix (A, ρ) , one first calls the normal encryption algorithm to produce a normal ciphertext consisting of $C_0, C, C', \{C_j, D_j\}$. One then chooses random exponents $a', \kappa', s' \in \mathbb{Z}_N$, a random vector $w \in \mathbb{Z}_N^n$ with s' as its first entry, a random exponent $\eta_i \in \mathbb{Z}_N$ for each attribute i , and a random exponent $\gamma_j \in \mathbb{Z}_N$ for each $j \in [\ell]$. The semi-functional ciphertext is formed as: $C_0, Cg_2^{s'}, C'g_2^{s'\kappa'}, \{C_jg_2^{a'A_j \cdot w - \eta_{\rho(j)}\gamma_j}, D_jg_2^{\gamma_j}\}$.

We observe that the structure of the elements in G_{p_2} here is similar to the structure in G_{p_1} , but is unrelated to the public parameters. More specifically, s' plays the role of s , w plays the role of v , a' plays the role of a , κ' plays the role of κ , $\eta_{\rho(j)}$ plays the role of $h_{\rho(j)}$, and γ_j plays the role of r_j . While the values of a , κ , and the values $h_{\rho(j)}$ are determined modulo p_1 by the public parameters, the values of $a', \kappa', \eta_{\rho(j)}$ are freshly random modulo p_2 . These values $a', \kappa', \{\eta_i\}$ are chosen randomly once and then fixed - these same values will also be involved in additional types of semi-functional keys which we will define below.

We let Q denote the total number of key queries that the attacker makes. For each k from 0 to Q , we define Game_k as follows.

Game_k In this game, the ciphertext given to the attacker is semi-functional, as are the first k keys. The remaining keys are normal.

The outer structure of our hybrid argument will progress as follows. First, we transition from Game_{real} to Game_0 , then to Game_1 , next to Game_2 , and so on. We ultimately arrive at Game_Q , where the ciphertext and *all* of the keys given to the attacker are semi-functional. We then transition to Game_{final} , which is defined to be like Game_Q , except that the ciphertext given to the attacker is a semi-functional encryption of a *random message*. This will complete our security proof, since any attacker has a zero advantage in this final game.

The transitions from Game_{real} to Game_0 and from Game_Q to Game_{final} are relatively easy, and can be accomplished directly via computational assumptions. The transitions from Game_{k-1} to Game_k require more intricate arguments. For these steps, we will need to treat Phase I key requests (before the challenge ciphertext) and Phase II key requests (after the challenge ciphertext) differently. We will also need to define two additional types of semi-functional keys:

Nominal Semi-functional Keys. These keys will share the values a', κ', η_i modulo p_2 with the semi-functional ciphertext. To produce a nominal semi-functional key for an attribute set S , one first calls the normal key generation algorithm to produce a normal key consisting of $K, K', K'', \{K_i\}_{i \in S}$. One then chooses random exponents $t', u' \in \mathbb{Z}_N$ and forms the nominal semi-functional key as: $Kg_2^{a't' + \kappa'u'}, K'g_2^{u'}, K''g_2^{t'}, K_ig_2^{t'\eta_i} \forall i \in S$. We note that a nominal semi-functional key still correctly decrypts a semi-functional ciphertext, since the terms in the G_{p_2} will cancel out upon completion of the decryption algorithm.

Temporary Semi-functional Keys. These keys will still share the values η_i modulo p_2 with the semi-functional ciphertext, but the G_{p_2} component attached to K will now be randomized. More formally, to produce a temporary semi-functional

key for an attribute set S , one first calls the normal key generation algorithm to produce a normal key consisting of $K, K', K'', \{K_i\}_{i \in S}$. One then chooses a random $W \in G_{p_2}$ and random exponents $t', u' \in \mathbb{Z}_N$. The temporary semi-functional key is formed as: $KW, K'g_2^{u'}, K''g_2^{t'}, Kig_2^{t'\eta_i} \forall i \in S$.

For each k from 1 to Q , we define the following additional games:

Game $_k^N$ This is like Game $_k$, except that the k^{th} key given to the attacker is a nominal semi-functional key. The first $k - 1$ keys are still semi-functional in the original sense, while the remaining keys are normal.

Game $_k^T$ This is like Game $_k$, except that the k^{th} key given to the attacker is a temporary semi-functional key. The first $k - 1$ keys are still semi-functional in the original sense, while the remaining keys are normal.

The fact that the values a', κ', η_i are shared among semi-functional ciphertexts, nominal semi-functional keys, and temporary semi-functional keys means that these values are fixed whenever they first appear in a security game. This could be when the semi-functional ciphertext is generated, when a nominal semi-functional key is generated, or in the case of the η_i values, when a temporary semi-functional key is generated. The structure of temporary semi-functional keys is designed to fit the outcome of applying selective proof techniques to a single key and ciphertext pair within our hybrid game sequence.

In order to get from Game $_{k-1}$ to Game $_k$ in our hybrid argument, we will transition first from Game $_{k-1}$ to Game $_k^N$, then to Game $_k^T$, and finally to Game $_k$. The transition from Game $_k^N$ to Game $_k^T$ will require different computational assumptions for Phase I and Phase II key queries. We let Q_1 denote the number of Phase I queries, and we will address this transition separately for $k \leq Q_1$ and $k > Q_1$. Our handling of Phase I queries will closely resemble the selective security proof strategy for KP-ABE in [19], while our handling of Phase II queries will closely resemble the selective security proof strategy for CP-ABE in [34].

The original versions of these arguments in [19, 34] relied on assumptions very close to ours, with the main difference being that the assumptions in [19, 34] had challenge terms in the target group G_T instead of G . This is because the selective security arguments could afford to deal with all keys at once, and hence could use an assumption with a challenge in the target group to change the ciphertext to an encryption of a random message. This kind of change simultaneously affects the interaction of the ciphertext with *all* keys. In our hybrid framework, we need to handle keys individually, and hence we use an assumption with a challenge in the source group to change the nature of individual keys one at a time, saving our progress incrementally until we arrive at the final step and can afford to change to an encryption of a random message.

Our hybrid argument is accomplished in the following lemmas. Due to space constraints, some of the more standard lemma proofs are omitted here, but can be found in the full version.

Lemma 1. *Under the general subgroup decision assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game $_{real}$ and Game 0 .*

Lemma 2. *Under the general subgroup decision assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{k-1} and Game_k^N for any k from 1 to Q.*

The proofs of these first two lemmas can be found in the full version.

Lemma 3. *Under the three party Diffie-Hellman assumption in a subgroup (and assuming it is hard to find a non-trivial factor of N), no polynomial time attacker can achieve a non-negligible difference in advantage between Game_k^N and Game_k^T for an k from 1 to Q₁ (recall these are all the Phase I queries).*

Proof. Given a PPT attacker \mathcal{A} achieving a non-negligible difference in advantage between Game_k^N and Game_k^T for some k between 1 and Q_1 , we will create a PPT algorithm \mathcal{B} to break the three party Diffie-Hellman assumption in a subgroup. \mathcal{B} is given $g_1, g_2, g_3, g_2^x, g_2^y, g_2^z, T$, where T is either g_2^{xyz} or a random element of G_{p_2} . \mathcal{B} will simulate either Game_k^N or Game_k^T with \mathcal{A} depending on the nature of T .

\mathcal{B} first chooses random exponents $\alpha, a, \kappa, \{h_i\} \in \mathbb{Z}_N$ and sets the public parameters as: $\text{PP} = \{N, g = g_1, g^a = g_1^a, g^\kappa = g_1^\kappa, e(g_1, g_1)^\alpha, H_i = g_1^{h_i} \forall i\}$. It gives these to \mathcal{A} . We note that \mathcal{B} knows the MSK, and hence can use the normal key generation algorithm to make normal keys in response to \mathcal{A} 's key requests from the $k+1$ request and onward. To respond to \mathcal{A} 's first $k-1$ key requests, \mathcal{B} creates semi-functional keys by first creating a normal key and then multiplying K by a random element of G_{p_2} (this can be obtained by raising the generator g_2 to a random exponent modulo N).

We let S denote the attribute set requested in the k^{th} key query by \mathcal{A} . Since we are assuming the k^{th} key query occurs in Phase I, S is declared *before* \mathcal{B} must produce the challenge ciphertext. This allows \mathcal{B} to define the values η_i modulo p_2 to be shared by the k^{th} key and the semi-functional ciphertext *after* learning the set S . To set these values, \mathcal{B} chooses random exponents $\eta_i \in \mathbb{Z}_N$ for each $i \in S$. For $i \notin S$, it implicitly sets η_i modulo p_2 to be equal to $x\tilde{\eta}_i$ modulo p_2 , where random exponents $\tilde{\eta}_i \in \mathbb{Z}_N$ are chosen for each $i \notin S$. It also implicitly sets a' equal to xy modulo p_2 .

To form the k^{th} key, \mathcal{B} first calls the normal key generation algorithm to produce a normal key, $K, K', K'', \{K_i\}_{i \in S}$. It then chooses random exponents $\kappa', u' \in \mathbb{Z}_N$ and implicitly sets t' modulo p_2 equal to z modulo p_2 . It sets the key as: $Kg_2^{\kappa' u'} T, K'g_2^{u'}, K''g_2^z, K_i = (g_2^z)^{\eta_i} \forall i \in S$. We observe that if $T = g_2^{xyz}$, this is a properly distributed nominal semi-functional key, and when T is random in G_{p_2} , this is a properly distributed temporary semi-functional key.

To create the semi-functional challenge ciphertext for an $\ell \times n$ access matrix (A, ρ) and message M_b , \mathcal{B} first runs the normal encryption algorithm to produce a normal ciphertext, $C_0, C, C', \{C_j, D_j\}_{j \in [\ell]}$. We note the attribute set S cannot satisfy the access policy of (A, ρ) . As a result, \mathcal{B} can efficiently find a vector $\tilde{w} \in \mathbb{Z}_N^n$ such that $\tilde{w} \cdot A_j = 0$ modulo N for all j such that $\rho(j) \in S$ and the first entry of \tilde{w} is nonzero modulo each prime dividing N . Such a vector will exist as long as $(1, 0, \dots, 0)$ is not in the span of $\{A_j\}_{\rho(j) \in S}$ modulo each of p_1, p_2, p_3 . We may assume this holds with all but negligible probability, since

we are assuming it is hard to find a non-trivial factor of N . This vector \tilde{w} can be efficiently found by performing row reduction modulo N (we note that if one encounters a nonzero, non-invertible element of N during this process, then one has found a nontrivial factor of N). Once \tilde{w} is found, its first entry can be randomized by multiplying the vector by a random value modulo N . Thus, we may assume the first entry of \tilde{w} is random modulo p_2 . We call this first entry s' .

\mathcal{B} also chooses a random vector $w' \in \mathbb{Z}_N^n$ with first entry equal to 0. It will implicitly set the sharing vector w modulo p_2 so that $a'w = xy\tilde{w} + w'$ (i.e. $w = \tilde{w} + (xy)^{-1}w'$). We note that w is randomly distributed since the first entry of \tilde{w} is random and the remaining entries of w' are random. \mathcal{B} also chooses random values $\gamma_j \in \mathbb{Z}_N$ for each j such that $\rho(j) \in S$, and random values $\tilde{\gamma}_j \in \mathbb{Z}_N$ for each j such that $\rho(j) \notin S$. For these j 's such that $\rho(j) \notin S$, it will implicitly set $\gamma_j = y\tilde{\eta}_{\rho(j)}^{-1}A_j \cdot \tilde{w} + \tilde{\gamma}_j$. We note that all of these values are properly distributed modulo p_2 .

It forms the semi-functional ciphertext as:

$$\begin{aligned} C_0, \quad &Cg_2^{s'}, \quad C'g_2^{s'\kappa'}, \quad C_jg_2^{A_j \cdot w'}g_2^{-\eta_{\rho(j)}\gamma_j}, \quad D_jg_2^{\gamma_j} \quad \forall j \text{ s.t. } \rho(j) \in S, \\ &C_jg_2^{A_j \cdot w'}(g_2^x)^{-\tilde{\eta}_{\rho(j)}\tilde{\gamma}_j}, \quad D_j(g_2^y)^{\tilde{\eta}_{\rho(j)}^{-1}A_j \cdot \tilde{w}}g_2^{\tilde{\gamma}_j} \quad \forall j \text{ s.t. } \rho(j) \notin S. \end{aligned}$$

To see that this is a properly formed semi-functional ciphertext, note that for j such that $\rho(j) \notin S$:

$$a'A_j \cdot w - \eta_{\rho(j)}\gamma_j = A_j \cdot (xy\tilde{w} + w') - x\tilde{\eta}_{\rho(j)}(y\tilde{\eta}_{\rho(j)}^{-1}A_j \cdot \tilde{w} + \tilde{\gamma}_j) = A_j \cdot w' - x\tilde{\eta}_{\rho(j)}\tilde{\gamma}_j.$$

Here, \mathcal{B} has embedded a y into the γ_j term and used the x embedded in the $\eta_{\rho(j)}$ term to cancel out the xy term in $a'A_j \cdot w$ that it cannot produce.

When $T = g_2^{xyz}$, \mathcal{B} has properly simulated Game_k^N , and when T is random in G_{p_2} , \mathcal{B} has properly simulated Game_k^T . Hence \mathcal{B} can leverage \mathcal{A} 's non-negligible difference in advantage between these games to achieve a non-negligible advantage against the three party Diffie-Hellman assumption in a subgroup.

Lemma 4. *Under the source group q -parallel BDHE assumption in a subgroup (and assuming it is hard to find a non-trivial factor of N), no polynomial time attacker can achieve a non-negligible difference in advantage between Game_k^N and Game_k^T for a $k > Q_1$ using an access matrix (A, ρ) of size $\ell \times n$ where $\ell, n \leq q$.*

Proof. Given a PPT attacker \mathcal{A} achieving a non-negligible difference in advantage between Game_k^N and Game_k^T for some k such that $Q_1 < k \leq Q$ using an access matrix with dimensions $\leq q$, we will create a PPT algorithm \mathcal{B} to break the source group q -parallel BDHE assumption in a subgroup. Our \mathcal{B} is given: $g_1, g_3, g_2, g_2^f, g_2^{df}, g_2^{c^i} \forall i \in [2q] \setminus \{q+1\}$, $g_2^{c^i/b_j} \forall i \in [2q] \setminus \{q+1\}, j \in [q]$, $g_2^{dfb_j} \forall j \in [q]$, $g_2^{dfc^i b_{j'}/b_j} \forall i \in [q], j, j' \in [q]$ such that $j \neq j'$, and T , where T is either equal to $g_2^{dc^{q+1}}$ or is a random element of G_{p_2} . \mathcal{B} will simulate either Game_k^N or Game_k^T with \mathcal{A} , depending on T .

\mathcal{B} chooses random exponents $\alpha, a, \kappa, \{h_i\} \in \mathbb{Z}_N$, and sets the public parameters as $\text{PP} = \{N, g = g_1, g^a = g_1^a, g^\kappa = g_1^\kappa, e(g_1, g_1)^\alpha, H_i = g_1^{h_i} \forall i\}$. It gives these to \mathcal{A} . We note that \mathcal{B} knows the MSK, and hence can use the normal key generation algorithm to make normal keys in response to \mathcal{A} 's key requests from the $k+1$ request and onward. To make the first $k-1$ semi-functional keys, \mathcal{B} can first make a normal key and then multiply the K by a random element of G_{p_2} (this can be obtained by raising g_2 to a random exponent modulo N).

Since we are assuming the k^{th} key query is a Phase II key query, \mathcal{A} will request the challenge ciphertext for some $\ell \times n$ access matrix (A, ρ) before requesting the k^{th} key. This allows \mathcal{B} to define the exponents $a', \kappa', \{\eta_i\}$ after seeing (A, ρ) . \mathcal{B} chooses random values $\tilde{\kappa}, \{\tilde{\eta}_i\} \in \mathbb{Z}_N$. It will implicitly set $a' = cd$ modulo p_2 and $\kappa' = d + \tilde{\kappa}$ modulo p_2 . For each attribute i , we let J_i denote the set of indices j such that $\rho(j) = i$. \mathcal{B} defines $g_2^{\eta_i}$ as:

$$g_2^{\eta_i} = g_2^{\tilde{\eta}_i} \prod_{j \in J_i} \left(g_2^{c/b_j} \right)^{A_{j,1}} \cdot \left(g_2^{c^2/b_j} \right)^{A_{j,2}} \cdots \left(g_2^{c^n/b_j} \right)^{A_{j,n}}.$$

We note that all of these terms $g_2^{c/b_j}, \dots, g_2^{c^n/b_j}$ are available to \mathcal{B} , since we are assuming $n, \ell \leq q$. We note that a' is uniformly random because d is random, κ' is randomized by $\tilde{\kappa}$, and each η_i is randomized by $\tilde{\eta}_i$.

To form the challenge ciphertext, \mathcal{B} chooses random exponents $\{\tilde{\gamma}_j\} \in \mathbb{Z}_N$. It creates the normal components of the ciphertext as in the encryption algorithm. To create the semi-functional components (the parts in G_{p_2}), it implicitly sets $s' = f$ modulo p_2 and $\gamma_j = dfb_j + \tilde{\gamma}_j$ for each j from 1 to ℓ . We note that these values are properly distributed because $f, \tilde{\gamma}_j$ are random. It also chooses random values $y_2, \dots, y_n \in \mathbb{Z}_N$ and implicitly sets the sharing vector w as:

$$w := (f, fc + y_2(a')^{-1}, \dots, fc^{n-1} + y_n(a')^{-1}).$$

This is properly distributed as a random vector up to the constraint that the first entry is $s' = f$ (note that a' is nonzero with all but negligible probability).

For each j from 1 to ℓ , we observe that

$$a' A_j \cdot w - \eta_{\rho(j)} \gamma_j = df(cA_{j,1} + \dots + c^n A_{j,n}) + y_2 A_{j,2} + \dots + y_n A_{j,n} \quad (1)$$

$$-dfb_j \left(\sum_{j' \in J_{\rho(j)}} cA_{j',1}/b_{j'} + \dots + c^n A_{j',n}/b_{j'} \right) \quad (2)$$

$$-dfb_j \tilde{\eta}_{\rho(j)} - \tilde{\gamma}_j \tilde{\eta}_{\rho(j)} - \tilde{\gamma}_j \left(\sum_{j' \in J_{\rho(j)}} cA_{j',1}/b_{j'} + \dots + c^n A_{j',n}/b_{j'} \right) \quad (3)$$

Since $j \in J_{\rho(j)}$, the first quantity in (1) will be canceled by (2). What is left of (2) will be terms of the form $dfc^i b_j / b_{j'}$, where $i \leq n \leq q$ and $j \neq j'$. We note that \mathcal{B} is given all of these in the exponent of g_2 in the assumption. \mathcal{B} also has $g_2^{dfb_j}$ for all j from 1 to $q \geq \ell$ and $g_2^{c^i/b_{j'}}$ for all $j' \in J_{\rho(j)}, i \leq n \leq q$. Thus, \mathcal{B} can

form $g_2^{a' A_j \cdot w - \eta_{\rho(j)} \gamma_j}$ for each j . It can also compute $g_2^{s'} = g_2^f$, $g_2^{s' \kappa'} = g_2^{df} (g_2^f)^{\tilde{\kappa}}$, and $g_2^{\gamma_j} = g_2^{df b_j} g_2^{\tilde{\gamma}_j}$. \mathcal{B} multiplies these G_{p_2} components by the normal ciphertext to produce the semi-functional ciphertext, which it gives to \mathcal{A} .

Now, when \mathcal{A} later requests the k^{th} key for some attribute set S not satisfying (A, ρ) , \mathcal{B} responds as follows. It first creates a normal key by calling the usual key generation algorithm. To create the semi-functional components, it first chooses a vector $\theta = (\theta_1, \dots, \theta_n) \in \mathbb{Z}_N^n$ such that $\theta \cdot A_j = 0$ modulo N for all j such that $\rho(j) \in S$ and the first entry of θ is nonzero modulo each prime dividing N . Such a vector will exist as long as $(1, 0, \dots, 0)$ is not in the span of $\{A_j\}_{\rho(j) \in S}$ modulo each of p_1, p_2, p_3 . As in the proof of the previous lemma, we may assume this holds with all but negligible probability and we note that such a θ can be efficiently computed.

\mathcal{B} chooses a random value $\tilde{u} \in \mathbb{Z}_N$ and implicitly sets

$$\begin{aligned} u' &= -\theta_2 c^q - \theta_3 c^{q-1} - \dots - \theta_n c^{q-n+2} + f\tilde{u}, \\ t' &= \theta_1 c^q + \theta_2 c^{q-1} + \dots + \theta_n c^{q-n+1}. \end{aligned}$$

We note that these are random modulo p_2 because \tilde{u} and θ_1 are random (and c, f are nonzero with all but negligible probability). We observe that \mathcal{B} can now form $g_2^{u'}$ and $g_2^{t'}$ as follows:

$$\begin{aligned} g_2^{u'} &= \left(g_2^{c^q}\right)^{-\theta_2} \left(g_2^{c^{q-1}}\right)^{-\theta_3} \dots \left(g_2^{c^{q-n+2}}\right)^{-\theta_n} \left(g_2^f\right)^{\tilde{u}}, \\ g_2^{t'} &= \left(g_2^{c^q}\right)^{\theta_1} \left(g_2^{c^{q-1}}\right)^{\theta_2} \dots \left(g_2^{c^{q-n+1}}\right)^{\theta_n}. \end{aligned}$$

For each attribute $i \in S$, we recall that the vector θ is orthogonal to A_j for all rows j such that $\rho(j) = i$ (i.e. all $j \in J_i$). Thus, we observe:

$$t' \eta_i = t' \tilde{\eta}_i + \sum_{j \in J_i} \sum_{\substack{m_1, m_2=1 \\ m_1 \neq m_2}}^n \theta_{m_1} A_{j, m_2} b_j^{-1} c^{q+1+m_2-m_1}.$$

Since $q+1+m_2-m_1$ is always in the set $[2q] \setminus \{q+1\}$, \mathcal{B} can compute $g_2^{t' \eta_i}$ from the terms it is given in the assumption. We also have that

$$a't' + k'u' = \theta_1 dc^{q+1} - \tilde{\kappa} (\theta_2 c^q + \dots + \theta_n c^{q-n+2}) + df\tilde{u} + f\tilde{\kappa}\tilde{u}.$$

Therefore, \mathcal{B} creates the semi-functional term for key component K as:

$$T^{\theta_1} \left(g_2^{c^q}\right)^{-\tilde{\kappa}\theta_2} \dots \left(g_2^{c^{q-n+2}}\right)^{-\tilde{\kappa}\theta_n} \left(g_2^{df}\right)^{\tilde{u}} \left(g_2^f\right)^{\tilde{\kappa}\tilde{u}}.$$

If $T = g_2^{dc^{q+1}}$, then this is a properly distributed nominal semi-functional key. If T is a random element of G_{p_2} , this is a properly distributed temporary semi-functional key. Hence, \mathcal{B} has properly simulated either Game_k^N or Game_k^T , depending on T , and can therefore leverage \mathcal{A} 's non-negligible difference in advantage to break the source group q -parallel BDHE assumption in a subgroup.

Lemma 5. *Under the general subgroup decision assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_k^T and Game_k for any k from 1 to Q .*

Lemma 6. *Under Assumption 1, no polynomial attacker can achieve a non-negligible difference in advantage between Game_Q and $\text{Game}_{\text{final}}$.*

The proofs of these last two lemmas can be found in the full version. This completes the proof of Theorem 1.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)
2. Parno, B., Raykova, M., Vaikuntanathan, V.: How to Delegate and Verify in Public: Verifiable Computation from Attribute-Based Encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)
3. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
4. Bellare, M., Waters, B., Yilek, S.: Identity-Based Encryption Secure against Selective Opening Attack. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 235–252. Springer, Heidelberg (2011)
5. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 321–334
6. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
8. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
9. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
10. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
11. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
12. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
13. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: CCS, pp. 456–465 (2007)

14. Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
15. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
16. Gentry, C., Halevi, S.: Hierarchical Identity Based Encryption with Polynomially Many Levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)
17. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
18. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded Ciphertext Policy Attribute Based Encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórssón, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
19. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
20. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
21. Lewko, A.: Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
22. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
23. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
24. Lewko, A., Waters, B.: Decentralizing Attribute-Based Encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)
25. Lewko, A., Waters, B.: Unbounded HIBE and Attribute-Based Encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011)
26. Okamoto, T., Takashima, K.: Homomorphic Encryption and Signatures from Vector Decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008)
27. Okamoto, T., Takashima, K.: Hierarchical Predicate Encryption for Inner-Products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
28. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
29. Ostrovksy, R., Sahai, A., Waters, B.: Attribute based encryption with non-monotonic access structures. In: ACM Conference on Computer and Communications Security, pp. 195–203 (2007)
30. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

31. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
32. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
33. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
34. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)

Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption

Amit Sahai^{1,*}, Hakan Seyalioglu^{2,**}, and Brent Waters^{3,***}

¹ Department of Computer Science, UCLA

² Department of Mathematics, UCLA

³ Department of Computer Science, University of Texas at Austin

Abstract. Motivated by the question of access control in cloud storage, we consider the problem using Attribute-Based Encryption (ABE) in a setting where users' credentials may change and ciphertexts may be stored by a third party. Our main result is obtained by pairing two contributions:

- We first ask how a third party who is not trusted with secret key information can process a ciphertext to disqualify revoked users from decrypting data encrypted in the past. Our core tool is a new procedure called *ciphertext delegation* that allows a ciphertext to be ‘re-encrypted’ to a more restrictive policy using only public information.
- Second, we study the problem of revocable attribute-based encryption. We provide the first fully secure construction by modifying an attribute-based encryption scheme due to Lewko et al. [9] and prove security in the standard model.

We then combine these two results for a new approach for revocation on stored data. Our scheme allows a storage server to update stored ciphertexts to disqualify revoked users from accessing data that was encrypted before the user's access was revoked while key update broadcasts can dynamically revoke selected users.

* Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

** Research supported by a NSF Graduate Research Fellowship.

*** Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

1 Introduction

The need to store information externally has never been higher: With users and organizations expecting to access and modify information across multiple platforms and geographic locations, there are numerous advantages to storing data *in the cloud*. However, there is a natural resistance to the idea of handing over sensitive information to external storage. Since these databases are often filled with valuable data, they are high value targets for attackers and security breaches in such systems are not uncommon, especially by insiders. In addition, organizations with access to extremely sensitive data might not want to give an outside server any access to their information at all. Similar problems can easily arise when dealing with centralized storage within a single organization, where users in different departments have access to varying levels of sensitive data.

A first step in addressing this problem of trust is to only store information in encrypted form. However, data access is not static – as employees are hired, fired or promoted, it will be necessary to change who can access certain data. A natural solution to this problem is to have users authenticate their credentials before giving them access to data; but such an approach requires a great deal of trust in the server: a malicious party may be able to penetrate the server and bypass authentication by exploiting software vulnerabilities. A solution that avoids this problem is to use cryptographically enforced access control such as attribute-based encryption (ABE) [18]. However, this fails to address the problem that the credentials of a user may change with time. This problem motivated the study of revocation [3] where a nightly key update would only allow non-revoked users to update their keys to decrypt newly encrypted data. Dynamic credentials in the context of stored data, however, present novel challenges that have not been considered in previous study on revocation. Take the following example.

A MOTIVATING STORY. Consider an employee with access to sensitive documents necessary for his work¹. One day, this employee is terminated and has his access revoked. Now, this employee with insider knowledge of the organization’s systems, and who has retained his old key, may attempt to penetrate the database server and decrypt all the files that he once had access to. How can we deal with this type of attack? At first glance, there appears to be an inherent intractability to this problem. Any encrypted file that the user could decrypt with his old key will still be decryptable, after all.

Despite these problems, we believe this situation presents a very real security threat to the organization and is important to address. One method to handle this problem is to decrypt and re-encrypt all stored data every time some employee’s credentials are revoked. However, the involvement of secret key information in this process both makes this process cumbersome and opens up the overall system to problematic new vulnerabilities. In general, we want to limit the use of secret key information to *only* key generation and not to database

¹ While gainfully employed, the worker may have incentives to exercise discretion by only accessing the files necessary for his work and not download all files he has access to. Such discretion may be enforced, for example, through access logs.

upkeep as the database is handling constant two way communication in our system and is therefore modeled as the most vulnerable party.

We propose a novel method to deal with this problem: We devise a revocable ABE system where the database, using only *publicly available information*, can periodically update the ciphertexts stored on the system, so that as soon as access is revoked for a user, all stored files (no matter how old) immediately become inaccessible to this user after the update process. The database does not even need to know the identities of users whose access was revoked. We emphasize that this is a significant security improvement over decrypting and re-encrypting (which cannot be done with only public information) since in our solution, the database server *never* needs access to any secret keys. Furthermore, secret key holders do not have to interact with the database for the purpose of maintaining access control.

We also note in passing that while re-encrypting each ciphertext after every revocation (in a repeated nesting fashion) can also be applied to solve the problem of access control, this solution is *inefficient* when a ciphertext needs to be updated many times. In such a solution, decryption time will increase linearly and the ciphertext may grow significantly upon each invocation (Even using hybrid encryption, this would add a potentially large ABE header each time).

Our Results. In this work, we provide the first Revocable ABE scheme that deals with the problem of efficiently revoking stored data. This result is obtained through two main technical contributions:

REVOCABLE STORAGE ATTRIBUTE-BASED ENCRYPTION. We provide ABE encryption schemes with a new property we call *revocable storage*. Revocable storage allows a third party storing ciphertexts to revoke access on previously encrypted data. Additionally, our scheme satisfies strong efficiency guarantees with regard to the lifetime of the database.

We realize revocable storage by introducing a notion of **ciphertext delegation**. In ABE, ciphertext delegation is the ability to restrict a ciphertext with access policy P to a more restrictive policy P' using only *publicly available information*, but without causing the ciphertext size to increase. We initiate the first systematic study of ciphertext delegation for both Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) by analyzing the type of delegation possible in a variety of existing schemes [8,18,19,9].

PROTECING NEWLY ENCRYPTED DATA. To utilize revocable storage we need a method for efficiently revoking users' credentials such that newly encrypted data will not be decryptable by a user's key if that user's access has been revoked. This topic of revoking credentials was considered by Boldyreva et al. [3] in the context of Identity-Based Encryption (and to restricted notions of security for ABE); however was not paired with the revocation of ciphertexts. In addition, ours is the first fully (vs. selectively) secure system².

² A related work [17] proposes a ‘fully secure Revocable ABE scheme’ under a significantly different model than the one presented here.

While the Boldyreva et al. system needed to be proven “from scratch”, we provide a methodology to obtain a simple construction and proof. We propose a natural modification to standard ABE which we call *Piecewise Key Generation*. This requirement is similar to the standard ABE requirement but allows for an adversary to build his key ‘piece by piece’. Many existing proof methods extend with only minor modifications to prove piecewise security of existing schemes. We show that variants of the transformation method of Boldyreva et al. [3] succeed in converting *any* ABE scheme with piecewise key generation to a revocable ABE scheme. We give a modification of Lewko et al.’s fully secure ABE scheme [9] that satisfies our requirement and prove its security. Combined with our new techniques for dealing with revocable storage, this yields our Revocable Storage KP-ABE and CP-ABE schemes.

Related Work. Originally proposed by Sahai and Waters [18], *attribute-based encryption* [8,16,19,9,15] has been an active research area in cryptography in part since it is a primitive with interesting functional applications [7,3] and can be implemented efficiently [2]. In a key-policy attribute-based encryption (KP-ABE) scheme every secret key is generated with a policy P and ciphertexts are generated with a set of attributes U and decryption is only possible if $P(U) = \text{True}$. While the problem of delegating a key to a more restrictive key has been considered [8], it is analyzed only in the context of the scheme proposed in the paper. The problem of *revocation* is also a well studied problem, both for general PKI [11,14,1,12,13,5,6], identity based encryption [3,10] and attribute-based encryption [17]. At a high level, our revocable storage results can be seen as taking methods from forward secure encryption [4] which were introduced for key management and applying them to ciphertext management by noticing that the key delegation infrastructure can be replicated for the ciphertext through the delegation mechanism we introduce.

Roadmap. We briefly give an organizational outline to the paper. We begin with an introduction to preliminary notions and notation including formally defining attribute-based encryption and revocable ABE in Section 2. In Section 3 we define the piecewise key generation requirement that implies a revocable ABE scheme in a black-box fashion. In Section 4 we define revocable storage. Ciphertext delegation is defined and studied in Section 5. We give a technical lemma to efficiently handle time in our final construction in Section 6. We combine all our previously introduced tools to give our main construction of a revocable storage ABE scheme from an ABE scheme with piecewise key generation and ciphertext delegation in Section 7. We finally give a construction of an ABE scheme with piecewise key generation and ciphertext delegation in Section 8.

2 Preliminaries and Notation

We will assume $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear pairing whenever it is used. We use $[i, j]$ to denote the set of all integers from i to j inclusive or $[i]$ as shorthand for $[1, i]$. Throughout this paper, $\log(x)$ will denote the logarithm of x to the base 2. The notation $V(\mathcal{T})$ for a tree \mathcal{T} will denote the set of nodes of this

tree. The notation $x \leftarrow X$ for X a randomized algorithm may denote by context either that x is a possible output of that algorithm with positive probability or that x is drawn from the distribution X . We now briefly give the syntax for the two central notions for this chapter.

Attribute-Based Encryption. Attribute-based encryption schemes are generally divided into two types depending on if the access policy is embedded in the keys or ciphertexts.

Definition 1. (Key-Policy ABE) *A KP-ABE scheme with attribute set Ω that supports policies \mathcal{P} with message space \mathbb{M} is defined by the following polynomial time algorithms:*

- **Setup**(1^λ) $\rightarrow (PK, MSK)$: Setup takes as input the security parameter and outputs the public key and master secret key.
- **KeyGen**(MSK, P) $\rightarrow SK_P$: Key generation outputs a secret key with policy $P \in \mathcal{P}$.
- **Encrypt**(PK, M, S) $\rightarrow C_S$: Encrypts a message $M \in \mathbb{M}$ under the attribute set $S \subseteq \Omega$.
- **Decrypt**(SK_P, C_S) $\rightarrow M$: Decryption successfully recovers the encrypted message if and only if $P(S) = 1$ (i.e. the attribute set satisfies the policy).

The security game requires that an adversary queries keys corresponding to policies, and in the challenge phase requests the encryption of one of two adaptively chosen messages with an attribute set of its choice (generated during the challenge phase). The adversary succeeds if it correctly outputs the encrypted message without ever querying a key for a policy that is satisfied by the attribute set of the challenge ciphertext (we defer a formal security guarantee until we also introduce revocability). In a **Ciphertext-Policy ABE** (CP-ABE) scheme the placement of the policy is reversed, the key generation algorithm takes a set of attributes as input while encryption takes a policy. In these conference proceedings, we only detail the KP-ABE version of our results and delay the definition of revocable storage CP-ABE and our construction to the full version.

Revocable Attribute-Based Encryption. A revocable attribute-based encryption scheme [3] has the added functionality that a user may be placed on a revocation list that will make him unable to decrypt any message encrypted after he was revoked. Such a scheme has a periodic broadcast by a trusted key generation authority that allows the un-revoked users to update their keys and continue decryption. In such a scheme we assume that the total number of users and the number of key updates the scheme can survive are both very large and therefore the scheme's parameters should only depend polylogarithmically on the total number of non-revoked users and time bound. As in previous work, we will assume the key generation procedure is *stateful* for these constructions (it can store information used to create other keys in internal state denoted by σ).

Definition 2. (Revocable KP-ABE) *A Revocable KP-ABE scheme with attribute set Ω that supports policies \mathcal{P} with message space \mathbb{M} , time bound T and identity length \mathcal{I} consists of the following algorithms:*

- **Setup**(1^λ) $\rightarrow (PK, MSK, \sigma)$: Setup takes as input the security parameter and outputs the public key, master secret key and initializes $\sigma = \emptyset$.
- **KeyGen**(MSK, P, ID, σ) $\rightarrow (SK_{P, ID}, \sigma)$: Key generation outputs a secret key with policy $P \in \mathcal{P}$ for the user $ID \in \{0, 1\}^{\mathcal{I}}$ and updates the state σ .
- **Encrypt**(PK, M, S, t) $\rightarrow C_{S, t}$: Encrypts $M \in \mathbb{M}$ with attribute set $S \subseteq \Omega$ at time $t \leq T$.
- **KeyUpdate**(MSK, rl, t, σ) $\rightarrow (K_t, \sigma)$: The update phase takes as input a revocation list rl (a set of strings in $\{0, 1\}^{\mathcal{I}}$) and the master secret key, outputs the key update information for time t and updates the state σ .
- **Decrypt**($SK_{P, ID}, K_t, C_{S, t}$) $\rightarrow M$: Decryption successfully recovers the encrypted message if and only if $P(S) = 1$, $t' \geq t$ and ID was not revoked at time t (it was not present on rl when K_t was generated).

Security Game Oracles. Define the following oracles to use in the security game. These oracles are given access to (PK, MSK, σ) that are generated at the beginning of the security game, and may have been modified since, at the time of the oracle’s invocation.

1. The Secret Key Generation oracle $SK(\cdot, \cdot)$ takes as input (P, ID) and returns $SK_{P, ID}$ generated from: $(SK_{P, ID}, \sigma) \leftarrow \text{KeyGen}(MSK, P, ID, \sigma)$.
2. The Key Update Generation oracle $K(\cdot, \cdot)$ takes as input t and a revocation list rl and returns K_t generated from: $(K_t, \sigma) \leftarrow \text{KeyUpdate}(MSK, t, rl, \sigma)$.

Note that for both oracles σ is not sent to the adversary but is used to update the current σ value of the scheme. For a p.p.t. adversary A define the following experiment (some additional constraints on the adversary’s actions will be enumerated after the experiment’s definition). We use the term *challenger* for an internal agent in the security game who participates in the experiment.

RKP-SECURITY $_A(1^\lambda)$:

1. The challenger runs $\text{Setup}(1^\lambda) \rightarrow (PK, MSK, \sigma)$ and sends PK to A ;
2. A is given oracle access to $SK(\cdot, \cdot)$, $K(\cdot, \cdot)$ until it signals the query phase is over;
3. After the query phase, A returns (M_0, M_1, S^*, t^*) to the challenger;
4. The challenger picks b a random bit and returns to A :

$$C_{S^*, t^*} \leftarrow \text{Encrypt}(PK, M_b, S^*, t^*);$$

5. A is once again given oracle access to the two oracles;
6. A returns a bit b' . The experiment returns 1 if and only if $b' = b$ and the conditions below concerning A ’s query history are satisfied.

The conditions placed on the adversary’s queries are as follows: For any query, $SK(P, ID)$ such that $P(S^*) = 1$, $ID \in rl$ for every query $K(t, rl)$ with $t \geq t^*$.

Informally this corresponds to the fact that every user with sufficient credentials to decrypt the challenge ciphertext should be revoked by time t^* for the message to remain hidden.

Definition 3. *A Revocable KP-ABE scheme is secure if for any polynomial time adversary A the advantage of this adversary in the RKP-SECURITY game:*

$$2 \Pr [\text{RKP-SECURITY}_A(1^\lambda) = 1] - 1$$

is negligible in λ .

3 Revocable Attribute-Based Encryption

The Revocable ABE and IBE constructions given by Boldyreva et al. [3] are built from an underlying ABE scheme through a new transformation they introduce. However, security of the underlying ABE scheme does not imply security of the transformation, and their resulting scheme was proven secure (in the ABE case, in the restricted selective security model) from scratch. In this work we aim to both extend and simplify previous work by investigating the additional properties an ABE scheme needs to satisfy to imply a Revocable ABE scheme following their transformation. Using our result, we modify the fully secure scheme due to Lewko et al. [9] to satisfy our requirement in both the KP-ABE and CP-ABE setting. This yields the first fully secure Revocable KP-ABE and CP-ABE schemes.

The Requirement: Piecewise Key Generation. We find that the necessary condition an ABE scheme should satisfy in order to imply a revocable ABE scheme is that key generation can be done in a dual componentwise fashion. In the KP-ABE setting keys will have two separate policies P_0 and P_1 such that decryption succeeds for an encryption with attribute set S if and only if $P_0(S) = 1$ and $P_1(S) = 1$. The adversary in the security game is allowed to query these components separately with access to two oracles KeyGen_0 and KeyGen_1 which take as input a policy and an identifier U such that a key $\text{KeyGen}_0(\text{MSK}, P_0, U)$ and $\text{KeyGen}_1(\text{MSK}, P_1, U')$ can only be combined if $U = U'$ (in our applications this U value will be related to, but will not exactly be a user's identity. For this reason, we switch notation from identities ID to identifiers U at this point).

This security definition is stronger than the standard ABE definition because these components may be queried in an *adaptive* manner, allowing the adversary to build his key *piece by piece*. Note the actual notation we use in our scheme is slightly different than the way it is presented above. For the rest of this section we will assume key generation is allowed to be stateful (as captured with σ in the Revocable ABE definition) but will omit the state being updated as part of the syntax of the key generation algorithm from this point on for notational simplicity.

Definition 4. (Piecewise Key Generation) A KP-ABE scheme is said to have piecewise key generation with attribute set Ω supporting policies in \mathcal{P} , message space \mathbb{M} and identifier length \mathcal{I} if key generation and encryption are modified to the following syntax:

KeyGen takes as input the master key MSK , a bit b a policy $P \in \mathcal{P}$ and an identifier $U \in \{0, 1\}^{\mathcal{I}}$:

- $\mathbf{KeyGen}(MSK, b, P, U) \rightarrow K_{P,U}^{(b)}$.

Decrypt takes two outputs of KeyGen as the key input instead of one:

- $\mathbf{Decrypt}(C_S, K_0, K_1) \rightarrow M$.

We now define correctness of the scheme:

Definition 5. (Correctness) A KP-ABE scheme with piecewise key generation is correct if for any $S \subseteq \Omega$ and:

- $(PK, MSK) \leftarrow \mathbf{Setup}(1^\lambda)$,
- $C_S \leftarrow \mathbf{Encrypt}(PK, M, S)$,
- $P_0, P_1 \in \mathcal{P}$ such that $P_0(S) = P_1(S) = 1$:

If $\mathbf{KeyGen}(MSK, 0, P_0, U) \rightarrow K_{P_0,U}^{(0)}$, $\mathbf{KeyGen}(MSK, 1, P_1, U) \rightarrow K_{P_1,U}^{(1)}$, then,

$$\mathbf{Decrypt}(C_S, K_{P_0,U}^{(0)}, K_{P_1,U}^{(1)}) = M.$$

The definition for security for a scheme with *piecewise key generation* is now as one would expect: Unless the adversary has queried a single identifier U to

$$\mathbf{KeyGen}(MSK, 0, P_0, U) \text{ and } \mathbf{KeyGen}(MSK, 1, P_1, U)$$

such that $P_0(S) = P_1(S) = 1$, he should not be able to distinguish which message has been encrypted. We formalize this through the following game:

PIECEWISE KPABE SECURITY $_A(1^\lambda)$:

1. The challenger runs $\mathbf{Setup}(1^\lambda) \rightarrow (PK, MSK)$ and sends PK to A ;
2. A makes queries of the type (b, P, U) for $b \in \{0, 1\}$, $P \in \mathcal{P}$ and $U \in \{0, 1\}^{\mathcal{I}}$. The challenger runs $\mathbf{KeyGen}(MSK, b, P, U)$ and returns the key to A ;
3. A signals the query phase is over and returns (M_0, M_1, S^*) ;
4. Challenger picks b a random bit, sends $\mathbf{Encrypt}(PK, M_b, S^*)$ to A ;
5. A has another query phase as previously;
6. A sends a bit b' to the challenger;
7. If for any U , A has queried $\mathbf{KeyGen}(MSK, 0, P_0, U)$ and $\mathbf{KeyGen}(MSK, 1, P_1, U)$ such that $P_0(S^*) = P_1(S^*) = 1$ return 0;
8. If $b' = b$ return 1, otherwise return 0.

Definition 6. A KP-ABE scheme with piecewise key generation is secure if for any polynomial time adversary A the advantage of this adversary in the PIECEWISE KPABE SECURITY game:

$$2 \Pr[\text{PIECEWISE KPABE SECURITY}_A(1^\lambda) = 1] - 1$$

is negligible in λ .

3.1 Revocability from Piecewise KP-ABE

To apply our results to revocability we will need to make a requirement on exactly what types of policies we are assuming our piecewise KP-ABE scheme can support. Since our ultimate goal is to apply our result to the construction of [9], we will state our result if the KP-ABE scheme supports access policy formatted as LSSS matrices (as in [9]).

LSSS Matrices. We begin with a brief overview of LSSS matrices. A LSSS (linear secret sharing scheme) policy is of the type (A, ρ) where A is an $n \times l$ matrix over the base field \mathbb{F} (whose dimensions may be chosen at the time of encryption) and ρ is a map from $[n]$ to Ω , the universe of attributes in the scheme. A policy (A, ρ) satisfies an attribute set $S \subseteq \Omega$ if and only if $1 = (1, 0, 0, \dots, 0) \in \mathbb{F}^l$ is contained in $\text{Span}_{\mathbb{F}}(A_i : \rho(i) \in S)$ where A_i is the i^{th} row of A . An LSSS policy (A, ρ) is called *injective* if ρ is injective. We now state our result on a black-box reduction between KP-ABE schemes with piecewise key generation supporting LSSS policies to Revocable KP-ABE supporting LSSS policies.

Theorem 1. Let \mathcal{E} be a KP-ABE scheme with piecewise key generation supporting injective LSSS matrices with attribute set of size ω . Then there exists a Revocable KP-ABE scheme \mathcal{F} supporting injective LSSS matrices with time bound T with an attribute set of size $\omega - 2 \log(T)$.

In Section 7 we prove a stronger statement: In language that will be introduced shortly, we will prove that a KP-ABE scheme with piecewise key generation and ciphertext delegation implies a Revocable-Storage KP-ABE scheme. The construction presented in Section 5 when the underlying scheme does not have ciphertext delegation fulfills the requirement of being a Revocable KP-ABE scheme and therefore we defer the construction to that section. In Section 8 we give a modification of the fully secure ABE scheme due to Lewko et al. [9] that has piecewise key generation.

4 Revocable-Storage Attribute Based Encryption

Motivated by settings in which a third party is managing access on encrypted data, we now present a new direction for revocability. We call the property we achieve *revocable storage* - the ability for a ciphertext to be updated using only the public key so that revoked users can no longer decrypt the refreshed ciphertext. This is an additional functionality that is added onto standard revocability,

allowing an untrusted third party storing ciphertexts to update them in a way to make revoked users unable to decrypt messages they were once authorized to access. This can be thought of as an adaptation of forward security [4] which is used to manage keys, to managing ciphertexts.

Definition 7. (Revocable Storage) *A Revocable KP-ABE scheme has Revocable Storage if it has the following additional algorithm:*

- **CTUpdate**($C_{S,t}, PK \rightarrow C_{S,t+1}$): Ciphertext updating transforms a ciphertext encrypted at time t to an independent encryption of the same message under the same set S at time $t + 1$.

Formally, for any attribute set $S \subset \Omega$, time $t \in [T - 1]$ and message $M \in \mathbb{M}$, if PK and C are such that $\text{KeyGen}(1^\lambda) \rightarrow (MSK, PK)$ and $\text{Encrypt}(PK, M, S, t) \rightarrow C$ then: $\text{Encrypt}(PK, M, S, t + 1) \equiv \text{CTUpdate}(C, PK)$, where \equiv denotes equality in distribution.

Note that this is a very strong distributional requirement as we are insisting that starting from *any* ciphertext allows complete resampling from the output of the encryption algorithm. This strong requirement is motivated further in the following section.

We will call a Revocable KP-ABE scheme with Revocable Storage, a Revocable Storage KP-ABE scheme for brevity. Notice that the above procedure allows us to accomplish our motivating applications; it allows a third party storing ciphertexts to update the ciphertexts after revocation has been done at time t so that only the non-revoked users can continue decrypting. We will impose the restriction that all parameters should only depend polylogarithmically on T , the upper bound for the time component and n , the total number of users in the scheme. It is worth observing that there are trivial inefficient ways to satisfy this requirement assuming a standard KP-ABE scheme (i.e. by having $C_{S,t} = \{\text{Encrypt}(PK, M, S, t') : t' \geq t\}$ and having the update procedure simply delete the lowest indexed ciphertext) that depend polynomially on T .

5 Ciphertext Delegation

Revocable storage centers around allowing an untrusted third party to manage access on ciphertexts by incrementing the time component. To accomplish this, we need a process by which a ciphertext can be made harder to decrypt using only public operations in a more efficient way than simply re-encrypting under the more restrictive policy.

We call this problem *ciphertext delegation* - where a user who has access to only the ciphertext and public key may process this information into a completely new encryption under a more restrictive access policy. We consider this problem for attribute based encryption and show a simple method to classify delegation made possible in existing ABE schemes. We say that a ciphertext with a given access policy can be *delegated* to a more restrictive policy if there is a procedure that given any valid encryption of a message under the first policy produces a

independent and uniformly chosen encryption of the same message under the new access policy. Note delegation is required to produce a new encryption of the same message that is independent of the randomness and access policy of the original ciphertext being delegated from. This requirement is crucial if multiple delegations from the same base ciphertext are ever used in a scheme. Without this guarantee, multiple delegations may have correlated randomness and the security of the underlying scheme would not imply any security in these applications.

5.1 KP-ABE Ciphertext Delegation

For monotone access policies (as are generally considered in the literature [8,9,18]), the natural way to restrict access is by removing attributes from the ciphertext's attribute set. Note that for non-monotone access policies, delegation is not achievable without severely limiting the key policies permitted as any change in attribute set will make the ciphertext decryptable to certain simple policies not previously authorized, implying that delegation would violate security if these policies are supported.

Definition 8. (KP Delegation) *A KP-ABE scheme \mathcal{E} with message space \mathbb{M} and attribute space Ω is said to have ciphertext delegation if there is an algorithm Delegate with the following guarantee: For any $S' \subseteq S \subseteq \Omega$ and any $(PK, MSK) \leftarrow \mathcal{E}.\text{Setup}(1^\lambda)$, $M \in \mathbb{M}$ and $C_S \leftarrow \mathcal{E}.\text{Encrypt}(PK, M, S)$:*

$$\mathcal{E}.\text{Delegate}(PK, C_S, S') \equiv \mathcal{E}.\text{Encrypt}(PK, M, S').$$

We show briefly how delegation is possible in the KP-ABE scheme due to Goyal et al. [8] as the delegation procedures in the other listed schemes follow similarly. In this scheme, a ciphertext C_S with attribute set S is encrypted as:

$$C_S = (S, MY^s, \{T_i^s : i \in S\})$$

where s is chosen uniformly in \mathbb{Z}_p and $\{T_i : i \in S\}$, Y are part of the public key. To delegate this to an encryption under attribute set $S' \subseteq S$, we first modify the ciphertext to be $(S', MY^s, \{T_i^s : i \in S'\})$ by replacing S with S' and deleting elements in the third component. While this is a valid ciphertext under attribute set S' notice that it is not a valid delegation since we require the delegated ciphertext to be a completely independent encryption. By generating $Y^{s'}$, $\{T_i^{s'} : i \in S'\}$ with s' uniformly chosen from \mathbb{Z}_p , the ciphertext can be modified to $(S', MY^{s+s'}, \{T_i^{s+s'} : i \in S'\})$ which is a fresh uniform encryption of M with attribute set S' . A similar simple analysis also holds for [9,18] which allows us to conclude:

Theorem 2. *The KP-ABE schemes in [8,9,18] have ciphertext delegation.*

In the full version of this paper we also give full delegation procedures for a variety of existing CP-ABE schemes whose access structures are either built on threshold trees [18,2] or LSSS matrices [19,9]. For this analysis we define

certain operations that we call *elementary ciphertext manipulations* that can be performed on ciphertexts which all the schemes we mentioned above allow and show how any CP-ABE scheme that allows elementary ciphertext manipulations actually allows robust delegation operations on the underlying ciphertext. We cite one of our results which follows as a special case of our more general analysis. A formal definition of threshold trees is present either in the full version of this paper or in either of the cited works³.

Theorem 3. *The CP-ABE schemes [18,2]) allow the following delegation: A ciphertext $C_{\mathcal{T}}$ encrypted under threshold tree \mathcal{T} can be delegated to a ciphertext $C_{\mathcal{T}'}$ if \mathcal{T}' can be derived from \mathcal{T} by any sequence of the following operations:*

1. Inserting a node x along an edge with threshold $n_x = 1$ (In other words, splitting an edge into two, connected by a node x).
2. Increasing a threshold $n_x \rightarrow n_x + 1$ while optionally adding another descendant leaf y to x assigned to an arbitrary attribute.
3. Deleting a subtree.

6 Managing the Time Structure Efficiently

In this section we will give the main technical lemma needed to achieve efficient delegation. We use a binary tree \mathcal{Q} of depth $\log(T) = r$ (from now on we will assume T is a perfect power of 2 for notational convenience, if not then the r will just be taken to $\log(T)$ rounded up to the next integer), in a method similar to that used by Canetti et al. [4] in the context of forward secure encryption. Nodes of this tree will correspond to different attribute sets, while a single encryption of the delegatable scheme, interestingly, will be comprised of multiple encryptions from the underlying KP-ABE scheme, each one corresponding to an attribute set from a different node of the tree. While only one of these ciphertext components may be necessary for a secret key holder to decrypt, the ciphertexts include multiple components for delegation purposes.

Labeling Nodes of the Tree. Associate with each leaf of \mathcal{Q} a string corresponding to its path from the root with 0's denoting that the path traverses the left child of the previous node and 1's indicating traversing through the right child. As an example, the string 0^r corresponds to the leftmost leaf of the tree while $0^{r-1} \circ 1$ corresponds to its right sibling. Associate non-leaf nodes to strings by the path from the root by using * values to pad the string to r bits. For example, the root node will be referred to by the string $*^r$ while $0 \circ *^{r-1}$ refers

³ Informally, a threshold tree is an access structure represented by a tree where leaves are labeled with attributes and each internal node is labeled with an integer threshold. To see if the tree evaluates to true on a set of attributes, first the leaves corresponding to these attributes are labeled true and all others are labeled false. Then an internal node on the second level is labeled true only if a number of its descendants equal to or exceeding its threshold are labeled true. After this process is recursed through all of the tree, if the root evaluates to true, the threshold tree is satisfied.

to its left child. The string associated with a node v will be labeled $b(v)$. We refer to the node associated with a string x as v_x ; notice this gives a bijection between the time components $t \in \{0, 1\}^r$, and the leaves of the tree \mathcal{Q} .

Managing the access structure will require associating each time $t \in [T]$ with a set of nodes in the tree through a process we describe below. The following theorem will be the main method through which we will handle the time component of our final revocable storage construction. This theorem is also implicit in the work of [4] but we provide a proof in the full version for containment. For the theorem statement we will replace the time bound T with q to avoid confusing it with the trees, that will be called \mathcal{T} . The value r is now $\log(q)$. Below the term ‘efficiently computable’ means in time linear in r .

Theorem 4. *There are (efficiently computable) subsets of $V(\mathcal{Q})$ (the node set of \mathcal{Q} where \mathcal{Q} is a tree of depth r), $\{\mathcal{T}_i : i \in [q]\}$ such that for all $t \in \{0, 1\}^r$:*

- Property 1. \mathcal{T}_t contains an ancestor of $v_{t'}$ if and only if $t \leq t'$;
- Property 2. If $u \in \mathcal{T}_{t+1}$ then there is an ancestor of u in \mathcal{T}_t ;
- Property 3. $|\mathcal{T}_t| \leq r$.

We first give an informal intuition of how this sequence of trees will be used in the scheme. A secret key for time t will be associated with the leaf v_t of \mathcal{Q} while a ciphertext at time t' will be associated with the set of nodes $\mathcal{T}_{t'}$. A key for time t will succeed in decryption (by using the underlying KP-ABE scheme) if and only if v_t is a descendant of some node of the ciphertext (Property 1. above will then imply that a key at time t will only succeed in decrypting ciphertexts from earlier times).

Additionally, in our implementation delegation will be possible by traversing down the tree - a ciphertext associated with a set of nodes will be delegatable to a ciphertext associated with another set if and only if for every node in the target set (for the ciphertext being delegated to), one of its ancestors is in the first set (the set associated with the ciphertext being delegated from). Property 2. allows us to update ciphertexts. Property 3. guarantees that this process can be done efficiently.

7 Revocable Storage KP-ABE

By combining the method above for achieving linear delegation with our fully secure KP-ABE scheme with piecewise key generation and ciphertext delegation (given in Section 8), we will now show the following theorem. We defer the construction of our KP-ABE scheme with the required guarantees until after this section as the specific construction and security proof is involved and unnecessary for understanding the connection between piecewise key generation, delegation and revocable storage.

Theorem 5. *Let \mathcal{E} be KP-ABE scheme with ciphertext delegation and piecewise key generation that supports injective LSSS matrices with attribute set size ω . Then there exists a Revocable Storage KP-ABE scheme \mathcal{F} that supports injective LSSS matrices with time bound T with attribute set size $\omega - 2 \log(T)$.*

To prove the above theorem, we will use a second tree \mathcal{U} for revocation management (as in [3]) with the identities $\{0, 1\}^{\mathcal{T}}$ labeling the leaves. For a set of leaves V , the function $\mathcal{U}(V)$ returns a (deterministically chosen) collection of nodes of \mathcal{U} such that some ancestor of a leaf v is included in $\mathcal{U}(V)$ if and only if $v \notin V$. That such a function exists and can be computed in polynomial time in the size of V and \mathcal{T} is shown in [3]. Define $\text{Path}(ID)$ for a string ID (where the name of the node identifies the path from the root to this node, as in Section 6) as the set of nodes from the root of \mathcal{U} to the leaf v_{ID} (including the root and leaf).

We separate the attribute set of \mathcal{E} , reserving some attributes to only be used to manage the time component. Write the attribute set of \mathcal{E} as $\Omega' \cup \Omega$ where $|\Omega'| = 2 \log(T)$ and label the components of Ω' as:

$$\Omega' = \{\omega_{i,b} : i \in [\log(T)], b \in \{0, 1\}\}$$

and for each node y define (where the string $b(y)$ is comprised of 0, 1 and *'s defined in Section 6 corresponding to the path from the root to this node with *'s padding the string to length $\log(T)$) the set s_y as follows. For all $i \in [\log(T)]$:

- If $b(y)[i] = 0$, $\omega_{i,0} \in s_y$ and if $b(y)[i] = 1$, $\omega_{i,1} \in s_y$,
- If $b(y)[i] = *$, then $\omega_{i,0}$ and $\omega_{i,1}$ are both included in s_y .

we will shortly explain the significance of defining this set. Next let P_t be the policy defined by:

$$P_t(S) = 1 \text{ if and only if } \omega_{i,t[i]} \in S \quad \forall i \in [\log(T)]$$

where $t[i]$ is the i^{th} bit of t . The important observation about P_t and s_y is that $P_t(s_y) = 1$ if and only if y is an ancestor of the leaf corresponding to t and that injective LSSS matrices suffice to express the policies P_t . The encryption and key generation procedures of \mathcal{F} operate over Ω (which removes $2 \log(T)$ attributes from the scheme). We now describe how to construct our Revocable Storage KP-ABE scheme \mathcal{F} from \mathcal{E} defined above. We use \mathcal{T}_t as defined in Theorem 4.

- **Setup**(1^λ): Return \mathcal{E} . $\text{Setup}(1^\lambda) = (PK, MSK)$
- **KeyGen**(MSK, P, ID): For all $x \in \text{Path}(ID)$ set

$$SK_{P,x}^{(0)} = \mathcal{E}.\text{KeyGen}(MSK, 0, P, x).$$

Return:

$$SK_{P, ID}^{(0)} = \{SK_{P,x}^{(0)} : x \in \text{Path}(ID)\}.$$

- **Encrypt**(PK, M, S, t) where $S \subseteq \Omega$: For all $x \in \mathcal{T}_t$ set:

$$C_{S,x} = \mathcal{E}.\text{Encrypt}(PK, M, S \cup s_x).$$

Return:

$$C_{S,t} = \{C_{S,x} : x \in \mathcal{T}_t\}.$$

- **KeyUpdate**(MSK, rl, t): For all $x \in \mathcal{U}(rl)$ set:

$$SK_{P_t, x}^{(1)} = \mathcal{E}.\text{KeyGen}(MSK, 1, P_t, x).$$

Return:

$$K_t = \{SK_{P_t, x}^{(1)} : x \in \mathcal{U}(rl)\}.$$

- **Decrypt**($SK_{P, ID}, K_{t'}, C_{S, t}$): If $ID \notin rl$ when $K_{t'}$ was generated, there is some $x \in \mathcal{U}(rl) \cap \text{Path}(ID)$ (by the definition of $\mathcal{U}(V)$). For this x there is:

$$SK_{P, x}^{(0)} \in SK_{P, ID} \text{ and } SK_{P_{t'}, x}^{(1)} \in K_{t'}.$$

Additionally, if $t' \geq t$ there is some $y \in \mathcal{T}_t$ such that y is an ancestor of the leaf $v_{t'}$, which implies $P_{t'}(s_y) = 1$. For this y , take $C_{S, y} \in C_{S, t}$ and return:

$$\mathcal{E}.\text{Decrypt}(SK_{P, x}^{(0)}, SK_{P_{t'}, x}^{(1)}, C_{S, y}).$$

If $P(S) = 1$ then $P(S \cup s_y) = P_{t'}(S \cup s_y) = 1$ implying decryption succeeds.

- **CTUpdate**($PK, C_{S, t}$): For all $x \in \mathcal{T}_{t+1}$ find $y \in \mathcal{T}_t$ such that y is an ancestor of x . Then there is a $C_{S, y}$ component in $C_{S, t}$. For all such x set:

$$C_{S, x} = \mathcal{E}.\text{Delegate}(PK, C_{S, y}, S \cup s_x)$$

Which is possible since y being an ancestor of x implies $s_x \subset s_y$. Return:

$$C_{S, t+1} = \{C_{S, x} : x \in \mathcal{T}_{t+1}\}$$

We now describe how security of the underlying \mathcal{E} implies security of \mathcal{F} in the Revocable KP-ABE security game. That **CTUpdate** is correct can be observed simply and it therefore only remains to argue that the above is a secure Revocable KP-ABE scheme.

Proof of RKP-ABE Security. Let A be such that RKP-SECURITY_A is non-negligible, we will construct an A' such that $\text{PIECEWISE KP-ABE}_{A'}$ is non-negligible. We will modify the **PIECEWISE** security game slightly and give an A' with non-negligible advantage when instead of a single challenge query, the adversary gives a pair of messages (M_0, M_1) as well as a tuple of sets $(S_1^*, S_2^*, \dots, S_\rho^*)$ and is returned the tuple: $\{\text{Encrypt}(PK, M_b, S_i^*) : i \in [\rho]\}$ by the challenger with the restriction that all S_i^* obey the restriction placed on S^* in the standard game. This implies security in the standard **PIECEWISE KP-ABE** security game by a standard hybrid argument.

A' begins by initializing the **PIECEWISE KP-ABE** security game and forwarding PK to A . To respond to an $SK(P, ID)$ query A' sends a query $(0, P, x)$ to its key generation oracle for all $x \in \text{Path}(ID)$ which drawn from the same distribution as the construction above. Similarly, for all queries $K(t, rl)$, A' sends a query $(1, P_t, x)$ for all $x \in \mathcal{U}(rl)$ to its key generation oracle to simulate the key update information.

When A makes a challenge query (M_0, M_1, S^*, t^*) in order to simulate this, A' in the modified game we described above will send as its challenge query

(M_0, M_1) and the tuple $S^* \cup s_x$ for all $x \in \mathcal{T}_{t^*}$. Notice that by responding to the queries in this fashion we have perfectly simulated the expected distribution for A . It remains only to show that as long as A does not submit an invalid query that causes the experiment to automatically output 0 our A' has not submitted an invalid query to the PIECEWISE KP-ABE oracle in the modified game.

Take any $S^* \cup s_x$ in the challenge query that A' makes as described above. Take any $y \in \mathcal{U}$ we now claim that either for either $b = 0$ or $b = 1$ all queries of the type (b, P, y) that A' makes while simulating the queries of A , $P(S^* \cup s_x) = 0$.

First consider the case where for some descendent leaf ID of y (which is a \mathcal{I} bit string) that A makes a query to $SK(P, ID)$ with $P(S^*) = 1$. In this case by the guarantee on the queries of A for all $K(t, rl)$ queries with $t \geq t^*$, $ID \in rl$. This implies that for all queries of the type $(1, P_t, z)$ that A' makes, either $t < t^*$ (in which case $P_t(S^* \cup s_x) = 0$ since P_t does not depend on S^* and $P_t(s_x) = 0$ since x is not an ancestor of t because $x \in \mathcal{T}_{t^*}$) or $ID \in rl$ which implies no ancestor of ID is contained in $\mathcal{U}(rl)$ and therefore, z is not an ancestor of ID and therefore $z \neq y$. So we have established that in this case, all $(1, P, y)$ queries have the property that $P(S^* \cup s_x) = 0$ as desired.

Next, consider the case where A does not make a query to a descendent leaf ID of y with $SK(P, ID) = 1$. Then, in simulating A' only makes $(0, P, y)$ queries where $P(S^*) = 0$ which implies $P(S^* \cup s_x) = 0$ (since the policies for the Revocable scheme are only over Ω). This shows the desired statement in both cases, proving the theorem. Using the construction given in Section 8 we conclude:

Theorem 6. *Under Assumptions 1, 2, and 3. (defined below), when \mathcal{E} is set to be the scheme given in Section 8 the above \mathcal{F} is a secure Revocable Storage KP-ABE scheme supporting injective LSSS Matrices.*

8 Piecewise KP-ABE Construction

We now introduce the assumptions we will be using to build our scheme. These are the same assumptions from the fully secure ABE construction due to Lewko et al. [9]:

Assumption 1. Let \mathbb{G} be a cyclic group of size $N = p_1 p_2 p_3$ with bilinear map e selected according to the group generator $\mathcal{G}(1^\lambda)$. Consider the event that we generate $g \leftarrow \mathbb{G}_{p_1}, X_3 \leftarrow \mathbb{G}_{p_3}, T_1 \leftarrow \mathbb{G}_{p_1, p_2}, T_2 \leftarrow \mathbb{G}_{p_1}$ uniformly at random. Assumption 1. states that for any probabilistic polynomial time A :

$$|\Pr[A(\mathbb{G}, g, X_3, T_1) = 0] - \Pr[A(\mathbb{G}, g, X_3, T_2) = 0]|$$

is negligible in λ .

Assumption 2. Let \mathbb{G} be a cyclic group of size $N = p_1 p_2 p_3$ with bilinear map e selected according to the group generator $\mathcal{G}(1^\lambda)$. Consider the event that we generate $g, X_1 \leftarrow \mathbb{G}_{p_1}, X_2, Y_2 \leftarrow \mathbb{G}_{p_2}, X_3, Y_3 \leftarrow \mathbb{G}_{p_3}, T_1 \leftarrow \mathbb{G}$ and $T_2 \leftarrow \mathbb{G}_{p_1 p_3}$ uniformly at random. Assumption 2. states that for any probabilistic polynomial time A (if we let $D = (\mathbb{G}, g, X_1 X_2, X_3, Y_2 Y_3)$):

$$|\Pr[A(D, T_1) = 0] - \Pr[A(D, T_2) = 0]|$$

is negligible in λ .

Assumption 3. Let \mathbb{G} be a cyclic group of size $N = p_1 p_2 p_3$ with bilinear map e selected according to the group generator $\mathcal{G}(1^\lambda)$ with target group \mathbb{G}_T . Consider the event that we generate $g \leftarrow \mathbb{G}$, $\alpha, s \leftarrow \mathbb{Z}_N$, $X_2, Y_2, Z_2 \leftarrow \mathbb{G}_{p_2}$, $X_3 \leftarrow \mathbb{G}_{p_3}$ uniformly at random. Finally select $T_1 = e(g, g)^{\alpha s}$ and $T_2 \leftarrow \mathbb{G}_T$. Assumption 3. states that for any probabilistic polynomial time A , if we let D denote $D = (\mathbb{G}, \mathbb{G}_T, N, g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$, then:

$$|\Pr[A(D, T_1) = 0] - \Pr[A(D, T_2) = 0]|$$

is negligible in λ .

We now provide the construction that fulfils the requirements put forth in the body of the paper. We prove this theorem in the full version of this paper.

Theorem 7. *The KP-ABE scheme given below has piecewise key generation, supports LSSS policies and has ciphertext delegation if Assumptions 1, 2, 3. hold.*

Since \mathbb{G} is cyclic, it has unique subgroups of size p_1 , p_2 and p_3 which we call \mathbb{G}_{p_1} , \mathbb{G}_{p_2} and \mathbb{G}_{p_3} respectively. We let the vector 1 stand as shorthand for $1 \circ 0 \circ 0 \dots 0$ when the dimension is specified by context. Below we will give the decryption algorithm the public key as input, which was not part of our original definition, but can be easily adapted to our definition by including the public key as part of the secret key.

Setup(1^λ) \rightarrow (PK, MSK): Choose a bilinear group of order $N = p_1 p_2 p_3$ (three distinct primes) according to $\mathcal{G}(1^\lambda)$. Then choose $\alpha \leftarrow \mathbb{Z}_N$ and $g \leftarrow \mathbb{G}_{p_1}$ uniformly. For each $i \in \Omega$, $s_i \leftarrow \mathbb{Z}_N$ uniformly at random. Pick $X_3 \in \mathbb{G}_{p_3}$ uniformly with $X_3 \neq 1$ and set:

$$PK = (N, g, e(g, g)^\alpha, X_3, T_i = g^{s_i} \text{ for all } i \in \Omega), \quad MSK = (\alpha, PK).$$

KeyGen($MSK, b, (A, \rho), U, PK$). If α_U has not been generated yet, generate it and store it. For each row A_i of A choose a uniform $r_i \leftarrow \mathbb{Z}_N$ and $W_i, V_i \in \mathbb{G}_{p_3}$. If $b = 0$ let u be a random l dimensional vector over \mathbb{Z}_N such that $1 \cdot u = \alpha_U$ otherwise, sample it subject to the restriction that $1 \cdot u = \alpha - \alpha_U$. For all $i \in [n]$ set:

$$K_i^{(1)} = g^{A_i \cdot u} T_{\rho(i)}^{r_i} W_i, \quad K_i^{(2)} = g^{r_i} V_i$$

and return $SK_{U, (A, \rho)}^{(b)} = \{K_i^{(1)}, K_i^{(2)} : i \in [n]\}$.

Encrypt(PK, M, S). Choose $s \leftarrow \mathbb{Z}_N$ at random. Return:

$$CT_S = (C = Me(g, g)^{\alpha s}, C_0 = g^s, (C_i = T_i^s : i \in S)).$$

Decrypt($CT_S, SK_{U, (A, \rho)}^{(0)}, SK_{U, (B, \beta)}^{(1)}, PK$). Take ω_i with $\sum_i \omega_i A_i = 1$.

Label the components of $SK_{U, (A, \rho)}^{(0)}$ as $K_i^{(1)}, K_i^{(2)}$ for $i \in [n]$ and set:

$$\prod_{\rho(i) \in S} \frac{e(C_0, K_i^{(1)})^{\omega_i}}{e(C_{\rho(i)}, K_i^{(2)})^{\omega_i}} = \prod_{\rho(i) \in S} \frac{e(g, g)^{s\omega_i A_i \cdot u} e(g, T_{\rho(i)}^{s r_i \omega_i})}{e(g, T_{\rho(i)})^{s r_i \omega_i}} = e(g, g)^{s\alpha_U}$$

And using the identical procedure with $SK_{U,(B,\beta)}^{(1)}$ recovers $e(g, g)^{s(\alpha - \alpha_U)}$ allowing recovery of $e(g, g)^{s\alpha}$ and M as $C/e(g, g)^{s\alpha}$.

Acknowledgements. We gratefully thank Thomas King and Daniel Manchala of Xerox for stimulating discussions regarding ABE with dynamic credentials. In particular, Thomas King and Daniel Manchala suggested to us that the problem of revoking access to stored data that had not yet been accessed could be of significant interest in practice, and their suggestion inspired us to consider this problem. We also thank the anonymous reviewers for their helpful comments on our writeup.

References

1. Aiello, W., Lodha, S.P., Ostrovsky, R.: Fast Digital Identity Revocation (Extended Abstract). In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 137–152. Springer, Heidelberg (1998)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
3. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: ACM CCS, pp. 417–426 (2008)
4. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
5. Gentry, C.: Certificate-Based Encryption and the Certificate Revocation Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
6. Goyal, V.: Certificate Revocation Using Fine Grained Certificate Space Partitioning. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 247–259. Springer, Heidelberg (2007)
7. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-box accountable authority identity-based encryption. In: CCS, pp. 427–436 (2008)
8. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS, pp. 89–98 (2006)
9. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
10. Libert, B., Vergnaud, D.: Adaptive-ID Secure Revocable Identity-Based Encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 1–15. Springer, Heidelberg (2009)
11. Micali, S.: Efficient certificate revocation. LCS/TM 542b, Massachusetts Institute of Technology (1996)
12. Micali, S.: NOVOMODO: Scalable certificate validation and simplified PKI management. In: Proc. of 1st Annual PKI Research Workshop (2002)

13. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. *ECCC* (043) (2002)
14. Naor, M., Nissim, K.: Certificate revocation and certificate update. *IEEE Journal on Selected Areas in Communications* 18(4), 561–570 (2000)
15. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
16. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: *CCS*, p. 203 (2007)
17. Qian, J., Dong, X.: Fully secure revocable attribute-based encryption. *Journal of Shanghai Jiaotong University (Science)* 16(4), 490–496 (2011)
18. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
19. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) *PKC 2011*. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)

Functional Encryption for Regular Languages

Brent Waters*

The University of Texas at Austin
`bwaters@cs.utexas.edu`

Abstract. We provide a functional encryption system that supports functionality for regular languages. In our system a secret key is associated with a Deterministic Finite Automata (DFA) M . A ciphertext CT encrypts a message m and is associated with an *arbitrary length* string w . A user is able to decrypt the ciphertext CT if and only if the DFA M associated with his private key accepts the string w .

Compared with other known functional encryption systems, this is the first system where the functionality is capable of recognizing an unbounded language. For example, in (Key-Policy) Attribute-Based Encryption (ABE) a private key SK is associated with a single boolean formula ϕ which operates over a *fixed* number of boolean variables from the ciphertext. In contrast, in our system a DFA M will meaningfully operate over an arbitrary length input w .

We propose a system that utilizes bilinear groups. Our solution is a “public index” system, where the message m is hidden, but the string w is not. We prove security in the selective model under a variant of the decision ℓ -Bilinear Diffie-Hellman Exponent (BDHE) assumption that we call the decision ℓ -Expanded BDHE problem.

1 Introduction

Functional encryption is an emerging new way of viewing encryption. Instead of encrypting data to a particular user, an encryptor will encrypt data under the public parameters of a system. Users in the system will obtain a secret key SK (issued from some authority) that is associated with a value k . What a user learns about the data depends on the systems functionality and their key value k .

Perhaps the most well known and representative functional encryption system is Attribute-Based Encryption (ABE) [25,18]. In a (Key-Policy) ABE system, a key is associated with a *single* boolean formula ϕ and a ciphertext is associated with a pair of a message m and variables \boldsymbol{x} . A user can decrypt and learn m if and only if \boldsymbol{x} satisfies the formula ϕ . One salient feature of this system is that the formula ϕ only operates over a *fixed* number of variables (i.e., a bounded description from the ciphertext). However, there are many applications where we might want the functionality key to operate over arbitrary sized input data.

* Supported by NSF CNS-0952692, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

For example, we could imagine a network logging application where an encryption input represents an arbitrary number of events logged. Another example is an encryption of a database of patient data that includes disease history paired with gene sequences where the number of participants is not a priori bounded or known. This restriction on fixed size inputs is not limited to Attribute-Based Encryption; other functional encryption systems such as Spatial Encryption [10,19] or Inner Product Encryption [21,23,24] also operate over fixed size inputs.¹

Functional Encryption over Arbitrary Size Inputs. We initiate the study of functional encryption systems that operate over arbitrary size inputs. We begin with systems that support a regular language functionality. Regular languages provide a natural starting point since they are well studied and are used in various practical applications such as specifying rules in firewall systems.

A general class of applications is where there is a family of documents that has a public (not hidden) template along with some secret data that fits within that template. For instance, one can consider tax returns — which can be arbitrarily long — where the template consists of the deductions claimed and the private data is the amount of these deductions. An auditor might be authorized to read the private data of a return if the deductions match a certain regular expression. Another example is a virus scanner for webpages. Here we might wish the scanner to be able to take a deeper look into pages that match a certain pattern, but have some content remain hidden otherwise.

In general there are multiple sources of data such as video and databases that can be of arbitrary length. While a long term goal is to realize functional encryption for arbitrary programs, regular languages capture interesting properties. For instance, one can check for the presence of substrings or check the parity of data.

Our Contribution. We construct what we call a Deterministic Finite Automata (DFA)-based Functional Encryption system. In our system the encryption algorithm takes in a pair of a message m and an arbitrary length string w over a finite alphabet Σ . The key generation algorithm takes as input the description of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and outputs a secret key. Briefly, Q is a set of states, Σ the alphabet, $\delta : Q \times \Sigma \rightarrow Q$ a next state transition function, $q_0 \in Q$ a unique start state and $F \subseteq Q$ a set of accept states.² A user with a secret key for DFA M will be able to decrypt a ciphertext CT associated with w if and only if M accepts the string w . Our system is of the “public index” type [11] (like ABE) in that while the message m is hidden w can be determined from the ciphertext (without a secret key).

Designing an encryption system to support a DFA-type functionality provides a new set of challenges. It is useful to compare them to those in building a boolean formula (ABE) type system. In existing ABE systems a ciphertext

¹ While a regular language can be recognized by a family of boolean formulas, a secret key in an ABE system is associated with a single formula.

² We give an overview of DFAs in Section 2.

had to represent the variables in \mathbf{x} . This generally consisted of either including or excluding a particular ciphertext component depending on whether a corresponding variable was true. In our DFA system we must meaningfully represent an arbitrary length string w . Here representing the order of the symbols comprising w is paramount. In an ABE key generation phase a boolean formula can be mapped closely with the linear secret sharing of the master secret key in the exponent. Similarly, decryption maps closely with the linear reconstruction of these exponents (post pairing with pieces of the ciphertext). Building a DFA description M into a secret key and enforcing its execution during decryption is a more complex task.

We now give a high level overview of some of the main ideas of our construction. We defer several details to Section 3 where the reader will have the benefit of our construction description. Our construction makes use of (prime order) bilinear groups and is focused on three main mechanisms for decryption. We call these the initiation, transition, and completion mechanisms. At a high level the only useful actions a decryptor or attacker should be able to take are by applying them. These mechanisms are realized by how we structure ciphertexts and private keys.

When encrypting a ciphertext for string w of ℓ symbols, the encryptor chooses $\ell + 1$ random exponents $s_0, s_1, \dots, s_\ell \in \mathbb{Z}_p$ where p is the order of the group. A private key machine $M = (Q, \Sigma, \delta, q_0, F)$ will have $|Q|$ random group elements chosen $D_0, \dots, D_{|Q|-1}$, where D_x is associated with state q_x .

Suppose a user is trying to decrypt a ciphertext associated with string w with a secret key SK for machine M . Throughout the process of decryption we would like to enforce that the user (or attacker) can only compute $e(g, D_x)^{s_j}$ if the machine M lands on state q_x after reading j symbols of w .

First, the initiation mechanism allows the decryptor to obtain the value $e(g, D_0)^{s_0} \in \mathbb{G}_T$. This initiation mechanism should only be good for computing this value and not useable for other D_i, s_i . Next, the decryptor can repeatedly apply the transition mechanism. The transition mechanism is structured such that one can compute $(e(g, D_x)^{s_j})^{-1} e(g, D_y)^{s_{j+1}}$ if and only if the $j + 1$ -th symbol of w is σ and the transition $\delta(x, \sigma) = y$. Multiplying this into the accumulated value will allow the decryption algorithm to change the accumulated value $e(g, D_x)^{s_j}$ to $e(g, D_y)^{s_{j+1}}$. Thus, mirroring the computation of the DFA M on w . Finally, we will end up with some accumulated value $e(g, D_x)^{s_\ell}$. If the state q_x is an accept state in F , then one can apply the completion mechanism which computes $e(g, g)^{\alpha s_\ell} (e(g, D_x)^{s_\ell})^{-1}$ and this can be multiplied in to simply obtain $e(g, g)^{\alpha s_\ell}$. This is the term used to blind the message.

We prove security in the selective model under a variant of the decision ℓ -Bilinear Diffie-Hellman Exponent (BDHE) assumption that we call the decision ℓ -Expanded BDHE assumption. The parameter ℓ of the assumption depends upon the length of the string w^* associated with the challenge ciphertext. Our proof uses what has been called a partitioning strategy where a string w^* is embedded into the public parameters in such a way that a reduction algorithm can create private keys for any DFA M that does not accept w^* . Our proof

will also need to somehow reflect the computation of M on w^* when creating private keys for M . The main reason for applying a parameterized assumption is that our reduction needs to embed an arbitrary size w^* into fixed small sized parameters.

Efficiency. We give a brief overview of the computation and storage costs associated with our system. The public parameters contain $5 + |\Sigma|$ group elements, where $|\Sigma|$ is the size of the alphabet. A ciphertext consists of $5 + 2|w|$ group elements and an encryption costs $5 + 3|w|$ exponentiations, where $|w|$ is the string associated with the ciphertext. A private key consists of $4 + 3|\mathcal{T}|$ group elements, where $|\mathcal{T}|$ is the number of transitions in the DFA M associated with the key. Finally, a successful decryption requires $4 + 3|w|$ pairing operations.

Backtracking, NFAs and Future Challenges. One additional complexity to the prior discussion of mechanisms is that the transition function can be applied in reverse to “backtrack” through the computation. At first this does not seem to be any issue at all since (using our above example) it seems an attacker will only be able to go back from a value $e(g, D_y)^{s_{j+1}}$, representing that he was at state q_y after $j + 1$ symbols, to a prior value $e(g, D_x)^{s_j}$ which represents that earlier the machine was at state q_x after j symbols. However, a twist occurs if there exists another state x' and the transition function $\delta(x', \sigma) = y$, where the $j + 1$ -th symbol of w is σ .

In this case the attacker can actually backtrack along the “wrong” transition and compute $e(g, D_x)^{s_j}$; falsely representing that after evaluating the first j symbols of w M was in state $q_{x'}$! It turns out that in the case of our DFA system — reflected in our proof — this is not a problem as an attacker can only go backwards so far. When it goes forward again the determinism of M means that we will eventually return to the same state q_y after $j + 1$ symbols.

While this backtracking attack primarily complicates the proof of our DFA-based system, it is indicative that realizing efficient constructions of Nondeterministic Finite Automata (NFA) Functional Encryption systems will likely be difficult. In particular, the most natural extension of our construction to NFAs falls victim to a backtracking attack where an attacker used the inverse transitions and followed by a different forward transitions (as could be allowed by the nondeterminism). Creating efficient NFA constructions is an interesting problem since the best generic way of representing an NFA (or regular expressions) in terms of a DFA requires an exponential blowup in the number of states. However, we suspect that achieving this will be difficult and perhaps related to the difficulty of building efficient ABE systems for circuits.

We also comment that our current proof techniques use both the selective model and a parameterized assumption. An interesting question is if either or both of these properties can be improved using Dual System Encryption [29] proof techniques. While Dual System Encryption was used to prove ABE systems fully secure under static assumptions [22], the core technique encumbered a “one-use” restriction on attributes in formulas which had to be overcome using a

certain encoding technique. The encoding bounded at setup the number of times an attribute could be used in a formula. The natural analog of this encoding and its resulting setup bound would negate the motivation of having arbitrary string sizes as the ciphertext input.

A final open challenge is to design a functional encryption system where the DFA M associated with a key is applied directly on encrypted data. That is the string w itself would be hidden and a user would only learn whether M did or did not accept w . Creating such a cryptosystem and moving beyond the public index model in this setting appears challenging.

1.1 Other Related Work

Identity-Based Encryption (IBE), which is one of the most basic forms of functional encryption, was introduced by Shamir [26] in 1984. It wasn't until much later that Boneh and Franklin [9] and Cocks [14] independently proposed IBE systems. There have been multiple IBE constructions using bilinear maps [5,28,15] and more recently lattice-based constructions [16,13,2,3].

The beginning of functional encryption can be traced back to early Attribute-Based Encryption systems [25,18]. There exists a complimentary form of ABE known as Ciphertext-Policy ABE [4,17,30] where the secret key is associated with a set of variables and the ciphertext with a boolean formula ϕ . A natural analog in our DFA-based encryption is to associate a DFA M with the ciphertext and a string with a private key.

In the terminology of [11] our scheme works in the public index model since the string w is not hidden. Starting with Anonymous IBE systems [8,1] systems there have been multiple systems that attempt to hide this information. Some examples include Hidden Vector Encryption [12,20] and Inner Product functionalities [21,23,24] among others.

2 Functional Encryption for DFAs: Definitions and Assumption

We now give a formal definition of our DFA-Based Functional Encryption scheme. In the terminology of Boneh, Sahai, and Waters [11] we give a functional encryption scheme for functionality $F(k, x = (w, m))$ where k is the description of a deterministic finite automata M and x is the pair of a string w and a message m . The functionality F outputs the encrypted message m if the machine M accepts w and otherwise outputs \perp . The functional encryption scheme is of the “public index” type in that the string w is not hidden.

While our system fits within the framework of functional encryption [11] (as sketched above), we choose to present a direct definition for DFA-based functional encryption. We begin by giving a brief overview of DFAs. Then we present our algorithms and game-based security definitions. Finally, we give our bilinear group assumption used to prove security.

2.1 Overview of Deterministic Finite Automata

We give a brief overview of Deterministic Finite Automata (DFA) using terminology and definitions from Sipser [27] and refer the reader to Sipser [27] for further details. A Deterministic Finite Automata M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ in which:

1. Q is a set of states
2. Σ is a finite set of symbols called the alphabet
3. $\delta : Q \times \Sigma \rightarrow Q$ is a function known as a transition function
4. $q_0 \in Q$ is called the start state
5. $F \subseteq Q$ is a set of accept states.

For convenience when describing our encryption systems we add the notation of \mathcal{T} as being the set of transitions associated with the function δ , where $t = (x, y, \sigma) \in \mathcal{T}$ iff $\delta(x, \sigma) = y$.

Suppose that $M = (Q, \Sigma, \delta, q_0, F)$. We say that M accepts a string $w = w_1, w_2, \dots, w_\ell \in \Sigma^*$ if there exists a sequence of states $r_0, r_1, \dots, r_n \in Q$ where:

1. $r_0 = q_0$
2. For $i = 0$ to $n - 1$ we have $\delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$.

We will use the notation $\text{ACCEPT}(M, w)$ to denote that the machine M accepts w and $\text{REJECT}(M, w)$ to denote that M does not accept w . A DFA M recognizes a language L if M accepts all $w \in L$ and rejects all $w \notin L$; such a language is called regular.

2.2 DFA-Based Functional Encryption

We now give our definition of a DFA-based Functional Encryption system. A (Key-Policy) DFA-based encryption scheme consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt. In addition to the security parameter, the setup algorithm will take as input the description of an alphabet Σ . This alphabet will be used for all strings and machine descriptions in the system.³ The algorithm descriptions follow:

Setup($1^n, \Sigma$). The setup algorithm takes as input the security parameter and the description of a finite alphabet Σ . The alphabet used is shared across the entire system. It outputs the public parameters PP and a master key MSK.

Encrypt(PP, w, m). The encryption algorithm takes as input the public parameters PP, an arbitrary length string $w \in \Sigma^*$, and a message m . It outputs a ciphertext CT.

³ To fit our system properly in the framework of [11] we would need to have a separate functionality for every alphabet Σ . Then we would have a family of functional encryption systems; one for each alphabet. Here we choose to let Σ be a parameter to the setup algorithm.

Key Generation(MSK, $M = (Q, \mathcal{T}, q_0, F)$). The key generation algorithm takes as input the master key MSK and a DFA description M . The description passed does not include the alphabet Σ since it is already determined by the setup algorithm. In addition, we encode the transitions as a set \mathcal{T} of three tuples as described above. The algorithm outputs a private key SK.

Decrypt(SK, CT). The decryption algorithm takes as input a secret key SK and ciphertext CT. The algorithm attempts to decrypt and outputs a message m if successful; otherwise, it outputs a special symbol \perp .

Correctness. Consider all messages m , strings w , and DFA M such that ACCEPT(M, w). If $\text{Encrypt}(\text{PP}, w, m) \rightarrow \text{CT}$ and $\text{KeyGen}(\text{MSK}, M) \rightarrow \text{SK}$ where PP, MSK were generated from a call to the setup algorithm, then $\text{Decrypt}(\text{SK}, \text{CT}) = m$.

Security Model for DFA-Based Functional Encryption. We now describe a game-based security definition for DFA-Based Functional Encryption. As in other functional encryption systems (e.g. [9,25]), an attacker will be able to query for multiple keys, but not ones that can trivially be used to decrypt a ciphertext. In this case the attacker can repeatedly ask for private keys corresponding any DFA M of his choice, but must encrypt to some string w^* such that every machine M for which a private key was requested for rejects w^* . The security game follows.

Setup. The challenger first runs the setup algorithm and gives the public parameters, PP to the adversary and keeps MSK to itself.

Phase 1. The adversary makes any polynomial number of private keys queries for machine descriptions M of its choice. The challenger returns $\text{KeyGen}(\text{MSK}, M)$.

Challenge. The adversary submits two equal length messages m_0 and m_1 .

In addition, the adversary gives a challenge string w^* such that for all M requested in Phase 1, $\text{REJECT}(M, w^*)$. Then the challenger flips a random coin $b \in \{0, 1\}$, and computes $\text{Encrypt}(\text{PP}, w, m_b) \rightarrow \text{CT}^*$. The challenge ciphertext CT^* is given to the adversary.

Phase 2. Phase 1 is repeated with the restriction that for all M requested $\text{REJECT}(M, w^*)$.

Guess. The adversary outputs a guess b' of b .

The advantage of an adversary \mathcal{A} in this game is defined as $\Pr[b' = b] - \frac{1}{2}$. We note that the definition can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 1. A DFA-based Functional Encryption system is secure if all polynomial time adversaries have at most a negligible advantage in the above game.

We say that a system is *selectively* secure if we add an Init stage before setup where the adversary commits to the challenge string w^* and alphabet Σ .

2.3 Expanded ℓ -BDHE Assumption

We define the decision ℓ -Expanded Bilinear Diffie-Hellman Exponent problem as follows. Choose a group \mathbb{G} of prime order $p > 2^n$ for security parameter n . Next choose random $a, b, c_0, \dots, c_{\ell+1}, d \in \mathbb{Z}_p^*$ and a random $g \in \mathbb{G}$. Suppose an adversary is given $\mathbf{X} =$

$$\begin{aligned} & g, g^a, g^b, g^{ab/d}, g^{b/d} \\ & \forall_{i \in [0, 2\ell+1], i \neq \ell+1, j \in [0, \ell+1]} \quad g^{a^i s}, \quad g^{a^i b s / c_j} \\ & \forall_{i \in [0, \ell+1]} \quad g^{a^i b / c_i}, \quad g^{c_i}, \quad g^{a^i d}, \quad g^{abc_i / d}, \quad g^{bc_i / d} \\ & \forall_{i \in [0, 2\ell+1], j \in [0, \ell+1]} \quad g^{a^i b d / c_j} \\ & \forall_{i, j \in [0, \ell+1], i \neq j} \quad g^{a^i b c_j / c_i}. \end{aligned}$$

Then it must remain hard to distinguish $e(g, g)^{a^{\ell+1} b s} \in \mathbb{G}_T$ from a random element in \mathbb{G}_T .

An algorithm \mathcal{B} that outputs $z \in \{0, 1\}$ has advantage ϵ in solving decision ℓ -Expanded BDHE in \mathbb{G} if

$$\left| \Pr[\mathcal{B}(\mathbf{X}, T = e(g, g)^{a^{\ell+1} b s}) = 0] - \Pr[\mathcal{B}(\mathbf{X}, T = R) = 0] \right| \geq \epsilon$$

Definition 1. We say that the decision ℓ -Expanded BDHE assumption holds if no polytime algorithm has a non-negligible advantage in solving the decision ℓ -Expanded BDHE problem.

We give a proof that the assumption holds in the generic group model in the full version of our paper.

3 Construction

We now present our construction of a DFA-based Function Encryption system. We first describe our construction and then provide some additional intuitive discussion. Our formal proof appears in the next section.

3.1 Algorithms

Setup($1^n, \Sigma$) The setup algorithm takes as input a security parameter n and an alphabet Σ . It first chooses a prime $p > 2^n$ and creates a bilinear group \mathbb{G} of prime order p . The algorithm then chooses random group elements $g, z, h_{\text{start}}, h_{\text{end}} \in \mathbb{G}$. In addition, for all $\sigma \in \Sigma$ it chooses random $h_\sigma \in \mathbb{G}$. Finally, an exponent

$\alpha \in \mathbb{Z}_p$ is randomly chosen. The master secret key MSK includes $g^{-\alpha}$ along with the public parameters. The public parameters are the description of the group \mathbb{G} and the alphabet Σ along with:

$$e(g, g)^\alpha, g, z, h_{\text{start}}, h_{\text{end}}, \forall_{\sigma \in \Sigma} h_\sigma.$$

Encrypt(PP, $w = (w_1, \dots, w_\ell), m$). The encryption algorithm takes in the public parameters, an arbitrary length string w of symbols, and a message $m \in \mathbb{G}_T$. Let w_i denote the i -th symbol of w and ℓ denote the length of w . The encryption algorithm chooses random $s_0, \dots, s_\ell \in \mathbb{Z}_p$ and creates the ciphertext as follows.

First, it sets

$$C_m = m \cdot e(g, g)^{\alpha \cdot s_\ell} \text{ and}$$

$$C_{\text{start}1} = C_{0,1} = g^{s_0}, \quad C_{\text{start}2} = (h_{\text{start}})^{s_0}$$

Then, for $i = 1$ to ℓ it sets:

$$C_{i,1} = g^{s_i}, \quad C_{i,2} = (h_{w_i})^{s_i} z^{s_{i-1}}.$$

Finally, it sets

$$C_{\text{end}1} = C_{\ell,1} = g^{s_\ell}, \quad C_{\text{end}2} = (h_{\text{end}})^{s_\ell}$$

The output ciphertext is

$$\text{CT} = (w, C_m, C_{\text{start}1}, C_{\text{start}2}, (C_{1,1}, C_{1,2}), \dots, (C_{\ell,1}, C_{\ell,2}), C_{\text{end}1}, C_{\text{end}2})$$

KeyGen(MSK, $M = (Q, \mathcal{T}, q_0, F)$). The key generation algorithm takes as input the master secret key and the description of a deterministic finite state machine M . The description of M includes a set Q of states $q_0, \dots, q_{|Q|-1}$ and a set of transitions \mathcal{T} where each transition $t \in \mathcal{T}$ is a 3-tuple $(x, y, \sigma) \in Q \times Q \times \Sigma$. In addition, q_0 is designated as a unique start state and $F \subseteq Q$ is the set of accept states. (Recall, Σ is given in the parameters.)

The algorithm begins by choosing $|Q|$ random group elements $D_0, D_1, \dots, D_{|Q|-1} \in \mathbb{G}$, where we associate D_i with state q_i . Next, for each $t \in \mathcal{T}$ it chooses random $r_t \in \mathbb{Z}_p$; it also chooses random $r_{\text{start}} \in \mathbb{Z}_p$ and $\forall q_x \in F$ it chooses random $r_{\text{end},x} \in \mathbb{Z}_p$.

It begins creating the key with

$$K_{\text{start}1} = D_0(h_{\text{start}})^{r_{\text{start}}}, \quad K_{\text{start}2} = g^{r_{\text{start}}}$$

For each $t = (x, y, \sigma) \in \mathcal{T}$ the algorithm creates the key components

$$K_{t,1} = D_x^{-1} z^{r_t}, \quad K_{t,2} = g^{r_t}, \quad K_{t,3} = D_y(h_\sigma)^{r_t}.$$

Finally, for each $q_x \in F$ it computes

$$K_{\text{end}x,1} = g^{-\alpha} \cdot D_x(h_{\text{end}})^{r_{\text{end},x}}, \quad K_{\text{end}x,2} = g^{r_{\text{end},x}}$$

$$\text{SK} = (M, K_{\text{start}1}, K_{\text{start}2}, \forall t \in \mathcal{T} (K_{t,1}, K_{t,2}, K_{t,3}), \forall q_x \in F (K_{\text{end}x,1}, K_{\text{end}x,2}))$$

Decrypt(SK, CT). Suppose we are given a ciphertext CT associated with a string $w = w_1, \dots, w_\ell$ and a secret key SK associated with a DFA $M = (Q, \mathcal{T}, q_0, F)$ where $\text{ACCEPT}(M, w)$. Then there exist a sequence of $\ell + 1$ states u_0, u_1, \dots, u_ℓ and ℓ transitions t_1, \dots, t_ℓ where $u_0 = q_0$ and $u_\ell \in F$ and for $i = 1, \dots, \ell$ we have $t_i = (u_{i-1}, u_i, w_i) \in \mathcal{T}$.

The algorithm begins by computing:

$$B_0 = e(C_{\text{start}_1}, K_{\text{start}_1}) \cdot e(C_{\text{start}_2}, K_{\text{start}_2})^{-1} = e(g, D_0)^{s_0}.$$

Then for $i = 1$ to ℓ it iteratively computes:

$$B_i = B_{i-1} \cdot e(C_{(i-1),1}, K_{t_i,1})e(C_{i,2}, K_{t_i,2})^{-1}e(C_{i,1}, K_{t_i,3}) = e(g, D_{u_i})^{s_i}$$

Since the machine accepts we have that $u_\ell = q_x$ for some $q_x \in F$ and $B_\ell = e(g, D_x)^{s_\ell}$.

It finally computes

$$B_{\text{end}} = B_\ell \cdot e(C_{\text{end}_x,1}, K_{\text{end}_x,1})^{-1} \cdot e(C_{\text{end}_x,2}, K_{\text{end}_x,2}) = e(g, g)^{\alpha s_\ell}.$$

This value can then be divided from C_m to recover the message m .

3.2 Further Discussion

As discussed in the introduction, our construction contains three primary mechanisms used for decryption. The first step of the decryption process gives what in the introduction we called the initiation mechanism, which starts by computing $e(g, D_0)^{s_0}$. This used the “start” values from the keys and ciphertexts. We observe that this mechanism has structural similarities to the Boneh-Boyen [5] Identity-Based Encryption system.

The next several steps of decryption provide the transition mechanism. The evaluation of $e(C_{(i-1),1}, K_{t_i,1})e(C_{i,2}, K_{t_i,2})^{-1}e(C_{i,1}, K_{t_i,3})$ results in the term $(e(g, D_x)^{s_{i-1}})^{-1}e(g, D_y)^{s_i}$, which updates the accumulated value to $e(g, D_y)^{s_i}$. Representing that the machine M is in state y after processing i symbols of w . A paramount feature is that the $C_{i,2}$ components of the ciphertext “chain” adjacent symbols together. This along with the structure of the private key enforces that $(e(g, D_x)^{s_{i-1}})^{-1}e(g, D_y)^{s_i}$ can only be computed if the i -th symbol is σ and the transition from state x to y on symbol σ is in the DFA for some σ .

Finally, the completion mechanism allows for the computation of $e(g, g)^{\alpha s_\ell}$ if the accumulated value reaches $e(g, D_x)^{s_\ell}$ for $q_x \in F$. The completion mechanism is very close in design to the initiation mechanism, but has the master key $g^{-\alpha}$ multiplied into its key component.

As described in the introduction, an attacker can backtrack to get some accumulated value that may not represent the actual computation of M on w . However, the attacker intuitively will only be able to backtrack so far and since

M is deterministic must eventually go forward again to the same spot if he is to decrypt. This intuition is captured rigorously in the security proof in the next section.

3.3 Rerandomization of Ciphertexts and Secret Keys

We give two algorithms for the rerandomization of ciphertexts and private keys. Our ciphertext rerandomization algorithm takes in any well formed ciphertext for string w and produces a new ciphertext for the string w which encrypts the same message as the original. Moreover, the output ciphertext has the same distribution as one created fresh from the encryption algorithm. Similarly, the key rerandomization algorithm takes any valid secret key SK for DFA M and outputs a new secret key for DFA M that where the distribution is the same as the KeyGen algorithm.

These algorithms will be used for proving security of our system in the next section. Our main reduction techniques will produce valid challenge ciphertexts and private keys; however, these might not be well distributed. The randomization algorithms can then be applied to get the properly distribution on ciphertexts and keys. We segregate rerandomization as a separate step to help simplify the presentation of our proofs.⁴ Our algorithms do simple additive (in the exponent) rerandomization. The presentation of these algorithms is in the full version of our paper.

4 Security Proof

We now prove security of our construction in the selective mode. We assume a successful attacker \mathcal{A} against our system. Our reduction algorithm \mathcal{B} will run \mathcal{A} and use it to break the decision ℓ^* -Expanded BDHE assumption, where ℓ^* is the length of the string w^* used in creating the challenge ciphertext. Our reduction describes how \mathcal{B} simulates the Setup, Key Generation, and Challenge Ciphertext generation phases of the security game.

We prove the following theorem.

Theorem 1. *Suppose the decision ℓ^* -Expanded BDHE assumption holds. Then no poly-time adversary can selectively break our DFA-based encryption system where the challenge string w^* encrypted is of length ℓ^* .*

Suppose we have an adversary \mathcal{A} with non-negligible advantage $\epsilon = \text{Adv}_{\mathcal{A}}$ in the selective security game against our construction for alphabet Σ . Moreover, suppose it chooses a challenge string w^* of length ℓ^* . Then \mathcal{B} runs \mathcal{A} and simulates the security game as follows.

Init \mathcal{B} takes a decision ℓ^* -Expanded BDHE challenge \mathbf{X}, T . The attacker, \mathcal{A} , declares a challenge string w^* of length ℓ^* . We let w^*_j denote the j -th symbol of w^* . In addition, we define $w^*_{\ell^*+1} = w^*_0 = \perp$ for a special symbol $\perp \notin \Sigma$.

⁴ Most prior reductions of a similar nature (e.g. [5,6,28]) build the rerandomization directly in the main reduction.

Setup. The reduction algorithm first chooses random exponents⁵ $v_z, v_{\text{start}}, v_{\text{end}} \in \mathbb{Z}_p$ and $\forall \sigma \in \Sigma v_\sigma$. The parameter values are chosen as follows:

$$e(g, g)^\alpha = e(g^a, g^b), g = g, z = g^{v_z} g^{ab/d}$$

This implicitly sets $\alpha = ab$. Next it sets:

$$h_{\text{start}} = g^{v_{\text{start}}} \prod_{j \in [1, \ell^*]} g^{-a^j b/c_j}, \quad h_{\text{end}} = g^{v_{\text{end}}} \prod_{j \in [2, \ell^*+1]} g^{-a^j b/c_j}.$$

Finally,

$$\forall_{\sigma \in \Sigma} h_\sigma = g^{v_\sigma} g^{-b/d} \cdot \prod_{\substack{j \in [0, \ell^*+1] \\ w^*_j \neq \sigma}} g^{-a^{(\ell^*+1-j)} b/c_{(\ell^*+1-j)}}.$$

The reduction algorithm embeds its knowledge of w^* into the public parameters. The parameter h_σ will have a term $g^{-a^{\ell^*+1-j} b/c_{\ell^*+1-j}}$ if and only if the j -th symbol of w^* is not σ . As we will see this embedding will be crucial to simulating a challenge ciphertext, while maintaining the ability to generate keys. The terms are well distributed since a is chosen randomly in the assumption and due to the ‘ v ’ exponents, which randomize the other parameters. We also observe that this equation is well defined since we defined $w^*_{\ell^*+1} = w^*_0 = \perp$. Since $\perp \notin \Sigma$ we have that for all σ the parameter h_σ always contains the terms $g^{-ba^{(\ell^*+1)}/c_{(\ell^*+1)}}$ and g^{-ba^0/c_0} .

Challenge. We describe how \mathcal{B} creates a challenge ciphertext. The reduction will intuitively set $s_i = sa^i \in \mathbb{Z}_p$. \mathcal{B} first chooses a coin β and begins to create a ciphertext CT. It first sets $w = w^*$ (from Init) and sets $C_m = m_\beta \cdot T$. Next, it sets

$$C_{\text{start}1} = g^s, \quad C_{\text{start}2} = (h_{\text{start}})^s = (g^s)^{v_{\text{start}}} \prod_{j \in [1, \ell^*]} g^{-a^j bs/c_j}$$

Then, for $i = 1$ to ℓ^* :

$$C_{i,1} = g^{a^i s}, \quad C_{i,2} = (g^{a^i s})^{v_{w^*_i}} (g^{a^{i-1} s})^{v_z} \cdot \prod_{\substack{j \in [0, \ell^*+1] \\ \text{s.t. } w^*_j \neq w^*_i}} g^{(-a^{\ell^*+1-j+i}) bs/c_j}.$$

We make two observations. First, the algorithm \mathcal{B} does not receive from the assumption the term $g^{-a^{\ell^*+1} bs/c_j}$ for any j . Thus, it is important that such a term not be included in $C_{i,2}$. We see that such a term can only appear in the product above when $i = j$, however, this cannot be the case since if $i = j$ then $w^*_j = w^*_i$, which is explicitly excluded from the product.

⁵ These values will be used to ensure that the public parameters are distributed as in the real system. While this is of course necessary, they are not central to the core ideas of the reduction. As a simplification, a reader might choose to ignore these (imagine they are all 0) to help understand the main ideas.

Second, we remark that $z^{s_{i-1}}$ produces a term $g^{a^i b s/d}$ which \mathcal{B} does not have. However, $(h_{w^*_i})^{s_i}$ produces a term which is the inverse of this and these cancel each other out in the computation of $C_{i,2} = z^{s_{i-1}}(h_{w^*_i})^{s_i}$. The remaining terms shown above are producable from the assumption.

These two observations reflect important points in the proof. The first shows how the embedding of the challenge string w^* in the h_σ values allows us the creation of the challenge ciphertext. The second cancellation reflects the “linking” of adjacent symbols of w^* that is at the core of the security of the ciphertext construction.

Finally, it creates

$$C_{\text{end}1} = g^{a^{\ell^*} s}, \quad C_{\text{end}2} = (h_{\text{end}})^{a^{\ell^*} s} = (g^{a^{\ell^*} s})^{v_{\text{end}}} \prod_{j \in [2, \ell^*+1]} g^{-a^{\ell^*+j} b s / c_j}$$

To finish, the \mathcal{B} must run the ciphertext rerandomization algorithm (specified in the full version) on CT to get a well distributed challenge ciphertext CT*. It then returns CT* to \mathcal{A} . If $T = g^{a^{\ell^*+1} b s}$ — is a valid Expanded BDHE tuple — then (the rerandomized) CT* is an encryption of m_β . Otherwise the ciphertext will reveal no information about β .

Key Generation. The key generation phases (1 and 2) are the most complex part of this reduction. We describe how \mathcal{B} can respond to a private key request for DFA $M = (Q, \mathcal{T}, q_0, F)$.

Before showing how \mathcal{B} creates the actual key components we start by doing some prep work to show how the D_x values will implicitly be assigned. \mathcal{B} will not actually be able to directly create these. We will describe the terms which comprise each D_x value and will create the keys to be consistent with these. Intuitively, the assignments should in some way match the execution of machine M on w^* .

We begin by introducing some notation. First we let $w^{*(i)}$ denote the last i symbols of w^* . It follows that $w^{*(\ell^*)} = w^*$ and $w^{*(0)}$ is the empty string. In addition, we for $k \in [0, |Q| - 1]$ we let $M_k = (Q, \mathcal{T}, q_k, F)$. That is M_k is the same DFA as M except the start state is changed to q_k . (Note that $M_0 = M$.)

Now for each $q_k \in Q$ we create a set S_k of indices between 0 and ℓ^* . For $i = 0, 1, \dots, \ell^*$ we put $i \in S_k$ if and only if $\text{ACCEPT}(M_k, w^{*(i)})$. Then we assign

$$D_k = \prod_{i \in S_k} g^{a^{i+1} b}.$$

This assignment of D_k values is meant to “mark” the computation of M on the challenge string w^* . The term $g^{a^{i+1} b}$ will appear in D_k if it is possible to reach an accepting final state using the last i symbols of w^* starting at state q_k . We make two important observations for creating private keys. First, the term $g^{a^{\ell^*+1} b}$ does not appear in D_0 . This follows from the fact that any valid request of a DFA M must reject w^* and is crucial to creating $K_{\text{start}1}$. Next for all $x \in F$, we have that the term $g^{a^1 b}$ does appear in D_x . This occurs since M_x will accept

the empty string if it starts at an accepting state x and is a critical fact needed for creating $K_{\text{end}_x,1}$.

The values embedded in D_k reflect more than just the computation of M on w^* . It embeds the execution of all M_k (M with all possible starting states) from all positions of the string w^* . This is done to reflect the possible backtracking attacks described in the last section. We emphasize that \mathcal{B} cannot actually produce these D_k values using the terms given from the assumption. Instead it will construct the key components to be consistent with these values. Uncomputable terms will cancel when creating the components. \mathcal{B} begins with creating the start and end key terms.

\mathcal{B} starts by implicitly setting $r_{\text{start}} = \sum_{i \in S_0} c_{i+1}$.

$$K_{\text{start}2} = g^{r_{\text{start}}} = \prod_{i \in S_0} g^{c_{i+1}}$$

$$K_{\text{start}1} = D_0(h_{\text{start}})^{r_{\text{start}}} = (K_{\text{start}2})^{v_{\text{start}}} \cdot \prod_{\substack{j \in [1, \ell^*], i \in S_0 \\ j \neq i+1}} g^{-a^j b c_{i+1}/c_j}$$

Our assignments canceled out the terms of the form $g^{a^{i+1}b}$ from D_0 and the remaining terms that are given in $K_{\text{start}1}$ are those that we are given from the assumption. Notice that since the term $g^{a^{\ell^*+1}b}$ was not in D_0 it did not need to be canceled; this is important since the setting of h_{start} gave no way to do this. (The term $g^{a^{\ell^*+1}b/c_{\ell^*+1}}$ is not in h_{start} .)

Next, for all $q_x \in F$ it creates the key components. It first creates starts by implicitly setting $r_{\text{end}_x} = \sum_{i \in S_x, i \neq 0} c_{i+1}$.

$$K_{\text{end}2,x} = g^{r_{\text{end}_x}} = \prod_{\substack{i \in S_x \\ i \neq 0}} g^{c_{i+1}}$$

$$K_{\text{end}1,x} = g^{-\alpha} D_x(h_{\text{end}})^{r_{\text{end}_x}} = (K_{\text{end}2,x})^{v_{\text{end}}} \cdot \prod_{\substack{j \in [2, \ell^*+1], i \in S_x \\ i \neq 0, j \neq i+1}} g^{-a^j b c_{i+1}/c_j}$$

Our assignment of $K_{\text{end}2,x}$ canceled out the terms of the form $g^{a^{i+1}b}$ from D_x except for when $i = 0$. Here, D_x has the term g^{ab} and this cancels with $g^{-\alpha} = g^{-ab}$. It is essential that D_x contain the term g^{ab} since h_{end} is structured such there is no other way of canceling with $g^{-\alpha}$. (The term $g^{a^1 b/c_1}$ is not in h_{end} .)

We finally need to create the key components for each $t = (x, y, \sigma) \in \mathcal{T}$. We organize this into a set of intermediate computations. For $i = 0$ to $\ell^* + 1$ we will define $(K_{t,1,i}, K_{t,2,i}, K_{t,3,i})$. Then we will let $K_{t,1} = \prod_{i \in [0, \ell^*+1]} K_{t,1,i}$, $K_{t,2} = \prod_{i \in [0, \ell^*+1]} K_{t,2,i}$, $K_{t,3} = \prod_{i \in [0, \ell^*+1]} K_{t,3,i}$.

Intuitively, for each transition $t = (x, y, \sigma)$ we step through each i from 0 to ℓ^* .⁶ For each i we describe how to set $K_{t,2,i}$ such that it will cancel $g^{a^{i+1}b}$

⁶ We explicitly note that -1 and $\ell^* + 1 \notin S_k$ for all k , which allows these cases to all be well defined for the range of i .

from D_x if this term appears in D_x and how to cancel $g^{a^i b}$ from D_y if this term appears in D_y in computing the $K_{t,1}, K_{t,3}$ key components. We step through four possible cases.

Case 1: $i \notin S_x \wedge (i - 1) \notin S_y$

Set $K_{t,1,i}, K_{t,2,i}, K_{t,3,i} = 1$ (the identity element). This is when there are not

term $g^{a^{i+1} b}$ in either D_x nor term $g^{a^i b}$ in D_y so nothing needs to be canceled.

Case 2: $i \in S_x \wedge (i - 1) \in S_y$

Set $K_{t,2,i} = g^{a^i d}$ and $K_{t,1,i} = (K_{t,2,i})^{v_z}$.

$$K_{t,3,i} = (K_{t,2,i})^{v_\sigma} \cdot \prod_{\substack{j \in [0, \ell^* + 1] \\ w^*_j \neq \sigma}} g^{-a^{(\ell^* + 1 - j + i)} bd / c_{(\ell^* + 1 - j)}}$$

This is when there are terms $g^{a^{i+1} b}$ in D_x and $g^{a^i b}$ in D_y . The setting of $K_{t,2,i}$ allows them to both be canceled and the remaining terms above are “collateral” which can be taken from the assumption. This action is independent of the symbol w^*_{i+1} . We can think of the setting of $K_{t,2,i}$ as a “copy action” in that a similar cancellation happens on both sides of the transition.

Case 3: $i \notin S_x \wedge (i - 1) \in S_y \wedge w^*_{\ell^* + 1 - i} \neq \sigma$

Set $K_{t,2,i} = g^{c_i}$ and $K_{t,1,i} = (K_{t,2,i})^{v_z} g^{abc_i / d}$.

$$K_{t,3,i} = (K_{t,2,i})^{v_\sigma} \cdot g^{-bc_i / d} \cdot \prod_{\substack{j \in [0, \ell^* + 1] \\ j \neq \ell^* + 1 - i \wedge w^*_j \neq \sigma}} g^{-a^{(\ell^* + 1 - j)} bc_i / c_{(\ell^* + 1 - j)}}$$

In this case there is *not* a term $g^{a^{i+1} b}$ in D_x , but there is a term $g^{a^i b}$ in D_y . Therefore we cannot apply the above “copy” technique from Case 2. Instead we use the fact that $w^*_{\ell^* + 1 - i} \neq \sigma$. (Note that $w^*_{\ell^* + 1 - i}$ is the first symbol of the string $w^{*(i)}$ — the string of the last i symbols of w^* .) We set $K_{t,2,i} = g^{c_i}$ which cancels the only g^{a^i} term in D_y . Since $w^*_{\ell^* + 1 - i} \neq \sigma$ we have that h_σ contains the term $g^{-a^i b / c_i}$. Raising this to c_i provides the desired cancellation. The remaining terms shown above are “collateral” that can be taken from the assumption.

Case 4: $i \in S_x \wedge (i - 1) \notin S_y \wedge w^*_{\ell^* + 1 - i} \neq \sigma$

First set, $K_{t,2,i} = g^{a^i d} g^{-c_i}$ and $K_{t,1,i} = (K_{t,2,i})^{v_z} g^{-abc_i / d}$.

Then, $K_{t,3,i} = (K_{t,2,i})^{v_\sigma} \cdot g^{bc_i / d}$.

$$\prod_{\substack{j \in [0, \ell^* + 1] \\ w^*_j \neq \sigma}} g^{-a^{(\ell^* + 1 - j + i)} bd / c_{(\ell^* + 1 - j)}} \prod_{\substack{j \in [0, \ell^* + 1] \\ j \neq \ell^* + 1 - i \wedge w^*_j \neq \sigma}} g^{a^{(\ell^* + 1 - j)} bc_i / c_{(\ell^* + 1 - j)}}$$

How we handle this case can best be understood as a combination of how we handle Cases 2 and three. Here there is a term $g^{a^{i+1} b}$ in D_x , but there

is a *not* term g^{a^ib} in D_y . We first put a term g^{a^id} in $K_{t,2,i}$ to invoke the “copy” mechanism from Case 2. This gives us the desired cancellation for part of D_x , but also creates an undesirable term g^{a^ib} that cannot be cancelled with D_y . Therefore we also include a term of g^{-c_i} in $K_{t,2,i}$ to invoke the cancellation of the “undesirable term”. As in Case 3, we have that $w^{\ell^*+1-i} \neq \sigma$ and h_σ contains the term g^{-a^ib/c_i} . Again, remaining terms shown above are “collateral” which can be taken from the assumption. *We observe that these terms are basically just those generated from Case 2 and Case 3 combined.*

These four cases cover all possibilities. By the definition of a DFA when $w^{\ell^*+1-i} = \sigma$ we have that $i \in S_x$ if and only if $i-1 \in S_y$. This is a consequence of the fact that M is deterministic. (If M were instead a Nondeterministic Finite Automata, the prior statement would not hold.)

This shows how \mathcal{B} creates all the key components. \mathcal{B} must run the key rerandomization algorithm from the previous section on SK to get a well distributed key $\tilde{\text{SK}}$ for the machine M . Then it returns $\tilde{\text{SK}}$ to \mathcal{A} .

Guess. The adversary will eventually output a guess β' of β . If $\beta = \beta'$, then \mathcal{B} then outputs 0 to guess that $T = e(g, g)^{a^{\ell^*+1}bs}$; otherwise, it outputs 1 to indicate that it believes T is a random group element in \mathbb{G}_T .

When T is a tuple the simulator \mathcal{B} gives a perfect simulation so we have that

$$\Pr \left[\mathcal{B} \left(\mathbf{y}, T = e(g, g)^{a^{\ell^*+1}bs} \right) = 0 \right] = \frac{1}{2} + \text{Adv}_{\mathcal{A}}.$$

When T is a random group element the message \mathcal{M}_β is completely hidden from the adversary and we have $\Pr [\mathcal{B}(\mathbf{X}, T = R) = 0] = \frac{1}{2}$. Therefore, \mathcal{B} can play the decision ℓ^* -Expanded BDHE game with non-negligible advantage.

5 Conclusions

We introduced a new type of functional encryption system that works over regular languages. Our system has secret keys that encode a Deterministic Finite Automata M and ciphertexts that are associated with an arbitrary length string w and an encrypted message m . A user with a secret key for DFA M can decrypt a ciphertext associated with w if and only if M accepts w . Our construction makes use of bilinear maps and constructs mechanisms that enforce the DFA evaluation.

Interesting future challenges include developing a system that can be proved adaptively secure and under a non-parameterized assumption. Looking farther out, one would like to be able to extend the functionality further, eventually all the way to support Turing Machines.

Acknowledgements. We thank Dan Boneh, Allison Lewko, and Amit Sahai for helpful comments and feedback.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *J. Cryptology* 21(3), 350–391 (2008)
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) *EUROCRYPT* 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)
3. Agrawal, S., Boneh, D., Boyen, X.: Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. In: Rabin, T. (ed.) *CRYPTO* 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
5. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT* 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: Franklin, M. (ed.) *CRYPTO* 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) *EUROCRYPT* 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
8. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT* 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
9. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.* 32(3), 586–615 (2003); Extended Abstract In: Kilian, J. (ed.) *CRYPTO* 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
10. Boneh, D., Hamburg, M.: Generalized Identity Based and Broadcast Encryption Schemes. In: Pieprzyk, J. (ed.) *ASIACRYPT* 2008. LNCS, vol. 5350, pp. 455–470. Springer, Heidelberg (2008)
11. Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: Ishai, Y. (ed.) *TCC* 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
12. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) *TCC* 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
13. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) *EUROCRYPT* 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
14. Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: Honary, B. (ed.) *Cryptography and Coding* 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
15. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) *EUROCRYPT* 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
16. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: *STOC*, pp. 197–206 (2008)

17. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded Ciphertext Policy Attribute Based Encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórssón, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
18. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
19. Hamburg, M.: Spatial encryption. IACR Cryptology ePrint Archive, 2011:389 (2011)
20. Iovino, V., Persiano, G.: Hidden-Vector Encryption with Groups of Prime Order. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 75–88. Springer, Heidelberg (2008)
21. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
22. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
23. Okamoto, T., Takashima, K.: Hierarchical Predicate Encryption for Inner-Products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
24. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
25. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
26. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
27. Sipser, M.: Introduction to the theory of computation. Thomson Course Technology (2006)
28. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
29. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
30. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)

Secure Database Commitments and Universal Arguments of Quasi Knowledge

Melissa Chase and Ivan Visconti

¹ Microsoft Research, Redmond, USA

melissac@microsoft.com

² Dipartimento di Informatica, University of Salerno, Italy

visconti@dia.unisa.it

Abstract. In this work we focus on a simple database commitment functionality where besides the standard security properties, one would like to hide the size of the input of the sender. Hiding the size of the input of a player is a critical requirement in some applications, and relatively few works have considered it. Notable exceptions are the work on zero-knowledge sets introduced in [14], and recent work on size-hiding private set intersection [1]. However, neither of these achieves a secure computation (i.e., a reduction of a real-world attack of a malicious adversary into an ideal-world attack) of the proposed functionality.

The first result of this submission consists in defining “secure” database commitment and in observing that previous constructions do not satisfy this definition. This leaves open the question of whether there is any way this functionality can be achieved.

We then provide an affirmative answer to this question by using new techniques that combined together achieve “secure” database commitment. Our construction is in particular optimized to require only a constant number of rounds, to provide non-interactive proofs on the content of the database, and to rely on the existence of a family of CRHFs. This is the first result where input-size hiding secure computation is achieved for an interesting functionality and moreover we obtain this result with standard security (i.e., simulation in expected polynomial time against fully malicious adversaries, without random oracles, without non-black-box extraction assumptions, without hardness assumptions against super-polynomial time adversaries).

A key building block in our construction is a universal argument enjoying an improved proof of knowledge property, that we call quasi-knowledge. This property is significantly closer to the standard proof of knowledge property than the weak proof of knowledge property satisfied by previous constructions.

Keywords: ZK sets, universal arguments, input-size hiding security.

1 Introduction

Secure Computation. The standard notion of “security” for any multi-party computation [10] involves describing an ideal model where parties have access

to a trusted functionality which will carry out the desired computation without revealing any information about the parties' secret inputs and outputs. Then, very informally, given a communication network in the real world, without any trusted functionality, a protocol π securely realizes a multi-party computation if any successful attack by an adversary of π can be translated into a successful attack in the ideal world. Since the latter is trivially secure this means π is secure in the real world as well.

Therefore, the real-world/ideal-world paradigm [10] allows one to prove the security of protocols in a simulation-based fashion, with strong security guarantees. This turns out to be very useful when protocols are used as subprotocols and provides robust security when the ideal functionality is correctly designed. All the fundamental primitives have been defined in terms of ideal functionalities and thus they can be securely realized in the real-world/ideal-world paradigm using the general results of [10].

However the current state-of-the art still does not properly address the issue of considering *the size of the input* of a player as information to be kept private. Here we consider the problem of hiding the input size in one of the most basic primitives: a commitment scheme.

Secure Database Commitments. Here we address the case where a player wants to commit to a large set of data, and then to partially open that commitment, both without revealing the size of the committed set. More specifically, we consider the setting where a party (the sender) may want to commit to a large elementary database composed by key-value pairs, without revealing its size. Then in the opening phase, the sender might want to reveal part of his database one item at a time depending on queries of the receiver. In particular, the receiver will ask for some keys, and the sender wants to convince the receiver of the value associated to each requested key. These partial opening queries should reveal no information about the rest of the database, including the size.

The Main Question: Feasibility of Secure Database Commitments. Following the above discussion, and the fact that we have general results for achieving secure multi-party computation, One could ask the following natural question: "Why should we focus on the design of protocols for secure database commitments if we can use the known constructions for secure two-party computation?". There is a crucial difference between this problem, and the one considered in [10] and in all subsequent work (to the best of our knowledge). Our notion of secure database commitment critically requires that the size of the prover's input must be hidden. In contrast, the [10] definition, according to [9] and to the known constructions, allows a party either at the beginning of or during the computation, to learn the size of the other party's input. This information is actually revealed in all known protocols, mainly because many of the tools used to allow extraction of the adversary's implicit input (e.g. zero-knowledge proofs of knowledge) have communication that depends directly on the size of the input.

Consequently, traditional results for secure two-party computation cannot be used to obtain secure database commitments, and we need to develop new tools

and a significantly new approach. This presents the following interesting open problem: *Is the notion of secure database commitment and more generally a meaningful notion of input-size hiding secure computation achievable?*

Difficulties in Achieving Input-Size Hiding Secure Computation with Standard Assumptions. We stress that the main challenge in size-hiding secure computation is as follows: when proving security against a real world adversarial prover, we need to design a simulator which can extract the input from the adversary so that it can provide it to the ideal functionality. At the same time, we can not assume any fixed polynomial upper bound for the size of the set (the protocol should work for *any* polynomial sized input), and we can not allow the amount of communication to vary with the size of the input.

The real difficulties lie in obtaining a standard security notion (showing an efficient ideal adversary from an efficient real-world adversary), therefore avoiding controversial (sometimes not even falsifiable) conjectures such as random oracles, non-black-box extraction assumptions (e.g., the Diffie-Hellman knowledge of exponent assumption and its variations [11]), complexity leveraging (i.e., hardness assumptions against super-polynomial time adversaries) and so on.

Relationship to Zero-Knowledge Sets. We note that the requirements described for secure database commitments are essentially those given in [14] where Micali, Rabin and Kilian introduced the concept of a zero-knowledge set (ZKS). This primitive allows a party to first commit to a database, and later to answer queries on that database and to prove that the responses are consistent with the original commitment, as in the secure database commitments described above.

However, the definition given by [14] and by all subsequent papers is a *property-based definition* that requires: 1) soundness: the commitment should be binding, i.e., for each query there should be only one response for which the prover can give a convincing proof; and 2) zero-knowledge: both the commitment and the proofs should not reveal any information about the rest of the database (beyond the information that is asked in the query), not even the number of elements it contains.

Furthermore, we argue that the constructions they provide (and later constructions for ZKS, see [7,5,8,6,15,13]) do not satisfy a typical real-world/ideal-world definition in the spirit of “secure computation”. To see this, note that all previous schemes for ZKS included a non-interactive commitment of the set. As mentioned above, we do not consider non-black-box extraction assumptions; moreover, standard non-black-box techniques introduced in [2] so far have been successfully used only in conjunction with interaction.

However, we argue that black box extraction is impossible for a scheme which is size hiding and has a non-interactive commitment phase. This follows because such a scheme must allow for sets of *any* polynomial size, which means there will be at least $2^{\text{superpoly}(k)}$ possible sets; at the same time, the length of the non-interactive commitment must be bounded by a fixed polynomial in k . Thus, a simple counting argument shows that there is no way (based on standard assumptions and security notions) that a simulator can correctly identify the

committed set given only the single commitment message. Note that this argument also holds in the CRS model, as long as the CRS is of length polynomial in k , as even in this case the simulator still gets only $\text{poly}(k)$ bits of information to identify one database out of $2^{\text{superpoly}(k)}$. Therefore, no previous construction of zero-knowledge sets can be proved to be a secure realization of the database commitment functionality (with a black box simulator).

Looking ahead, in our construction we will avoid these limitations by allowing an interactive commitment phase. The possibility of using interaction was already mentioned in some previous work, but only for the query/proof phase, thus without making any contribution towards addressing this extraction problem. Instead, our goal is to use interaction in the commitment phase, still keeping the query/answer phase essentially non-interactive.

Other Related Work. Recently similar issues have been considered by Ateniese et al. in [1] for the problem of Private Set Intersection. However their solution uses random oracles, and obtains security with respect to semi-honest adversaries only. The goal of our work is to obtain security against malicious players as required in secure computation, and we will obtain this result in the standard model using only standard complexity-theoretic assumptions.

Timing Attacks. We note that there may be some cases in which any attempt to hide the size of inputs is vulnerable to timing attacks as discussed in [12] (in which an adversary can guess the size of the input depending on the amount of the time required to perform computations). However, there are many settings where these attacks will have only limited impact. Notice that since the adversary does not necessarily know the amount of resources of the other player (the committer could perform his computation by distributing the workload among computing devices in number proportional to the size of the input) the timing may in fact give very little information.

1.1 Our Results

In this work we first put forth the notion of secure database commitment. Following the traditional notion of “secure computation” we define the natural ideal functionality for database commitment \mathcal{F}_{DbCom} and observe that it implies all required security guarantees needed by zero-knowledge sets.

We then present a constant-round protocol that securely realizes the \mathcal{F}_{DbCom} functionality. The protocol is interactive, and is secure in the standard model (i.e., we do not require any set-up assumptions) based on the existence of families of collision-resistant hash functions (CRHFs, for short) (notice that CRHFs are implied by the existence of ZK sets [5]).

We stress that our protocol has optimal amortized round complexity, as queries and answers are essentially non-interactive. In addition, the use of the real-world/ideal-world paradigm, and the simulation in polynomial time should make it much easier to use this primitive as part of a larger protocol.

Our construction is based on a special universal argument that enjoys a new proof of knowledge property which is closer to the traditional proof of knowledge

property. We define this new property, give a construction which satisfies it based on CRHFs, and show how it can be used to implement secure database commitments. (However, we stress that there were many other subtle issues to address, and our stronger notion of universal argument alone is not sufficient for directly obtaining input-size hiding secure computation.)

Techniques. As described above, the biggest challenge is in defining a simulator that can extract a witness whose length can be any polynomial, from only a fixed polynomial amount of communication. Note that the standard approach does not work in this setting: it might seem that a simple solution would be to ask the sender to give a commitment to the database, and an interactive zero-knowledge proof of knowledge of the database contained in the provided commitment. However, in general such proofs may reveal the size of the witness, which in this case means the size of the database.

Instead, we use a different tool, namely a witness indistinguishable universal argument of quasi knowledge (UAQK). A universal argument is a (computationally-sound) proof system which guarantees that the communication is only proportional to the size of the statement. At the same time, it guarantees that the honest prover can run in time polynomial in the size of the witness, which is the best that we can hope for.

Universal arguments were introduced by Barak [2] as part of the design of a non-black-box simulator. Barak showed a construction based on CRHFs which satisfied a weak proof of knowledge property. However, there are some inherent challenges in defining a standard proof of knowledge for universal arguments, and the extraction guarantees of his definition do not seem to be sufficient for our application. To deal with this we define a new proof of knowledge property which we call “quasi knowledge” which provides a functionality somewhat closer to that of a standard proof of knowledge. We show that it can be used in our application to implement the traditional commitment and proof of knowledge strategy described above. (Of course we are glossing over many issues here, and the application is not straightforward at all - see Section 4 for details.)

Finally, we note that this is of additional interest, because we use universal arguments for a completely different purpose than the one that motivated Barak’s original construction [2], which was the design of a non-black-box simulator.

Universal Arguments: From Weak Proof of Knowledge to Proof of Quasi Knowledge. The standard proof of knowledge property guarantees that if a prover can convince a verifier of the truthfulness of a theorem with some probability, then one can efficiently obtain an NP witness for that theorem with the same probability (up to negligible factors).

When one focuses instead on universal languages (i.e., when one would like to prove that a Turing machine accepts the provided input within a given — not necessarily polynomial — number of steps), then one can not always efficiently obtain a corresponding witness since its length can be superpolynomial. In this case a restricted proof of knowledge property might instead focus on extracting an implicit representation of the witness, in the form of a polynomial-sized circuit that when evaluated on input i provides the i -th bit of the witness.

Unfortunately there is no known construction of a universal argument with such a property. The current state of the art, [4], shows how to get a *weak* proof of knowledge property that includes one more restriction: when a prover proves a theorem with probability $1/q$, one can efficiently get an implicit representation of a circuit with probability $1/q'$ that is polynomially related to $1/q$. This essentially means that one can efficiently get a *candidate* implicit representation of a witness, with non-negligible probability when $1/q$ is non-negligible. However, there is no guarantee that the candidate implicit representation is actually correct (one can not simply test the circuit asking for all bits of the witness since the size of a witness can be superpolynomial).

Furthermore, there is an additional subtlety in the way this weak proof of knowledge property is defined. For any polynomial q , there is guaranteed to be a polynomial time extractor that can extract candidate representations of the witness from any prover that proves a theorem with probability $1/q$, and this extractor is guaranteed to succeed with the related probability $1/q'$; however, the choice of this extractor and its running time may depend on q . This has two disadvantages: (1) in order to use an extractor, we must first determine which q we are interested in, and (2) an efficient extractor is required to exist only for polynomials q , while nothing is required when q is super polynomial. (In fact, in the construction given in [4], the running time of the extractor is roughly q^d where d is some constant greater than 1, therefore when q is superpolynomial the running time of the extractor increases quickly and its expected running time is not polynomial). As a result of these disadvantages, converting a weak proof of knowledge system into a proof system with the standard proof of knowledge property is very non-trivial, even when one is happy with the extraction of an implicit representation of the witness. (This is because the standard proof of knowledge property requires that, regardless of the success probability of the adversarial prover, the extractor must run in expected polynomial time.)

In this paper we remove the above additional restriction of the *weak* proof of knowledge property and replace it with a more standard proof of knowledge property, requiring that an extractor outputs in expected polynomial time a correct implicit representation of the witness. The only caveat is that, while we require that the extracted implicit representation must correspond to some true witness, we do allow it to contain a few errors, as long as those errors are unlikely to be noticed by any polynomial time process. (Note that if the witness is polynomial sized, then this still implies the standard proof of knowledge property.) We will say that an argument system is an argument of “quasi” knowledge if it enjoys this property.

Finally, we construct a constant-round witness-indistinguishable universal argument of quasi-knowledge under standard complexity-theoretic assumptions.

2 Universal Arguments of Quasi Knowledge

A universal argument is an interactive argument system for proving membership in **NEXP**. For the formal definition, see [4]. We will use the following result.

Theorem 1. ([4,3], informal) Suppose there exists a hash function ensemble that is collision-resistant against polynomial-sized circuits. Then there exists a universal argument system that is constant-round, public-coin and witness indistinguishable.

Moreover, there exists a construction which satisfies two additional properties. First, the construction of this extractor is parameterized by a lower bound $\alpha(n)$ on the success probability of the adversarial prover P_n^* . The resulting extractor, that we denote by $E(\alpha(n), \cdot, \cdot)$ has the property that there is a fixed constant d such that, $E(\alpha(n), y, \cdot)$ has expected running time at most $(\frac{n}{\alpha(n)})^d$ for any n -bit instance y .

Furthermore, it has the property that there is a fixed known polynomial q^* such that for any polynomial-sized prover P_n^* that succeeds in causing verifier $V(y)$ to accept with probability $q(n)$ for n -bit instance y , and for any $\alpha < q(n)$, the probability that the extractor $E^{P_n^*}(\alpha, y, \cdot)$ succeeds in extracting bits of a valid witness is at least $\frac{\alpha}{q^*(n)}$. For a more formal statement, see the full version. For details of the construction and proof, see the proof of Lemma A.2.5 in [3].

2.1 A New Notion: Quasi-Knowledge

As described in Section 1.1, we aim to construct a universal argument with a more standard proof of knowledge property. The resulting proof of knowledge will resemble the standard proof of knowledge property with two exceptions.

The first is that the extractor will produce only an implicit representation of a witness. This is necessary since UAs are used to prove statements that are not necessarily in NP, therefore the length of the witness may not be polynomial, but still we want our extractor to run in (expected) polynomial time. We formalize this saying that the extractor will be a circuit which on input i will produce a candidate for the i -th bit of the witness w .

The second difference is that even when the extractor is successful, we do not require that the extracted witness be perfectly correct. We may allow some small fraction of the extracted bits to be incorrect, as long as this will be negligible when the proof is used in any polynomial-time process. We formalize this by saying that for any (potentially adversarially generated) polynomial sized set of indices I , it must be the case that with all but negligible probability, all of the associated bits produced by the extractor will be correct with respect to some valid witness \hat{w} .

Thus, our definition requires an extractor which satisfies two properties. The first is that, for any PPT (and potentially adversarial) sampling algorithm S which chooses a set of indices I , with all but negligible probability over the choice of random tape r , the extractor E_r with oracle access to P^* will output a set of bits $\{w_i\}_{i \in I}$ that are consistent with some valid witness \hat{w} . Note that we allow S to choose these indices adaptively, after querying E on indices of its choice. This allows for the fact that in a larger application, which positions of the witness need to be extracted may depend on the values of the bits which have been seen so far.

The second property requires that the expected running time of E be within a polynomial factor of $\frac{1}{p(|y|)}$ where $p(|y|)$ is the success probability of the adversarial prover P^* . There is a slight issue here in that, even if the *expected* running time of $E(y, i)$ was guaranteed to be polynomial for each i , it might still be possible that for any choice of random tape r , an adversarial and adaptive sampling algorithm S could find at least one index that causes $E_r(y, i)$ to run in super-polynomial time. Thus we also require that the running time be independent of i ; this implies that the expected running time of E on any set of inputs (even those that are chosen adversarially and adaptively) will also be polynomially related to $\frac{1}{p(|y|)}$. For our application this will be particularly useful, as it implies that E can be converted into a circuit which implicitly describes the witness and whose expected size is polynomially related to the positive success probability of P^* (and therefore the expected size of E is polynomial whenever P^* has non-negligible success probability). (We will see more details in Section 4.)

Definition of Universal Argument of Quasi Knowledge (UAQK). We construct a universal argument $\langle P(\cdot, \cdot), V(\cdot) \rangle$ such that the efficient verification, completeness and computational soundness properties hold as usual, but the weak proof of knowledge property is replaced as follows.

Definition 1. A universal argument system $\langle P(\cdot, \cdot), V(\cdot) \rangle$ for the universal language L_U is a universal argument of quasi knowledge (UAQK) if there exists an algorithm E and a constant c such that for any polynomial-size circuit family $\{P_n^*\}_{n \in N}$, there exists a negligible function ν such that for any $y \in L_U \cap \{0, 1\}^n$, the following two properties hold.

1. If $\text{Prob}[\text{out}_V(\langle P_n^*, V(y) \rangle) = 1]$ is non-negligible, then for any polynomial-time sampling algorithm S ,

$$\text{Prob}_r[I \leftarrow S^{E_r^{P_n^*}(y, \cdot)}(y); \{w_i \leftarrow E_r^{P_n^*}(y, i)\}_{i \in I}] :$$

$$\exists \hat{w} = \hat{w}_1, \dots, \hat{w}_s, (\hat{w}, y) \in R_U \wedge (w_i = \hat{w}_i \ \forall i \in I)] \geq 1 - \nu(|y|).$$

where the probability is over the choice of the coins r used by E . (Note that the same coins are used for all $i \in I$).¹

2. The running time of $E_r^{P_n^*}(y, i)$ is independent of the choice of i , and if we let $p(|y|) = \text{Prob}[\text{out}_V(\langle P_n^*, V(y) \rangle) = 1] > 0$, then the expected running time of $E_r^{P_n^*}(y, \cdot)$ is $O(\frac{|y|^c}{p(|y|)})$, where again the expectation is over the choice of the coins r used by E .

Note that if L_U is a language with polynomial-size witnesses, then this property implies the standard proof of knowledge property. The standard proof of knowledge extractor will first run $\langle P_n^*, V(y) \rangle$ once: if V rejects, it will output \perp , otherwise it will choose a random string r to be used as randomness to run $E_r^{P_n^*}(y, i)$ for each bit i of the witness, and then output the result. This extractor will have success probability negligibly far from $p(|y|)$, and expected running time $O(p(|y|) \cdot \frac{|y|^c}{p(|y|)}) = O(|y|^c)$, which is clearly polynomial.

¹ Here we assume for simplicity that if a witness w is expanded by appending any sequence of zeros, the resulting value is still a valid witness, so that \hat{w}_i is always defined for all $i \in I$.

Note also that we could have given a definition that explicitly requires the extractor to run in expected polynomial time, essentially resembling the definitions of the standard proof of knowledge and weak proof of knowledge properties. However, we prefer the above formulation as it makes clear that we can differentiate between (item 1) runs on which the extractor inadvertently produces a bad witness (recall that when the witness is not polynomial-sized, it may not be possible to efficiently identify invalid witnesses) and (item 2) runs on which the extractor aborts (e.g., when the interaction with P_n^* produces an invalid proof).

2.2 CRHFs \Rightarrow Constant-Round UAQK

Our approach will be to construct a UAQK out of a UA with the weak proof of knowledge property. As mentioned in Section 1.1, there are two difficulties when using the weak proof of knowledge property: (1) we must somehow estimate a lower bound on the success probability of the prover in order to determine which extractor to use, and (2) the running time of the extractor may grow faster than what we would like as compared with the success probability of the prover (here we will use the UA construction of [4], which gives an extractor with running time $O((\frac{|y|}{\alpha(|y|)})^d)$ where $\alpha(|y|)$ is a lower bound on the success probability of P^*).

The first issue we will deal with within the design of our extractor; it essentially requires balancing the accuracy of the estimate with the need to compute it in expected polynomial time. In order to address the second issue, we will attempt to increase the success probability of the adversarial prover. Essentially, we will run many instances of that UA sequentially, and accept only if all instances are accepted by the verifier. Then, if the prover succeeds in all instances with probability p , we will argue that there must be at least one instance in which it succeeds with significantly higher probability. If we use this instance, then we can run the extractor with a higher lower bound, and thus obtain a more efficient extractor.

The resulting construction goes as follows. We first ask the prover to commit to a Merkle hash of its witness, and then we run the UA several times sequentially. In each UA, the prover proves both that the statement is true, and that the witness used is consistent with the given commitment, i.e. it proves weak knowledge of a witness and an authentication chain for each bit of the witness. (This will allow us to verify that parts of the witness are consistent with the initial commitment without having to extract the entire witness.)

The Actual UAQK. Our construction uses as a subprotocol a universal argument $\langle P_{UA}(\cdot, \cdot), V_{UA}(\cdot) \rangle$ satisfying the conditions described in Theorem 1 for the universal language L_U . Let $\ell = d + 2$ where d is the value of the constant defined in Theorem 1 for this UA. We construct a UAQK $\langle P_{new}(y, w), V_{new}(y) \rangle$ for language L_U as depicted in Fig. 1.

2.3 Proving the Quasi-Knowledge Property

Intuition. At a high level, the proof proceeds as follows: Let p be the probability that the adversarial prover P^* convinces V_{new} to accept. Then for $j = 1, \dots, \ell$,

Players: prover P_{new} , verifier V_{new} .

Common input: the statement $y \in L_U$, such that $|y| = n$.

Input of P_{new} : witness w for $y \in L_U$.

Tools: a family $\{\mathcal{H}_n\}$ of CRHFs, a statistically hiding trapdoor commitment scheme (**SHTGen**, **SHCom**, **SHVer**, **SHTCom**, **SHTDec**), a constant-round statistical ZKAoK $\langle P_{szk}(\cdot, \cdot), V_{szk}(\cdot) \rangle$ for NP, a constant-round WI universal argument $\langle P_{UA}(\cdot, \cdot), V_{UA}(\cdot) \rangle$ (as defined in Theorem 1) with parameter d .

- 1) $V_{new} \rightarrow P_{new}$: V_{new} picks $h \xleftarrow{R} \{\mathcal{H}_n\}$, computes $(\eta, t) \xleftarrow{R} \text{SHTGen}(1^n)$ and sends (h, η) to P_{new} .
- 2) $V_{new} \leftrightarrow P_{new}$: V_{new} runs P_{szk} to prove to P_{new} running V_{szk} knowledge of a trapdoor t for η .
- 3) $P_{new} \rightarrow V_{new}$: P_{new} encodes the witness w as follows: Let $s = |w|$ be the length of the original witness w , and let $w = w_1, \dots, w_s$ be the bitwise representation of w . Then P_{new} uses h to form a Merkle hash tree over all the bits of w treating each bit as a leaf node in the tree. Let $root$ be the root of this tree, and for bit w_i of witness w , let $auth_i$ be the appropriate authentication chain (i.e., all of the values on the path from w_i to $root$, and all of the siblings of those values). Next P_{new} computes $(C_s, \text{dec}_s) \leftarrow \text{SHCom}_\eta(s)$ and $(C_{root}, \text{dec}_{root}) \leftarrow \text{SHCom}_\eta(root)$. Finally, P_{new} forms the new witness $w' = s \circ \text{dec}_s \circ root \circ \text{dec}_{root} \circ w_1 \circ auth_1 \circ \dots \circ w_s \circ auth_s$. Now, let $y' = C_s \circ C_{root} \circ y$ and let L'_U be the language which accepts (y', w') iff $w' = s \circ \text{dec}_s \circ root \circ \text{dec}_{root} \circ w_1 \circ auth_1 \circ \dots \circ w_s \circ auth_s$ such that (1) $s < 2^{|y'|}$, (2) (s, dec_s) is a valid decommitment for C_s and $(root, \text{dec}_{root})$ is a valid decommitment for C_{root} with parameter η , (3) for all $i = 1 \dots s$, $auth_i$ is a valid authentication chain for w_i with respect to $root$ and hash function h , and (4) $w = w_1 \dots, w_s$ is such that $(y, w) \in L_U$. P_{new} sends (C_s, C_{root}) to the verifier.
- 4) $P_{new} \leftrightarrow V_{new}$:
For $j = 1, \dots, \ell = d + 2$, sequentially:
4.j) P_{new} and V_{new} run the universal argument $\langle P_{UA}(\cdot, \cdot), V_{UA}(\cdot) \rangle$ for this new language L'_U . P_{new} will run $P_{UA}(w', y')$ while V_{new} will run $V_{UA}(y')$, where $y' = C_s \circ C_{root} \circ y$.
 V_{new} outputs 1 iff all instances of $V_{UA}(y')$ accept.

Fig. 1. Universal Argument of Quasi Knowledge $\langle P_{new}(\cdot, \cdot), V_{new}(\cdot) \rangle$

let p_j be the probability that V_{new} accepts the j -th internal UA instance conditioned on the event that it accepted all previous instances. Then the key observation is that $p = \prod_{j=1}^\ell p_j$, so we are guaranteed that for some j^* , $p_{j^*} \geq p^{1/\ell}$. Now, if we could estimate p_{j^*} , and identify a UA instance where P^* succeeds with probability roughly p_{j^*} , then we could run the UA extractor with this lower bound, and obtain an extractor with success probability roughly $\frac{p^{1/\ell}}{q(|y|)}$ (here q is the polynomial referred to as q^* in the discussion in Theorem 1), and running time roughly $O((\frac{|y|}{p_{j^*}})^d) = O(\frac{|y|^d}{p^{d/\ell}})$. However, we need an extractor with overwhelming success probability, so we will run this extractor approximately

$\frac{q(|y|)}{p^{1/\ell}}$ times, to ensure that at least one run will produce bits of a valid implicit witness.² The final result will have success probability nearly 1, and running time roughly $O(\frac{|y|^d}{p^{d/\ell}} \cdot \frac{q(|y|)}{p^{1/\ell}}) = O(\frac{|y|^d q(|y|)}{p^{(d+1)/\ell}}) = O(\frac{|y|^d q(|y|)}{p})$ (the last equality follows from our choice of ℓ).³

The main challenge in this process is finding a UA instance where P^* has success probability roughly $p^{1/\ell}$, and estimating this probability. Essentially, for each j , we want to find a starting state where P^* is about to begin the j -th UA, and where P^* 's success probability in that proof is roughly p_j ; then we need to identify and take the best of these states (i.e., p_{j^*}). We do this as follows: First, for each j , we record many states for P^* where P^* has successfully completed the first $j - 1$ UAs and has a reasonable chance of completing the next UA. In the full version we show that with overwhelming probability, one of these states will be such that P^* has probability at least $p_j/2$ of successfully completing the next UA, and at the same time that the time required to collect all these states is at most $O(\text{poly}(|y|)/p)$. Next, we attempt to identify one such state, and to estimate the corresponding success probability. We do this by running P^* from each state many times, and counting how long it takes for P^* to complete $|y|$ proofs starting from that state. We interleave the counting for all of the states corresponding to the j -th proof to ensure that we can stop as soon as one has resulted in $|y|$ successful proofs. (This allows us to avoid running for too long just because one candidate state was bad.) Finally, we consider the best states for $j = 1, \dots, \ell$, select the one with the highest estimated success probability, and use it as described above.

Our Extractor. To summarize, our extractor works as follows:

1. For each $j = 1, \dots, \ell$:
 - (a) Collect $n = |y|$ candidate states, where P^* has successfully completed the first $j - 1$ proofs and has a reasonable chance of successfully completing the next one.
 - (b) Repeatedly run the next UA from all the above n states, and identify the state beststate_j that is the first one that reaches n accepting executions of the next $\langle P_{UA}(\cdot, \cdot), V_{UA}(\cdot) \rangle$. Let m_j be the number of such executions from that state.

² There is some subtlety here in that we must guarantee that we can always recognize which set of extracted bits is the correct one (the one which is consistent with some valid witness). To do this we make use of the prover's initial commitment to the witness - if the extracted bits are consistent with the initial commitment, then we assume that they are the correct ones.

³ In fact this is slightly inaccurate, as we also need to guarantee, even when we run this extractor many times and boost P^* 's success probability a bit more, none of these times takes too long. We do this by estimating a reasonable upper bound for the running time of E based on our estimate of P^* 's success probability, and stopping E early if it runs more than this number of steps. As a result, we have to run E a few more times, and we set $\ell = d + 2$.

2. Given the above m_j for $j = 1, \dots, \ell$, let \hat{j} be the index of the UA where $m_{\hat{j}}$ is minimal. $\frac{n}{2m_{\hat{j}}}$ is an estimate of a lower bound on the adversarial prover's success probability $p_{\hat{j}}$ in state $beststate_{\hat{j}}$.
3. Run approximately $q(|y|)/p_{\hat{j}}$ instances of the UA extractor. Return the result that agrees with the initial commitment sent by the prover.

A more formal description of the extractor and a more detailed analysis can be found in the full version of the paper. We also note that for technical reasons, we also need a statistically hiding trapdoor commitment scheme and a statistical zero-knowledge argument of knowledge of the trapdoor, which help us to prove the witness indistinguishability of our construction.

Theorem 2. *Under the assumption that a family of collision-resistant hash functions exists, then the protocol depicted in Fig. 1 is a constant-round witness indistinguishable UAQK.*

3 Secure Database Commitments

We consider the notion of a secure database commitment such that each element x appears at most once. As in [14], we will consider a formulation that captures both sets and databases. For a set S , we can define $Db[x]$ to be 1 if x appears in the set and \perp if it does not. More generally, for a database of pairs (x, y) , we define $Db[x]$ to be y if x appears in the set, and y otherwise. Since the difference between the two primitives is almost cosmetic, we will use the terms of sets and database interchangeably. We will assume that each element x belongs to $\{0, 1\}^L$ and L is polynomial in the security parameter k . Thus, by the requirements above, this means that the database contains at most 2^L entries. We will make no other requirement on the size of the database.

The functionality in question is \mathcal{F}_{DbCom} and is the natural extension of the standard commitment functionality, but in \mathcal{F}_{DbCom} we must hide the size of the committed message and we also have to deal with queries and responses. (These responses must hide all other information about the database, thus there are also similarities with the zero-knowledge functionality.)

Ideal Functionality \mathcal{F}_{DbCom} . The database commitment functionality consists of two phases: one in which a prover commits to a database, and a second in which the prover answers queries about that database. Here we will generalize this definition to say that the prover can commit to any database for which he knows an implicit representation. In particular, we will allow the prover to commit to a circuit which evaluates the desired database: it takes as input a string x , and returns $Db[x]$ (which will be \perp if x is not in the database).

The formal specification is given in Fig. 2. For simplicity we assume that all queries made by the honest verifier V are fixed in advance. The definition can be generalized to adaptive queries, where the next query depends on the output of the previous ones. Our construction will satisfy this stronger definition as well.

Remark. Note that our definition of \mathcal{F}_{DbCom} does not exclude the possibility that a player is able to commit to a very large (e.g. superpolynomial sized)

set, for which he only knows some implicit representation. However, this would not violate any intuitive security requirement of secure database commitments since the critical requirement is that the prover is committed to some database (whose size is not a priori upperbounded by any fixed polynomial) at the end of the commitment phase, and that it must answer correctly according to this database during the query phase. We will show a construction in Section 4 that will require that the honest sender knows an explicit representation of the database he is committing to (i.e., the honest sender runs on input the sequence of elements it wants to commit to) (Note that an explicit representation can always be efficiently converted into an implicit representation \mathcal{C}_{Db} .) Obtaining a scheme where a real-world honest prover is allowed to have as input a polynomial-sized implicit representation that corresponds to a super-polynomial number of elements is also an interesting direction, and we defer it to future research.

Defining Security. In the ideal execution, when initiated with input, parties interact with \mathcal{F}_{DbCom} . The adversary Sim has control of a party and thus can deviate from the prescribed computation, while the other player H_P follows the prescribed computation. We denote by $\text{IDEAL}_{H_P, \text{Sim}}^{\mathcal{F}_{DbCom}}(k, z)$ the output of Sim on input an auxiliary input z , security parameter k and uniform randomness. Moreover we denote by $\{\text{IDEAL}_{H_P, \text{Sim}}^{\mathcal{F}_{DbCom}}(k, z)\}$ the corresponding ensemble of distributions, with $k \in \mathcal{N}$ and $z \in \{0, 1\}^*$.

In the real execution parties run a given protocol π . Let H_P be the honest party and \mathcal{A} be the adversary, which has control of the other party. We denote by $\text{EXEC}_{H_P, \mathcal{A}}^\pi(k, z)$ the output of \mathcal{A} on input an auxiliary input z , security parameter k and uniform randomness. Moreover we denote by $\{\text{EXEC}_{H_P, \mathcal{A}}^\pi(k, z)\}$ the corresponding ensemble of distributions, with $k \in \mathcal{N}$ and $z \in \{0, 1\}^*$.

Definition 2. A protocol π securely realizes the functionality \mathcal{F}_{DbCom} if for any real-world adversary \mathcal{A} there exists an ideal-world adversary Sim such that for every sufficiently large k , and for every $z \in \{0, 1\}^*$, $\{\text{EXEC}_{H_P, \mathcal{A}}^\pi(k, z)\}$ and $\{\text{IDEAL}_{H_P, \text{Sim}}^{\mathcal{F}_{DbCom}}(k, z)\}$ are computationally indistinguishable.

4 Constant-Round Secure Database Commitments

In this section we present a constant-round protocol that securely realizes the \mathcal{F}_{DbCom} functionality. Our construction uses a constant-round zero-knowledge argument of knowledge (ZKAoK) for all NP, a constant-round WI universal argument of quasi knowledge, a 2-round trapdoor commitment scheme, and a zero-knowledge set scheme with two additional properties. As all of these ingredients can be instantiated through CRHFs, this gives a secure realization of \mathcal{F}_{DbCom} based only on CRHFs. An additional feature of our protocol is that the amortized round complexity of a proof is optimal (i.e., proofs are actually non-interactive).

Zero-Knowledge Sets. We start by reviewing a major building block for our construction: zero-knowledge sets. At a high level, a non-interactive

FUNCTIONALITY \mathcal{F}_{DbCom}

Players: Prover P , Verifier V .

Input of P : circuit \mathcal{C}_{Db} . **Input^a of V :** x_1, \dots, x_m with $m = \text{poly}(k)$.

Computation of \mathcal{F}_{DbCom} :

1. Upon receiving $(\text{Commit}, \mathcal{C}_{Db})$ from P , if $(\text{Commit}, \mathcal{C}_{Db})$ was already received then ignore, otherwise record (\mathcal{C}_{Db}) , send (Receipt) to V .
2. Upon receiving (Query, x) from V , if $(\text{Commit}, \mathcal{C}_{Db})$ was previously received from P , then send (Query, x) to P , otherwise ignore.
3. Upon receiving $(\text{Open}, 1, x)$ from P , if (Query, x) was not previously sent to P then ignore, otherwise evaluate the circuit \mathcal{C}_{Db} on input x to obtain output $Db[x]$, and send $(\text{Open}, x, Db[x])$ to V .
4. Upon receiving $(\text{Open}, 0, x)$ from P , if (Query, x) was not previously sent to P then ignore, otherwise send $(\text{Open-Abort}, x)$ to V .
5. Upon receiving (Halt) from V , if the same message was previously received from V then ignore, otherwise send (Halt) to P .

Computation of P :

1. Upon activation, send $(\text{Commit}, \mathcal{C}_{Db})$ to \mathcal{F}_{DbCom} .
2. Upon receiving (Query, x) from \mathcal{F}_{DbCom} , send $(\text{Open}, 1, x)$ to \mathcal{F}_{DbCom} .
3. Upon receiving (Halt) from \mathcal{F}_{DbCom} go to Output.

Computation of V :

1. Upon receiving (Receipt) from \mathcal{F}_{DbCom} send (Query, x_1) to \mathcal{F}_{DbCom} .
2. Upon receiving (Open, x_i, y_i) from \mathcal{F}_{DbCom} , if $0 < i < m$ then send (Query, x_{i+1}) to \mathcal{F}_{DbCom} .
3. Upon receiving (Open, x_m, y_m) or $(\text{Open-Abort}, x)$ from \mathcal{F}_{DbCom} , send (Halt) to \mathcal{F}_{DbCom} and go to Output.

Output:

P : Output (x_1, \dots, x_t) where $t = \text{poly}(k)$ and x_i for $0 < i \leq t$ is such that (Query, x_i) is the i -th message received from \mathcal{F}_{DbCom} .

V : Output $((x_1, y_1), \dots, (x_{m'}, y_{m'}))$, where each x_i for $0 < i \leq m'$ was part of a message (Query, x_i) sent to \mathcal{F}_{DbCom} , and y_i for $0 < i \leq m'$ was part of a message (Open, x_i, y_i) received from \mathcal{F}_{DbCom} while y_i is the empty string in case $(\text{Open-Abort}, x_i)$ has been received from \mathcal{F}_{DbCom} .

^a We stress that inputs can also be computed adaptively.

Fig. 2. Ideal computation of functionality \mathcal{F}_{DbCom}

zero-knowledge set scheme is composed of four algorithms as follows: an algorithm $ZKS\text{Setup}$, which generates some parameters, an algorithm $ZKS\text{Com}$, which commits to a database in a size independent way, an algorithm $ZKS\text{Prove}$, which allows the owner of the database to prove that a particular query response was consistent with the committed database, and an algorithm $ZKS\text{Verify}$ for verifying such proofs. We will also use “ZKS proof system” to refer to the interaction between $ZKS\text{Prove}$ and $ZKS\text{Verify}$.

We will use non-interactive zero-knowledge sets as a building block in our construction of secure database commitment. In the full version of this work we give

a definition for zero-knowledge sets which is similar to that in [7]. We also define a somewhat stronger hiding property (which we call *special zero knowledge*), in which zero knowledge holds for all valid parameters output by the simulator set-up algorithm, and the simulated commitment is an honest commitment to an empty database. In the full version we will discuss how to achieve these properties based on CRHFs. We finally stress that even though non-interactive zero-knowledge sets have been defined in common reference string model, we will use them in the standard model by using interaction.

Intuition for Our Protocol. The basic idea behind this construction is fairly straightforward: we give a concise commitment to the database (using the ZKS commitment algorithm), and use a universal argument of quasi knowledge to prove knowledge of a valid opening. Then we can use the ZKS proof system to respond to queries. However, we need a few additional properties from the ZKS proof system to make this work. The issue is that when we are dealing with a corrupt prover, we need to be able to extract a circuit representation of the committed database from the UAQK (by which we mean a circuit which on input x returns $Db[x]$). On the other hand, the UAQK only guarantees an extractor which, given i , returns the i th bit of a valid witness. So we augment the definition of ZKS with an additional property which will allow us to construct an implicit database representation from a special witness string. In particular, we need to guarantee that the witness will be in a format such that we can efficiently extract $Db[x]$ for any x given only the UAQK extractor which allows queries to individual bits of the witness. Furthermore, in order to argue that the responses in the query phase must be consistent with the extracted database, we also need to be able to efficiently extract a ZKS proof for each x . This must still be done by just querying the circuit representation. Therefore the mere use of UAQK is not sufficient to securely realize \mathcal{F}_{DbCom} , and we will provide the needed additional ingredients.

Definition 3. A ZKS proof system ($ZKS\text{Setup}$, $ZKS\text{Com}$, $ZKS\text{Prove}$, $ZKS\text{Verify}$) allows local witnesses if there exist additional polynomial time deterministic algorithms FormWitness , TMVer , Eval , PfGen such that the following properties hold:

Witness verifiability. For any sufficiently large k , for all polynomial sized Db , and random strings r ,

$$\Pr[\text{ZKSPAR} \leftarrow \text{ZKS}\text{Setup}(1^k); zks \leftarrow \text{ZKS}\text{Com}(\text{ZKSPAR}, Db, r);$$

$$w \leftarrow \text{FormWitness}(\text{ZKSPAR}, Db, r) : \text{TMVer}(\text{ZKSPAR}, zks, w) = 1] = 1.$$

Local evaluation. There exists a polynomial q such that for any sufficiently large k , for any $x \in \{0, 1\}^k$ and $w \in \{0, 1\}^{2^k}$, $\text{Eval}(x, w)$ only accesses $q(k)$ bits of w and runs in time polynomial in k .

Local verification. There exists a polynomial q' such that for any sufficiently large k , $x \in \{0, 1\}^k$ and $w \in \{0, 1\}^{2^k}$, $\text{PfGen}(x, w)$ only accesses $q'(k)$ bits of w and runs in time polynomial in k . Moreover for any $x \in \{0, 1\}^k$ and $w \in \{0, 1\}^{2^k}$ and polynomial sized zks ,

$$\Pr[\text{ZKSPAR} \leftarrow \text{ZKS}\text{Setup}(1^k) : \text{TMVer}(\text{ZKSPAR}, zks, w) = 1 \wedge$$

$$\wedge \text{ZKS}\text{Verify}(\text{ZKSPAR}, zks, x, \text{Eval}(x, w), \text{PfGen}(x, w)) = 0] = 0.$$

The intuition of the above definition is that there must exist a procedure **FormWitness** that transforms the witness while still preserving the content of the database (indeed this is verifiable through **TMVer**). Moreover one can extract the membership or non membership of an element and the corresponding proof by accessing only a polynomial number of bits of this witness, according to the algorithms **Eval** and **PfGen**. The reason why we allow w to be of superpolynomial size is that the adversary can bypass the procedure **FormWitness** and produce a circuit that implicitly represent a superpolynomial sized witness. The interesting property of our definition is that even in this case, **Eval** and **PfGen** will be efficient. We now show that standard constructions of ZKS ([14,7,5]) have this property.

Previous ZKS Schemes Allow Local Witnesses. We will use ideas from the ZKS construction of [14,7]. We summarize only the main properties we need.

The constructions work by building a tree. For each x in the database, we parse x as bits b_1, \dots, b_L . Then at position b_1, \dots, b_L in the tree (where $b_i = 0$ denotes the left branch and $b_i = 1$ denotes the right branch at height i), we store a commitment v_x to the corresponding y (all commitments here use a special commitment scheme). We construct the tree from the bottom up. For every ancestor $x' = b_1, \dots, b_i$ of such an x , we compute a commitment to a hash of the two children: $v_{x'} = \text{Com}(h(v_{x'||0}, v_{x'||1}))$. If one of the children has not yet been specified (it is the root of a subtree with no leaves in the database), we set that child node to $\text{Com}(\perp)$.

Then a proof for value $(x, y) \in Db$ can be formed by producing all sibling and ancestor nodes and openings for all ancestor commitments. A proof for value $x \notin Db$ is more complex, but it can be formed by first finding the root of the largest empty subtree containing x . The value of this root node (call it x_\perp) should be \perp . Then the proof will include all sibling and ancestor nodes of x_\perp , special openings of all ancestor commitments, and an additional value which can be constructed given x and the randomness used to form the commitment at x_\perp .

More formally we have that for every empty subtree, there exists a polynomial sized string $info$, such that for all leaves x in this subtree, $\text{PfGen}(x, info)$ produces output identical to **ZKSProve**. It is also possible to efficiently verify that $info$ is formed correctly for a given subtree.

We now sketch the necessary algorithms as defined in Definition 3:

FormWitness : will first form the ZKS tree as described above. Then we will transform this into a single linear witness. For each nonempty, non- \perp node x in the tree we will form an entry consisting of the value of v_x , the values of its children, the opening of the commitment v_x , and the positions of the children nodes in the tree. For each \perp node, we form an entry with the commitment v_\perp and the extra $info$ necessary to open the commitment. The witness will begin with the bit position of the entry corresponding to the root commitment.

TMVer: will traverse the entire tree, and verify all commitments and hashes. This algorithm can easily be described by a polynomial sized TM.

Eval(x): will follow the path through the tree corresponding to x , beginning with the root and continuing until it reaches either \perp or a leaf node with value y , and return the result.

PfGen: will follow the path through the tree corresponding to x until it reaches an empty subtree or a leaf. If it is a leaf, it will return all of the ancestor and sibling nodes and the openings, as described above. If it is an empty subtree, it reads the accompanying value $info$, runs $\text{PfGen}(x, info) \rightarrow \pi$ and returns π and all the ancestor and sibling nodes and openings. In both cases the output for an honestly generated witness will be identical to the output of **ZKSProve**.

PfVer: will verify each of the hashes and commitments in the chain. If the final node is \perp , it will verify the accompanying proof π .

Our Construction. We will use a constant-round zero-knowledge argument of knowledge (ZKAoK) $\langle \mathcal{P}(\cdot, \cdot), \mathcal{V}(\cdot) \rangle$, a constant-round WI UAQK $\langle \text{UAP}(\cdot, \cdot), \text{UAV}(\cdot) \rangle$, a 2-round trapdoor commitment scheme $(\text{TGen}, \text{Com}, \text{TCom}, \text{TDec}, \text{Ver})$, and a special ZKS $(\text{ZKSSetup}, \text{ZKSCom}, \text{ZKSProve}, \text{ZKSVerify})$ which allows local verification with zero knowledge simulator $(\text{ZKSSimSetup}, \text{ZKSSimCom}, \text{ZKSSimProve})$. A description of our scheme is found in Fig. 3.

Security Parameter: k .

Input to P : $Db = ((x_1, y_1), \dots, (x_s, y_s))$ where $s = \text{poly}(k)$ and $x_i \in \{0, 1\}^k$ for $0 < i \leq s$.

Input to V : x'_1, \dots, x'_m , where $m = \text{poly}(k)$ and $x'_i \in \{0, 1\}^k$ for $0 < i \leq m$.

Commitment Phase:

1. $V \rightarrow P$: set $(\text{ZKSPAR}, \text{ZKSTRAP}) \leftarrow \text{ZKSSimSetup}(1^k)$, $(\text{crs}, \text{aux}) \leftarrow \text{TGen}(1^k)$ and send $(\text{ZKSPAR}, \text{crs})$ to P .
2. $V \leftrightarrow P$: V proves knowledge of **ZKSTRAP**, aux (i.e., V and P run $\mathcal{P}((\text{ZKSPAR}, \text{crs}), (\text{ZKSTRAP}, \text{aux}))$ and $\mathcal{V}((\text{ZKSPAR}, \text{crs}))$ respectively). If P rejects the proof, then it aborts.
3. $P \rightarrow V$: pick $r \in \{0, 1\}^k$, set $(c, \text{dec}) = \text{Com}(\text{crs}, zks) = \text{ZKSCom}(\text{ZKSPAR}, Db, r)$ and send c to V .
4. $P \leftrightarrow V$: P execute **FormWitness**(ZKSPAR, Db, r) to generate witness w , and then execute the code of **UAP** on input $c||\text{ZKSPAR}||\text{crs}$ with witness $zks||\text{dec}||w$. V executes the code of **UAV** on input $c||\text{ZKSPAR}||\text{crs}$. If **UAV** rejects, V aborts. **UAP** proves quasi knowledge of a witness of the form $zks||\text{dec}||w$, where zks, dec is a valid opening for commitment c under crs , and w is such that **TMVer**(ZKSPAR, zks, w) accepts.
5. $P \rightarrow V$: open the commitment c by sending zks, dec to V . If the opening is not correct, V aborts.

Query/Answer Phase:

For $i = 1, \dots, m$ do:

6. $V \rightarrow P$: send x'_i to P .
7. $P \rightarrow V$: send $(y'_i = Db[x'_i], \pi = \text{ZKSProve}(\text{ZKSPAR}, x'_i, Db[x'_i], Db, r))$ to V . V outputs (x'_i, y'_i) if $\text{ZKSVerify}(\text{ZKSPAR}, zks, x'_i, y'_i, \pi) = 1$ and (x'_i, \perp) otherwise.

Fig. 3. Our scheme for secure database commitments

Round Optimality. We stress that simply by inspection one can observe that the query/proof phase is non-interactive (optimal).

Security. The protocol depicted in Fig. 3 securely realizes the \mathcal{F}_{DbCom} functionality, i.e., it is a constant-round protocol for secure database commitment. For lack of space the proof is presented in the full version.

Theorem 3. *If $\langle \mathcal{P}(\cdot, \cdot), \mathcal{V}(\cdot) \rangle$ is a ZKAoK, $\langle \text{UAP}(\cdot, \cdot), \text{UAV}(\cdot) \rangle$ is a WI UAQK, $(\text{TGen}, \text{Com}, \text{TCom}, \text{TDec}, \text{Ver})$ is a 2-round trapdoor commitment scheme, and $(\text{ZKSSetup}, \text{ZKSCom}, \text{ZKSProve}, \text{ZKSVerify})$ is a special zero-knowledge set scheme which allows local witnesses, then the protocol depicted in Fig. 3 securely realizes the \mathcal{F}_{DbCom} functionality.*

Corollary 1. *Under the assumption that there exists a family of CRHFs, then there exists an efficient protocol which securely realizes the \mathcal{F}_{DbCom} functionality.* This follows from the fact that all of the primitives mentioned in Theorem 3 can be realized based on CRHFs.

References

1. Ateniese, G., De Cristofaro, E., Tsudik, G.: (If) Size Matters: Size-Hiding Private Set Intersection. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 156–173. Springer, Heidelberg (2011)
2. Barak, B.: How to Go Beyond the Black-Box Simulation Barrier. In: FOCS 2001, pp. 106–115. IEEE Computer Society Press (2001)
3. Barak, B.: Non-Black-Box Techniques in Cryptography, Ph.D. Thesis. Weizmann Institute of Science (2004)
4. Barak, B., Goldreich, O.: Universal Arguments and Their Applications. In: CCC 2002. IEEE Computer Society Press (2002)
5. Catalano, D., Dodis, Y., Visconti, I.: Mercurial Commitments: Minimal Assumptions and Efficient Constructions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 120–144. Springer, Heidelberg (2006)
6. Catalano, D., Fiore, D., Messina, M.: Zero-Knowledge Sets with Short Proofs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 433–450. Springer, Heidelberg (2008)
7. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial Commitments with Applications to Zero-Knowledge Sets. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 422–439. Springer, Heidelberg (2005)
8. Gennaro, R., Micali, S.: Independent Zero-Knowledge Sets. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 34–45. Springer, Heidelberg (2006)
9. Goldreich, O.: Foundations of Cryptography - Volume II - Basic Applications. Cambridge Press (2004)
10. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game - A Completeness Theorem for Protocols with Honest Majority. In: STOC 1987, pp. 218–229 (1987)
11. Hada, S., Tanaka, T.: On the Existence of 3-Round Zero-Knowledge Protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998)

12. Ishai, Y., Paskin, A.: Evaluating Branching Programs on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
13. Libert, B., Yung, M.: Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 499–517. Springer, Heidelberg (2010)
14. Micali, S., Rabin, M., Kilian, J.: Zero-knowledge sets. In: FOCS 2003, pp. 80–91 (2003)
15. Prabhakaran, M., Xue, R.: Statistically Hiding Sets. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 100–116. Springer, Heidelberg (2009)

Succinct Arguments from Multi-prover Interactive Proofs and Their Efficiency Benefits

Nir Bitansky^{1,*} and Alessandro Chiesa²

¹ TAU

nirbitan@tau.ac.il

² MIT

alexch@csail.mit.edu

Abstract. *Succinct arguments of knowledge* are computationally-sound proofs of knowledge for NP where the verifier’s running time is independent of the time complexity of the NP nondeterministic machine for the considered language.

Existing succinct argument constructions are, typically, based on techniques that combine cryptographic hashing and probabilistically-checkable proofs (PCPs), and thus, in light of today’s state-of-the-art PCP technology, are quite inefficient: either one uses long PCP proofs with lots of redundancy to make the verifier fast but at the cost of making the prover slow, or one uses short PCP proofs to make the prover fast but at the cost of making the verifier slow.

To obtain better efficiency, we propose to investigate the alternative approach of constructing succinct arguments based on multi-prover interactive proofs (MIPs) and stronger cryptographic techniques:

(1) We construct a one-round succinct MIP of knowledge protocol where (i) each prover is highly efficient in terms of time AND space, and ALSO (ii) the verifier is highly efficient.

(2) We show how to transform any one round MIP protocol to a succinct four-message argument (with a single prover), while preserving the time and space efficiency of the original MIP protocol.

As a main tool for this transformation, we construct a *succinct multi-function commitment* that (a) allows the sender to commit to a vector of functions in time and space complexity that are essentially the same as those needed for a single evaluation of the functions, and (b) ensures that the receiver’s running time is essentially independent of the function. The scheme is based on fully-homomorphic encryption (and no additional assumptions are needed for our succinct argument).

(3) In addition, we revisit the problem of *non-interactive* succinct arguments of knowledge (SNARKs), where known impossibilities rule out solutions based on black-box reductions to standard assumptions. We formulate a natural (though non-standard) variant of homomorphic encryption that has a *homomorphism-extraction property*. We then show that this primitive essentially allows to “squash” our interactive protocol, while again preserving time and space efficiency. We further show that this variant is, in fact, implied by the existence of SNARKs.

* This research was done while visiting Boston University and funded by the Check Point Institute for Information Security, by Marie Curie grant PIRG03-GA-2008-230640, an ISF grant 0603805843, and the Fulbright program.

1 Introduction

Interactive Proofs & Succinctness. Interactive proofs [GMR89] are central to modern cryptography and complexity theory. One extensively-studied aspect of interactive proofs is their expressibility; this study culminated with the celebrated result that $\text{IP} = \text{PSPACE}$ [Sha92]. Another aspect, which is the focus of this work, is that proofs for NP-statements can potentially be verified much faster than by directly checking an NP witness.

Unfortunately, in interactive proofs with statistical soundness, any non-trivial savings in verification time is unlikely (see, e.g., [BHZ87, GH98, GVW02, Wee05]). However, if we settle for proof systems with only *computational* soundness (also known as argument systems [BCC88]), then significant savings can be made.

Indeed, using collision-resistant hash functions (CRHs) and probabilistically-checkable proofs (PCPs) [BFLS91], Kilian [Kil92] showed a four-message interactive argument where membership of an instance y in an NP language L can be verified in time that is bounded by $p(k, |y|, \log t)$, where t is the time to evaluate the NP verification relation for L on input y and a valid witness, p is a fixed polynomial independent of L , and k is a security parameter. Following tradition, we call such argument systems *succinct*.

A natural strengthening of computational soundness is (computational) *proof of knowledge*: it requires that, when the verifier is convinced by an efficient prover, not only can we conclude that a valid witness for the theorem *exists*, but also that such a witness can be *extracted* efficiently from the prover. Proof of knowledge is a natural property (satisfied by most proof system constructions, including the aforementioned one of Kilian [BG08]) that is very useful in many applications of succinct arguments.

A special case of succinct arguments that has received a lot of attention is the one of succinct *non-interactive* arguments of knowledge (SNARKs). Indeed, SNARKs are known for their powerful applications including: non-interactive delegation of computation, succinct non-interactive secure computation, extractable cryptographic primitives [BCCT11], and constructions of proof-carrying data [CT10, BCCT12].

1.1 Existing Succinct Arguments and Their Efficiency

Kilian’s four-message succinct argument of knowledge works as follows: the prover first uses a Merkle hash tree to bind itself to a polynomial-size PCP oracle for the statement to be proven, and then answers the PCP verifier’s queries while demonstrating consistency with the previous Merkle tree commitment.

Most SNARK constructions [Mic00, DCL08, BCCT11, DFH11, GLR11] are based on techniques for “squashing” Kilian’s protocol into a non-interactive one, and hence are also based on the “commit to a PCP oracle and then reveal” paradigm.¹

Room for Improvement. The current state of affairs is unsatisfying in two respects: on the one hand, from an “understanding perspective” it is unsatisfying that the only

¹ An exception are the SNARKs constructed in [BCCT12], which rely on recursive composition and “bootstrapping” of SNARKs with an expensive offline phase [Gro10, Lip11, GGPR12]; indeed, these SNARKs do not invoke the PCP theorem. Unfortunately the techniques of [BCCT12] do not seem to extend to the interactive setting.

way we know how to build succinct arguments (at least from standard assumptions) is through Kilian-type protocols; on the other hand, from a practical perspective, this lack of alternative constructions is forcing us to use PCPs, and thus compromise on efficiency — indeed, the concrete efficiency of PCPs is currently poorly understood (though recent progress was shown in [BSCGT12b, BSCGT12a]).

Thus, there is room to consider alternative succinct argument constructions that are potentially more efficient. Moreover, even for the weaker (but still very desirable) goal of delegating deterministic polynomial-time computations (rather than NP computations), we *also* do not have solutions with satisfying efficiency. Any progress on the practicality of succinct arguments will have direct implications for the practicality of delegation protocols (which can naturally be constructed based on succinct arguments).

In light of the above discussion, we consider the following questions:

- *Can we obtain succinct arguments via a construction not of the Kilian type?*
- *Can we avoid the inefficiencies of PCPs?*

The question of whether succinct arguments can be built by using tools that are “lighter” than polynomial-size PCPs (perhaps at the expense of relying on stronger cryptographic primitives) was raised by Ishai et al. [IKO07]. Specifically, Ishai et al. showed how to use Hadamard-based PCPs (together with additively-homomorphic encryption) to obtain simpler but only “semi-succinct” arguments (i.e., arguments where only the prover-to-verifier communication is succinct). We follow the same path and seek techniques for obtaining “fully-succinct” arguments via constructions that are simpler, avoid the use of polynomial-size PCPs, and are potentially more efficient.

Note that a potential obstruction to achieving our goal is that Rothblum and Vadhan [RV09] proved that PCPs are in a certain sense inherent to succinct argument constructions: they showed that any succinct argument (even if interactive) can be transformed into a PCP, as long as its security is established via a black-box reduction to standard cryptographic assumptions. So perhaps the inefficiencies of PCPs cannot be avoided.

The transformation of Rothblum and Vadhan, however, incurs significant overhead in the general case. Thus, it is *still* possible for there to exist a succinct argument construction that is more efficient than any construction directly relying on PCPs (e.g., a Kilian-type one); this holds *even* if such a construction induces a corresponding PCP. Looking ahead, one of the results of this paper is that this is indeed the case.

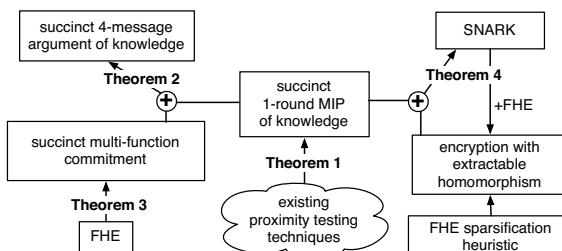


Fig. 1. Summary of our results

2 Summary of Our Results

At high-level, we show how PCPs in succinct arguments can be replaced by *multi-prover interactive proofs* (MIPs), when combined with the proper cryptographic tools, thus achieving new constructions with several efficiency benefits. Specifically (and as summarized in Figure 1):

We revisit the MIP model and prove that, using existing proximity testing techniques:

Theorem 1: “There is a one-round succinct MIP of knowledge where, to prove correctness of a t -time s -space computation, every prover runs in $t \cdot \text{polylog}(t)$ time and $s \cdot \text{polylog}(t)$ space, and the verifier runs in $\text{polylog}(t)$ time.”

Our construction does not hide any large constants and is quite simple and efficient. We then proceed to the task of trying to use our first theorem to construct succinct arguments (with a single prover) that are potentially more efficient than those constructed via PCP-based techniques. We show that:

Theorem 2: “Assuming the existence of fully-homomorphic encryption, there exists a four-message succinct argument of knowledge with the same efficiency as our MIP protocol, up to polynomial factors in the security parameter.”

We stress that succinct arguments with the time and space efficiency as in our construction are *not* known to be achievable via Kilian-type constructions that use PCPs.

The main tool in the above theorem is a *succinct multi-function commitment*, which enables a sender to commit to a vector of functions in time and space complexity that are essentially the same as those needed for a single evaluation of the functions, and where the receiver’s running time is essentially independent of the function.

Theorem 3: “Assuming the existence of fully-homomorphic encryption, there is a succinct multi-function commitment.”

In addition, we explore methods to construct SNARKs based on MIPs. Here, known impossibilities rule out solutions based on black-box reductions to standard assumptions [GW11]. We formulate and study a natural (but non-standard) variant of homomorphic encryption that we call *encryption with extractable homomorphism*, and suggest a candidate construction for this primitive. We show that:

Theorem 4: “Assuming the existence of fully-homomorphic encryption with extractable homomorphism, there exists a SNARK (with the same efficiency as our MIP protocol, up to polynomial factors in the security parameter). Furthermore, such encryption schemes are implied by the existence of SNARKS.”

In the following sections, we discuss and explain in somewhat more detail our results. For technical details, see the full version of this paper [BC12].

3 A Simple 1-Round Succinct MIP of Knowledge

A beautiful proof model that has not played so far a major role in the development of succinct arguments is that of *multi-prover* interactive proofs (or MIP protocols), originally introduced by Ben-Or et al. [BOGKW88]. In this model, a verifier conducts a simultaneous interactive protocol with several provers that, crucially, are assumed to be unable to communicate during the protocol.

Because MIPs have not been studied with an eye towards concrete efficiency (e.g., the construction in [BFL90] does not seem so practical), we revisit MIPs and show that, despite the fact that PCPs can be used to construct MIPs and vice versa [TS96], our present ability to exhibit practical constructions for the two is vastly different. Specifically, when studying the problem of making PCPs very efficient, we are faced with two difficult challenges:

- **Proof length vs soundness tradeoff.** We know how to construct either PCPs with great soundness but proofs that are long [RS97, MR08] (and thus with fast verifiers but slow provers), or PCPs with short proofs but soundness that is low [BSS08, BSGH⁺05] (and thus with not-as-slow provers but quite slow verifiers — due to the large number of repetitions needed to achieve, say, constant soundness). Obtaining a PCP that *simultaneously* has short proof length and great soundness is an exciting open problem (though [BSCGT12b] show recent progress).
- **Prover high space complexity.** We know of PCPs where the prover runs in $\tilde{O}(t)$ time but such running time is achieved via the use of FFT-like methods [BSCGT12b], which unfortunately demand $\Omega(t)$ space — essentially, the prover is asked to evaluate a $\Omega(t)$ -degree polynomial over a domain of $\Omega(t)$ size. For large t , space usage is a severe problem in practice (more so than time is). One may wonder if a prover could do better in its space usage. After all, naïvely verifying a theorem y with witness w may take time t but only space s with $s \ll t$. This is for example what we would expect if y encodes the correctness of some program that runs, given input w , for a long time on a single computer. Thus, it is reasonable to ask whether we can have PCP provers with space complexity that is only $O(s)$ (or even $s \cdot \text{polylog}(t)$) instead of $\Omega(t)$. A natural suggestion to alleviate the space requirements would be to allow the prover to naïvely evaluate the polynomial at every single one of the $\Omega(t)$ points in the hopes that, by not using FFTs, it could run in smaller space. This suggestion could go through as long as the “witness reduction” from w to the polynomial to evaluate could be done in polylog space; this is for example the case by using the computational Levin reduction² from random-access machines of [BSCGT12a] (which relies on the existence of collision-resistant hash functions). However, while the resulting prover would run in polylog space, its running time would now be $\Omega(t^2)$ — too slow.

On the other hand, we show that:

Theorem 1. *There is a one-round succinct MIP of knowledge with $O((\log t)^2)$ number of provers and constant soundness error, where:*

- the verifier is very efficient, AND
- each honest prover runs in quasilinear time.

Moreover, by using a computational reduction (so to obtain a Multi-Prover Interactive Argument), each honest prover runs in BOTH quasilinear time and polylog space.

While the above theorem may be quite surprising (since PCPs with similar efficiency as our construction are not known), it has a natural high-level explanation of how it circumvents the two difficulties afflicting PCPs:

² Levin reductions guarantee that there is not only an instance reduction, but a witness reduction that “goes both ways”; this ensures that proof of knowledge is preserved.

- **NO proof length vs soundness tradeoff.** With MIPs, there is no such thing as proof length, because the provers are *functions*. In other words, provers do not have to write down long proofs, but only answer specific questions of the verifier, which amounts to just a few evaluations of certain polynomials. Thus, we can design a simple and efficient MIP protocol by using the proximity testing tools at the basis of PCPs with long proofs and relatively high soundness, because we will not be paying for proof length. For example, we can leverage the high soundness of the Subspace vs. Point Test of Raz and Safra [RS97] (and its detailed analysis of Moshkovitz and Raz [MR08]) without worrying about the field size or proximity proof length.
- **NO prover with high space complexity.** Each honest prover is asked only a few evaluations of a polynomial, and thus naïve evaluation of the polynomial suffices; this, coupled with the appropriate (computational) reduction of [BSCGT12a] from random-access machines, will yield provers that only require polylog space.

Our construction ensures that the verifier is “adaptive” (namely, can generate queries without knowing in advance the theorem to be proved) and also ensures a proof of knowledge. Both properties play an important role in the other results in this paper.

Construction Outline. We follow a paradigm of probabilistic checking that is by now standard. We identify a convenient succinct *algebraic* constraint satisfaction problem (involving properties of polynomials) and then build on a low-degree proximity test to construct a probabilistic verifier (in our case, relying on the help of multiple non-communicating provers) for this problem.³

More concretely, the main ingredients of our construction are the low-degree test of Raz and Safra [RS97] for the Reed-Muller code (i.e., multivariate low-degree polynomials), a technique of Ben-Sasson and Sudan [BSS08, Lemma 4.5] for transforming a low-degree test for the Reed-Muller code to a low-degree test for the *Vanishing* Reed-Muller code (namely, the subcode consisting of those polynomials vanishing on a certain subset of the field), and a simple consistency check.

The technical challenge when constructing an MIP protocol instead of a PCP is that provers are functions and not strings, so that querying a given function at several locations becomes problematic. In our construction, we show how to “distribute” the various functions across as few provers as possible while at the same time ensuring that we do not have to query any given alleged function more than once. (Actually, we will have to query a function more than once only for the consistency check, and additional care will be needed to do so.)

For details, see full version of the paper.

Implementing MIPs with a Single Prover. Our MIP construction is simple and efficient, and so it suggests the possibility of obtaining alternative succinct argument constructions that are more efficient than PCP-based ones; thus, the question is:

Can we “implement” the MIP model with a *single prover* using cryptographic means, in a way that preserves its efficiency benefits?

³ Of course, one must also ensure that there are sufficiently efficient reductions from the universal language on random-access machines to this problem. That this is the case is not clear a priori. Nonetheless, we observe that the aforementioned reduction of [BSCGT12a] followed by an arithmetization of [Har04] will suffice.

(In particular, we are “not allowed” to deduce a PCP system [TS96] and rely on previous Kilian-type constructions!) We show two results in this direction, respectively described in Section 4 and Section 5.

Prior MIP Work. The question of implementing the MIP model for the purpose of constructing succinct arguments was asked by Dwork et al. [DLN⁺04] with the motivation being non-interactivity (rather than efficiency). Specifically, after pointing out the unsoundness of the proposed SNARK protocol of Aiello et al. [ABOR00], Dwork et al. suggested that an approach to constructing SNARKs would be to implement the MIP model by encrypting the query for each MIP prover using independent PIR instances and send all the encrypted queries to the same prover. They point out that for this approach to work (without additional assumptions on the MIP), the PIR should be immune to a specific class of attacks, called “spooky interactions”. They were not able to prove any PIR scheme to have this property, nor exhibit a counterexample. By now we know that such PIR schemes cannot be constructed via black-box reductions to falsifiable assumptions [GW11], but such PIR schemes are implied by the existence of SNARKs.

Later, Ishai et al. [IKO07] constructed a compiler from *linear* MIP protocols to four-message interactive arguments by leveraging *commitments with linear decommitment* (which can be obtained, e.g., from additively-homomorphic encryption schemes); by plugging into their compiler a linear MIP based on the Hadamard code, Ishai et al. obtained an argument where the prover-to-verifier communication is small, but the verifier running time is not succinct.

4 Four-Message Succinct Arguments from MIPs

A Simple But Wasteful Solution. Given a one-message succinct MIP (of knowledge) protocol and a collision-resistant hash family, it is easy to construct a four-message succinct argument (of knowledge): in the first message the verifier sends to the prover a seed for a collision-resistant hash; then the prover commits (separately and in parallel) to the evaluation table of each MIP prover by sending a Merkle commitment for each; then the verifier sends to the prover the desired message for each MIP prover, and finally the prover replies with an answer to each message, accompanied by an authentication path relative to the appropriate root. Committing to each evaluation table of each MIP prover before any messages are sent by the verifier ensures that a malicious prover cannot “correlate” its answers depending on the messages sent by the verifier in the following message. In other words, one can simply run a copy of Kilian’s protocol for each MIP prover, in parallel.

However, the above approach *does not preserve* the efficiency properties of the MIP protocol. For example, suppose that the first honest prover P_1 of the protocol is a function $f: A \rightarrow B$ that requires time t to evaluate at a single point. Then the above approach would require the prover to evaluate f at every point of A , which could in principle require as much time as $|A| \cdot t$. While the MIP protocol could have started out as quite efficient, now having to “think” of each prover as a table of values, instead of as a function, could sink the resulting construction into complete impracticality. (For example our MIP construction would indeed become too slow if we were to evaluate

each prover function everywhere on its domain, even if the resulting string is “only” of polynomial size.⁴⁾

Even if there are algorithms for evaluating f everywhere on A that are faster than the naïve “point-by-point” evaluation, these faster algorithms may come with hefty space requirements compared to the space requirement of a single evaluation of f . (As discussed in the previous section, this is for example the case when f is a polynomial and we seek to evaluate it faster, at many points, via the use of FFT techniques.)

Thus, the approach of using Merkle trees to implement the MIP protocol does not seem to take us any further, in terms of efficiency, than previous PCP-based protocols.

Ideally, we would want a way to implement the MIP protocol that *preserves* its efficiency: namely, the resulting succinct argument prover and verifier should have time and space complexities that are the same as in the corresponding MIP protocol, up to $\text{poly}(k)$ factors, where k is the security parameter (and ideally $\text{poly}(k)$ is also small).

With the above goal in mind, we prove the following result:

Theorem 2. *Assuming the existence of fully-homomorphic encryption, there exists an efficiency-preserving compiler that transforms a given succinct MIP of knowledge into a corresponding four-message succinct argument of knowledge.⁵ (Furthermore, the resulting succinct argument is “universal” in the sense that there is one protocol for all NP languages.)*

Our Efficiency-Preserving Solution. The tool that enables the above theorem is a generic construction of a *Succinct Multi-Function Commitment* (SMFC), a commitment that is, informally, an “efficiency-preserving analogue” of a Merkle tree commitment for functions. We next elaborate on this construction.

Generalizing the idea of commitments with linear decommitment of Ishai et al. [IKO07] to arbitrary functions, we define an SMFC to be a commitment scheme that works as follows: given a vector of ℓ functions $\vec{f}: A \rightarrow B$ where f_i can be evaluated in time t_i :⁶

- During a commitment phase, the sender and receiver interact; at the end of this phase, the sender has committed to the vector of functions \vec{f} ; both parties maintain a private state for the next phase.
- During a decommitment phase, the receiver may request the value of \vec{f} at a vector $\vec{\alpha} \in A^\ell$ by interacting with the receiver. At the end of this phase, the receiver either outputs $\vec{\beta} \in B^\ell$ or \perp .

The commitment guarantees that, if both parties are honest, $\vec{\beta} = \vec{f}(\vec{\alpha})$; moreover, it has the following *computational binding* property: for any efficient malicious sender, after

⁴ Furthermore, for functions with superpoly-size domain, writing out the evaluation table of the function is simply not feasible! (For example, this is the case in [IKO07].)

⁵ More precisely, the preservation of efficiency holds as long as each MIP prover has a sufficiently tight *deterministic* reduction to circuits. This technical condition is quite mild, and does hold for our MIP construction.

⁶ We can of course consider functions with different domains and ranges, but for simplicity here we set them equal.

the commitment phase, there is some vector of functions \vec{f}^* such that, for any query $\vec{\alpha}$, the receiver either outputs $\vec{f}^*(\vec{\alpha})$ or rejects (except with negligible probability).

The commitment is *succinct* in the sense that the sender runs in time $\text{poly}(k, \vec{t})$ and the receiver in time $\ell \cdot (\log |A| + \log |B|) \cdot \text{poly}(k)$. Crucially, the receiver running time does not depend on the size of the description or running time of \vec{f} .

We do not make the additional requirement that the commitment is hiding (although our commitment can be easily enhanced to also be function-hiding).

Theorem 3. *Assuming the existence of fully-homomorphic encryption, there is a succinct multi-function commitment. Moreover, if each function in \vec{f} can be computed “gate-by-gate” as a circuit by an evaluator algorithm in time t and space s , then the sender runs in time $\ell \cdot t \cdot \text{poly}(k)$ and space $\ell \cdot s \cdot \text{poly}(k)$ — we call such a property strongly succinct.⁷*

Succinct Arguments from Multi-function Commitments. Before moving on to describe some of the details behind Theorem 3, we briefly describe how to obtain succinct arguments given succinct multi-function commitments. The protocol essentially consists of two phases: in the first, the prover commits to ℓ functions corresponding to the provers P_1, \dots, P_ℓ given by the one round MIP protocol. Then, in the second phase the verifier produces the queries q_1, \dots, q_ℓ to the MIP provers, and asks for the corresponding decommitments. In case all the decommitments are valid, and the MIP verifier is satisfied by the corresponding answers, the verifier accepts.

The fact that the prover is committed in advance to each one of the ℓ functions, ensures that it cannot correlate the answers according to the joint vector of queries, allowing to prove security in a rather direct way.

The second part of Theorem 3 is what enables us, through our concrete MIP construction in the proof of Theorem 1, to perform the above transformation in an efficiency-preserving way, as claimed in Theorem 2.

We now provide more details regarding our succinct multi-function commitments (given by Theorem 3) and then elaborate on how efficiency is preserved.

The first step towards a multi-function commitment scheme is noting that it is enough to construct a single-function commitment. Once this is achieved a commitment to a vector of functions is done by independently committing to each one of the functions. Hence, we focus on the single-function case.

The starting point of our construction is the cut-and-choose delegation scheme of Chung et al. [CKV10], which works as follows:

- given a function f (to be delegated), during a setup phase, the verifier generates k independent encryptions c_1, \dots, c_k of 0 and then homomorphically computes

$$\hat{c}_1 := \text{Eval}(f, c_1), \dots, \hat{c}_k := \text{Eval}(f, c_k) ;$$

⁷ Succinct multi-function commitments can of course be constructed generically using succinct arguments of knowledge for NP together with Merkle hashing, but the efficiency of existing succinct argument constructions is not good enough to imply the strong efficiency properties that we need for succinct multi-function commitments (captured by the *strong* succinctness of the second part of the theorem).

- afterwards, during the “online” phase, if the verifier wishes to delegate the computation of f on a given input x , the verifier will encrypt k times the input x to obtain c_{k+1}, \dots, c_{2k} , and will send, in random order, c_1, \dots, c_{2k} to the prover;
- the prover should then homomorphically evaluate f on each ciphertext to obtain $\hat{c}'_1, \dots, \hat{c}'_{2k}$ (in some permuted order) and send them to the verifier; and
- the verifier will check that $\hat{c}'_1 = \hat{c}_1, \dots, \hat{c}'_k = \hat{c}_k$ and that $\hat{c}'_{k+1}, \dots, \hat{c}'_{2k}$ all decrypt to the same value.

Crucially, the verifier’s check on the challenges is done “on the ciphertexts”; this enables the security reduction to go through (and there is an attack if the check is instead performed on the underlying ciphertexts).

In our setting, we do not have a prover and a weak verifier, but a sender and a weak receiver. In particular, we are not interested in the sender computing a specific function f , but instead we are interested in the sender committing to *some* function (more precisely, vector of functions). Furthermore, we are not willing to allow the receiver to engage with the sender in an expensive setup phase.

We show that that in a sense we can “delegate” the expensive setup phase of the Chung et al. protocol to obtain an interactive function commitment protocol where the receiver is completely succinct. More precisely, because the sender is the one deciding the function to compute, we can simply ask him during the first round to evaluate the challenges $\hat{c}_1 := \text{Eval}(f, c_1), \dots, \hat{c}_k := \text{Eval}(f, c_k)$ himself for some function f of his choosing; of course, we cannot do this “in the clear” (as the sender would see the secret challenges), so we simply conduct the first round under another layer of fully-homomorphic encryption. In the resulting protocol, the receiver is indeed “fully succinct”.

While the intuition for the security of our modified protocol is strong, proving its binding property does not appear to be trivial. Our security reduction works as follows:

- We first prove a “weak” binding property for a non-amplified version of the protocol. We show that an adversary that is able, with high probability, to open to two different values of the function for the same query during two independent invocations of the decommitment phase can be used to break semantic security in a certain two-prover game.
- We then augment the weakly binding protocol so that we can apply a parallel repetition theorem for interactive arguments (such as [Hai09] or [CL10]), and get a protocol that is strongly binding. This is done by also having the decommitment phase executed under a (second) layer of fully-homomorphic encryption. Unlike typical commitments, where this transformation is rather straight forward, in our case this transformation turns out to be more involved, because the decommitment phase is interactive (and naively making it non-interactive causes an undesired computational overhead).

It may be surprising that, while the security reduction of the Chung et al. [CKV10] protocol is quite straightforward, a simple (and “direct”) proof of security for our protocol seems much harder to find. Perhaps the increased difficulty may be attributed to the fact that the Chung et al. [CKV10] protocol only has one round and in this case parallel repetition is, typically, “simpler” [CHS05]; in contrast, in our case proving security of

the protocol amounts to proving a special case of parallel repetition for interactive arguments with at least four messages for a protocol that is not of the public coin type (or more generally, one for which sampling random protocol continuations [Hai09] is quite challenging). For more details on the construction and its proof of security the reader is referred to the full version of this paper [BC12].

Prover Complexity. The sender in our SMFC protocol is required to (doubly) homomorphically evaluate each function in the vector \vec{f} ; in general, we do not know how to homomorphically evaluate a function that can be evaluated in time t in less than $t^2 \cdot \text{poly}(k)$ time [BSCGT12a], and that is why, for general functions, the running time of the sender of our protocol is only bounded by $\text{poly}(k, \vec{t})$.

If, however, we apply our SMFC protocol to a vector of functions \vec{f} where we do know that every function can be computed by a circuit of size at most t , then we can improve the time bound of the sender to $\ell \cdot t \cdot \text{poly}(k)$. While it is clear that a circuit C can be homomorphically evaluated in time $|C| \cdot \text{poly}(k)$, it is not as immediate that this is *also* the case for *double* homomorphic evaluation (and this fact is what enables the better time bound); indeed, the homomorphic evaluation algorithm is in fact two “local algorithms”, homomorphic addition and multiplication over \mathbb{F}_2 , iteratively applied to the gates of the circuit; each of these local algorithms can be generically reduced with a quadratic blow up to a corresponding circuit, but this time around we do not mind the quadratic blow up because the running time that we are squaring is only $\text{poly}(k)$ (and not, say, dependent on $|C|$).

Furthermore, if we also know that each function in our vector of functions \vec{f} can be evaluated as a circuit in time t and space s (by some “gate-by-gate” evaluator algorithm), then we can also bound the space complexity of the sender by $\ell \cdot s \cdot \text{poly}(k)$. Whenever $s \ll t$ (for example, when the circuits computing the functions have a somewhat succinct representation), this better space bound is very attractive.

Thus, overall, in our application of the SMFC protocol in Theorem 2, we use homomorphic encryption in a way that enables us to preserve the efficiency of the MIP protocol we construct in Theorem 1. (Though, of course that current FHE constructions are still quite inefficient in practice.) For details, see full version of the paper [BC12].

Interpretation. As discussed in the introduction, the motivation of Ishai et al. [IKO07] was to construct simpler succinct arguments by using simpler probabilistic checking tools, possibly at the expense of stronger cryptographic assumptions. (In their case, they relied on additively-homomorphic encryption instead of only collision-resistant hash functions.) Because both MIPs and PCPs based on Hadamard codes are very simple, working with Hadamard codes is a natural choice. However, techniques based on (the exponentially-long) Hadamard codes are somehow “too simple”: their simplicity comes from great proximity testing properties at the expense of a very poor rate; this poor rate makes it very difficult to construct succinct arguments (as merely producing a query indexing into the code is as expensive as the entire computation).

One interpretation of our Theorem 1 and Theorem 2 is that Ishai et al. in a sense “overshot”, and one can already find great simplicity while at the same time succeed at obtaining succinct arguments by using standard Reed-Muller proximity testing techniques to construct an MIP protocol and then implement it (based on the stronger cryptographic assumption of fully-homomorphic encryption).

We note that, unlike Kilian’s protocol, the interactive protocol we obtain in Theorem 2 is *not* of the public-coin type. It is an interesting open question to understand if there is an efficiency-preserving transformation from MIPs that yields a public-coin succinct argument. (Note that this question is interesting even in the random-oracle model, where it is also not clear how to obtain a public-coin function commitment!)

5 SNARKs from MIPs

Having discussed in Section 4 how to construct an interactive succinct argument from an MIP protocol, we next consider the natural question of whether we can succeed in the analogous task of constructing a succinct *non-interactive* argument of knowledge (SNARK) from an MIP protocol.

5.1 Known SNARK Constructions

Before we discuss our results in this direction, we briefly recall existing constructions.

In the Random-Oracle Model. Micali [Mic00] showed how to construct publicly-verifiable *one-message* succinct non-interactive arguments for NP, in the random oracle model, by applying the Fiat-Shamir paradigm [FS87] to Kilian’s protocol; later, Valiant [Val08] showed that Micali’s protocol is a proof of knowledge.

In the Plain Model. Micali’s protocol is essentially as good as one can hope for in the random oracle model. In the plain model, such “totally non-interactive” succinct arguments (against non-uniform provers) do not exist except for “quasi-trivial” languages (i.e., languages in $\text{BPtime}(n^{\text{polylog} n})$), because the impossibility results for statistical soundness can be directly extended to this case. Nonetheless, known impossibility results leave open the possibility of succinct non-interactive arguments in a slightly more relaxed model, where a generator (run by the verifier or a trusted entity) produces ahead of time a short *reference string* σ for the prover and a short *verification state* τ for the verifier (and both strings are independent of the statements to be proven later). Indeed, the definition of SNARKs in the plain model refers to this more relaxed setting.

A set of works [BCCT11, DFH11, GLR11] showed how to construct designated-verifier SNARKs (i.e., τ needs to remain secret) from a non-standard cryptographic primitive called *extractable collision-resistant hashes*. The protocol used in these works revisits a previous protocol proposed by Di Crescenzo and Lipmaa [DCL08] who, in turn, followed up on ideas of Dwork et al. [DLN⁺04] and Aiello et al. [ABOR00] for “squashing” Kilian’s protocol using succinct private information retrieval schemes.

Provable Limitations. Gentry and Wichs [GW11] showed that no non-interactive succinct argument can be proven to be (adaptively) sound via a black-box reduction to a falsifiable assumption (as defined in [Nao03]), even in the designated-verifier case. Their result suggests that non-standard assumptions, such as knowledge (extractability) assumptions may be inherent for constructing succinct *non-interactive* arguments (even if we were to drop the proof of knowledge requirement). Thus, constructing SNARKs by relying on (reasonable) knowledge assumptions might be justified.

In light of the [GW11] impossibility, it seems that a SNARK construction (which would likely rely on a non-standard assumption) would circumvent the result of Rothblum and Vadhan [RV09]. This suggests that exhibiting a SNARK construction that does not rely on PCPs at all (not even implicitly so) is a possibility. In fact, [BCCT12] show that this is indeed the case. In this paper, we seek to also do so but we restrict our focus to the problem of constructing SNARKs directly from MIP protocols; indeed, despite “implicitly” using PCPs, MIP protocols have constructions (such as the one in the proof of our Theorem 1) that seem much more efficient than current PCP constructions.

5.2 Our Theorem

We adopt a similar strategy to the one in [BCCT11, DFH11, GLR11]:

1. In Kilian’s protocol, the main cryptographic tool is collision-resistant hash functions, which are used to commit to the PCP string. By defining the non-standard cryptographic primitive of extractable collision-resistant hashes (ECRH), the works mentioned above show that it is possible to commit and reveal to the PCP string in a single message.

In our protocol from Section 4, the main cryptographic tool is fully-homomorphic encryption, which is used to succinctly commit to a vector of functions. Our goal is to define a (non-standard) cryptographic primitive for revealing values of a vector of functions in a *single* message (as opposed to via an interactive commitment scheme) and show that this is enough to obtain a SNARK from MIPs.

2. Then, [BCCT11] show that ECRHs are necessary to the construction of SNARKs and exhibit several candidate constructions.

We will also aim at showing that our primitive is necessary and at presenting candidate constructions.

The Extractable Primitive. We formulate a natural (though non-standard) primitive of *encryption with extractable homomorphism*: an encryption scheme where homomorphic operations cannot be performed “obliviously”. Specifically, these are (public- or secret-key) encryption schemes that are homomorphic (with respect to some given class of function \mathcal{F}). The scheme has a specialized decryption algorithm Dec that takes as input a “source” cipher $c = \text{Enc}_{\text{pk}}(m)$ and an evaluated cipher \hat{c} (which is allegedly the homomorphic evaluation of some function f on c). If \hat{c} is indeed such an evaluation, then $\text{Dec}_{\text{sk}}(c, \hat{c}) = f(m)$; however, if the adversary “obliviously” computes some function of c , we require that $\text{Dec}_{\text{sk}}(c, \hat{c}) = \perp$. More precisely, the guarantee is given in terms of extraction:

For any efficient \mathcal{A} there is an efficient extractor $\mathcal{E}_{\mathcal{A}}$ such that:
if \mathcal{A} , given $c = \text{Enc}_{\text{pk}}(m)$, outputs \hat{c} such that $\text{Dec}_{\text{sk}}(c, \hat{c}) = v \neq \perp$,
then $\mathcal{E}_{\mathcal{A}}$, given the same input c , outputs a function f , such that $f(m) = v$.

We stress that the extractable-homomorphism property does *not* provide any guarantee regarding the extracted function f (e.g., it may not be in the family \mathcal{F}). In particular, the property does not ensure targeted malleability in the sense of, e.g., [BSW11].

The requirement can then be extended to multiple ciphers (and multiple corresponding functions). Coupled with semantic-security, such a primitive effectively yields a two-message succinct multi-function commitment. Indeed, whenever the adversary manages to transform $c_1 = \text{Enc}_{\text{pk}_1}(q_1), \dots, c_\ell = \text{Enc}_{\text{pk}_\ell}(q_\ell)$ into evaluations $\hat{c}_1, \dots, \hat{c}_\ell$ that are accepted by the decryption algorithm, the extractor will output corresponding functions f_1, \dots, f_ℓ that “explain” the evaluations; moreover, because of semantic security, these functions are independent of the plaintext queries. More precisely, for any two vectors of queries \vec{q} and \vec{q}' and malicious adversary \mathcal{A} ,

$$\left\{ \vec{f} \mid \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^k) \\ \vec{c} \leftarrow \text{Enc}_{\text{pk}}(\vec{q}) \\ \vec{\epsilon} \leftarrow \mathcal{A}(\text{c}) \\ \vec{f} \leftarrow \mathcal{E}_{\mathcal{A}}(\text{c}) \end{array} \right\} \text{ and } \left\{ \vec{f}' \mid \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^k) \\ \vec{c}' \leftarrow \text{Enc}_{\text{pk}}(\vec{q}') \\ \vec{\epsilon}' \leftarrow \mathcal{A}(\text{c}') \\ \vec{f}' \leftarrow \mathcal{E}_{\mathcal{A}}(\text{c}') \end{array} \right\}$$

are computationally indistinguishable distributions over functions.

SNARKs from HEE. Given a fully-homomorphic encryption with extractable homomorphism, we are able to obtain a SNARK construction that is, in a sense, a “squashed” version of the interactive argument described in Section 4: the SNARK verifier simply generates MIP queries and sends them encrypted, using an extractable-homomorphism encryption scheme, to the SNARK prover. The prover, in turn, homomorphically evaluates each of the MIP provers on the corresponding encrypted query. The resulting scheme can be seen as an efficiency-preserving variant of the PIR-based protocol suggested by Dwork et al. [DLN⁺04] as a heuristic for squashing Kilian’s protocol.

We thus prove the following theorem (for details, see the full version of this paper):

Theorem 4. *Assuming the existence of fully-homomorphic encryption with extractable homomorphism, there exists an efficiency-preserving compiler that transforms a given succinct one-round MIP of knowledge into a corresponding SNARK.⁸*

Furthermore, the existence of fully-homomorphic encryption and SNARKs implies the existence of fully-homomorphic encryption with extractable homomorphism.

Is the Existence of Encryption with Extractable Homomorphism Plausible? Leaving efficiency aside and purely from an existential perspective, homomorphism-extractable encryption (HEE) schemes exist as long as extractable collision-resistant hashes (ECRHs) and FHE schemes exist: [BCCT11] showed how to construct SNARKs from ECRHs, and we will show that SNARKs and FHE together imply HEE schemes.

However, because our motivation for studying HEE schemes is the construction of SNARKs that are potentially more efficient than those built using PCPs, we believe that a more interesting question is to understand how plausible are “direct” constructions of HEE schemes. Consider, for example, the fully-homomorphic encryption (FHE) scheme of [BV11], based on LWE. Their FHE scheme likely does *not* have the extractable homomorphism property: consider an adversary that, given a ciphertext c of some message m , does not apply the honest evaluation algorithm to some function but

⁸ More precisely, as in Theorem 2, the preservation of efficiency holds as long as each MIP prover has a sufficiently tight deterministic reduction to circuits, which is the case for our MIP construction. (See Footnote 5.)

instead applies some arbitrary transformation to c to obtain a new ciphertext \hat{c} . With high probability, the new ciphertext \hat{c} will decrypt to some valid plaintext, but such an adversary may not “know” a corresponding function that can be homomorphically evaluated on the original plaintext c to result in the same ciphertext \hat{c} .

Intuitively, the reason is that the ciphertext space of [BV11] is not “sparse”: by merely applying an arbitrary operation in the ciphertext space, an adversary can “jump” from a valid ciphertext to another without being aware of what function he is applying on the underlying plaintext.⁹

Nonetheless, there *is* a natural method to generate sparsity in a given FHE scheme, and this method seems to serve as a heuristic for ensuring that the resulting encryption scheme can be plausibly assumed to have the extractable-homomorphism property.

The idea is to create sparsity via “amplification”: we consider a new encryption scheme, built out of the old one, that, in order to encrypt a given message m , encrypts m under many different independently-generated public keys, and the new decryption algorithm will check that a given vector of ciphertexts decrypts to a vector of messages that are *all equal*. For this amplified encryption scheme, it seems that the oblivious adversary described earlier does not work anymore: if the adversary only applies algebraic operations to the ciphertexts, he is likely to obtain ciphertexts that decrypt to different values, that is, an invalid ciphertext of the new scheme.

Acknowledgements. We thank Eli Ben-Sasson for discussions about MIP constructions. We also thank Ran Canetti, Omer Paneth, and Ben Riva for valuable discussions on MIP-based SNARKs.

References

- [ABOR00] Aiello, W., Bhatt, S., Ostrovsky, R., Rajagopalan, S.R.: Fast Verification of Any Remote Procedure Call: Short Witness-Indistinguishable One-Round Proofs for NP. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 463–474. Springer, Heidelberg (2000)
- [BC12] Bitansky, N., Chiesa, A.: Succinct arguments from multi-prover interactive proofs and their efficiency benefits. Cryptology ePrint Archive (2012)
- [BCC88] Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences 37(2), 156–189 (1988)
- [BCCT11] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. Cryptology ePrint Archive, Report 2011/443 (2011)
- [BCCT12] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for snarks and proof-carrying data. Cryptology ePrint Archive, Report 2012/095 (2012)
- [BFL90] Babai, L., Fortnow, L., Lund, C.: Nondeterministic exponential time has two-prover interactive protocols. In: Proceedings of the 31st Annual Symposium on Foundations of Computer Science, SFCS 1990, pp. 16–25 (1990)

⁹ Interestingly, sparsity also arises as a necessary condition when studying candidate constructions of ECRHs [BCCT11].

- [BFLS91] Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in polylogarithmic time. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, STOC 1991, pp. 21–32 (1991)
- [BG08] Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM Journal on Computing 38(5), 1661–1694 (2008); Preliminary version appeared in CCC 2002
- [BHZ87] Boppana, R.B., Hästad, J., Zachos, S.: Does co-NP have short interactive proofs? Information Processing Letters 25(2), 127–132 (1987)
- [BOGKW88] Ben-Or, M., Goldwasser, S., Kilian, J., Wigderson, A.: Multi-prover interactive proofs: how to remove intractability assumptions. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 113–131 (1988)
- [BP04a] Bellare, M., Palacio, A.: The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
- [BP04b] Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
- [BSCGT12a] Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: Fast reductions from RAMs to delegatable succinct constraint satisfaction problems. Cryptology ePrint Archive, Report Report 2012/071 (2012)
- [BSCGT12b] Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: On the concrete-efficiency threshold of probabilistically-checkable proofs. Electronic Colloquium on Computational Complexity, TR12-045 (2012)
- [BSGH⁺05] Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.: Short PCPs verifiable in polylogarithmic time. In: Proceedings of the 20th Annual IEEE Conference on Computational Complexity, CCC 2005, pp. 120–134 (2005)
- [BSS08] Ben-Sasson, E., Sudan, M.: Short PCPs with polylog query complexity. SIAM Journal on Computing 38(2), 551–607 (2008)
- [BSW11] Boneh, D., Segev, G., Waters, B.: Targeted malleability: Homomorphic encryption for restricted computations. Cryptology ePrint Archive, Report 2011/311 (2011)
- [BV11] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, FOCS 2011 (2011)
- [CHS05] Canetti, R., Halevi, S., Steiner, M.: Hardness Amplification of Weakly Verifiable Puzzles. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 17–33. Springer, Heidelberg (2005)
- [CKV10] Chung, K.-M., Kalai, Y., Vadhan, S.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
- [CL10] Chung, K.-M., Liu, F.-H.: Parallel Repetition Theorems for Interactive Arguments. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 19–36. Springer, Heidelberg (2010)
- [CT10] Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. In: Proceedings of the 1st Symposium on Innovations in Computer Science, ICS 2010, pp. 310–331 (2010)
- [Dam92] Damgård, I.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)

- [DCL08] Di Crescenzo, G., Lipmaa, H.: Succinct NP Proofs from an Extractability Assumption. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 175–185. Springer, Heidelberg (2008)
- [DFH11] Damgård, I., Faust, S., Hazay, C.: Secure two-party computation with low communication. Cryptology ePrint Archive, Report 2011/508 (2011)
- [DLN⁺04] Dwork, C., Langberg, M., Naor, M., Nissim, K., Reingold, O.: Succinct NP proofs and spooky interactions (December 2004), <http://www.openu.ac.il/home/mikel/papers/spooky.ps>
- [FS87] Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
- [GGPR12] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. Cryptology ePrint Archive, Report 2012/215 (2012)
- [GH98] Goldreich, O., Håstad, J.: On the complexity of interactive proofs with bounded communication. Information Processing Letters 67(4), 205–214 (1998)
- [GLR11] Goldwasser, S., Lin, H., Rubinstein, A.: Delegation of computation without rejection problem from designated verifier CS-proofs. Cryptology ePrint Archive, Report 2011/456 (2011)
- [GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989); Preliminary version appeared in STOC 1985
- [Gro10] Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
- [GVW02] Goldreich, O., Vadhan, S., Wigderson, A.: On interactive proofs with a laconic prover. Computational Complexity 11(1/2), 1–53 (2002)
- [GW11] Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, STOC 2011, pp. 99–108 (2011)
- [Hai09] Haitner, I.: A parallel repetition theorem for any interactive argument. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, pp. 241–250 (2009)
- [Har04] Harsha, P.: Robust PCPs of Proximity and Shorter PCPs. PhD thesis, MIT, EECS (September 2004)
- [IKO07] Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient arguments without short PCPs. In: Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity, CCC 2007, pp. 278–291 (2007)
- [Kil92] Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC 1992, pp. 723–732 (1992)
- [Lip11] Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. Cryptology ePrint Archive, Report 2011/009 (2011)
- [Mic00] Micali, S.: Computationally sound proofs. SIAM Journal on Computing 30(4), 1253–1298 (2000); Preliminary version appeared in FOCS 1994
- [MR08] Moshkovitz, D., Raz, R.: Two-query PCP with subconstant error. Journal of the ACM 57, 1–29 (2008); Preliminary version appeared in FOCS 2008
- [Nao03] Naor, M.: On Cryptographic Assumptions and Challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)

- [RS97] Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, STOC 1997, pp. 475–484 (1997)
- [RV09] Rothblum, G.N., Vadhan, S.: Are PCPs inherent in efficient arguments? In: Proceedings of the 24th IEEE Annual Conference on Computational Complexity, CCC 2009, pp. 81–92 (2009)
- [Sha92] Shamir, A.: IP = PSPACE. *Journal of the ACM* 39(4), 869–877 (1992)
- [TS96] Ta-Shma, A.: A note on PCP vs. MIP. *Information Processing Letters* 58, 135–140 (1996)
- [Val08] Valiant, P.: Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008)
- [Wee05] Wee, H.: On Round-Efficient Argument Systems. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 140–152. Springer, Heidelberg (2005)

On the Security of TLS-DHE in the Standard Model

Tibor Jager¹, Florian Kohlar², Sven Schäge^{3,*}, and Jörg Schwenk²

¹ Karlsruhe Institute of Technology, Germany
tibor.jager@kit.edu

² Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{florian.kohlar,joerg.schwenk}@rub.de

³ University College London, United Kingdom
s.schage@ucl.ac.uk

Abstract. TLS is the most important cryptographic protocol in use today. However, up to now there is no complete cryptographic security proof in the standard model, nor in any other model. We give the first such proof for the core cryptographic protocol of TLS ciphersuites based on ephemeral Diffie-Hellman key exchange (TLS-DHE), which include the cipher suite `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` mandatory in TLS 1.0 and TLS 1.1. It is impossible to prove security of the TLS Handshake protocol in any classical key-indistinguishability-based security model (like for instance the Bellare-Rogaway or the Canetti-Krawczyk model), due to subtle issues with the encryption of the final `Finished` messages. Therefore we start with proving the security of a truncated version of the TLS-DHE Handshake protocol, which has been considered in previous works on TLS. Then we define the notion of authenticated and confidential channel establishment (ACCE) as a new security model which captures precisely the security properties expected from TLS in practice, and show that the combination of the TLS Handshake with data encryption in the TLS Record Layer can be proven secure in this model.

Keywords: authenticated key exchange, SSL, TLS, provable security, ephemeral Diffie-Hellman.

1 Introduction

Transport Layer Security (TLS) is the single most important Internet security mechanism today. Session keys in TLS are established in the TLS Handshake protocol, using either encrypted key transport (TLS-RSA) or (ephemeral) Diffie-Hellman key exchange (TLS-DH(E)), whereas authentication can be provided mutual or server-only. Due to a subtle interleaving of the TLS Handshake with the TLS Record Layer it is impossible to prove the security of TLS using well-established security models [4,10,9], which define security via indistinguishability of keys (see [18] for a detailed description of this issue). Therefore there is no security proof for the complete protocol up to now.

* Supported by EPSRC grant number EP/G013829/1.

The paradox that the most important authenticated key-exchange (AKE) protocol cannot be proven secure in any existing security model can be solved in two ways. Either one considers a modified version of the TLS Handshake protocol ('truncated TLS'), which was subject to previous work [20], or a new security model for the combination of TLS Handshake protocol and data encryption in the TLS Record Layer must be devised. In this paper we follow both approaches.

1.1 Contributions

We provide new security results for the core cryptographic protocol of TLS based on ephemeral Diffie-Hellman key exchange (TLS-DHE).

First we give a formal proof that the truncated version of the TLS-DHE Handshake protocol from [20] is a secure authenticated key exchange protocol. We consider a security model which extends the well-known Bellare-Rogaway model [4] to adaptive corruptions and perfect forward secrecy in the public-key setting (cf. [6]). This allows to compare our results to previous work.

Second we define the notion of authenticated and confidential channel establishment (ACCE). ACCE protocols are an extension of AKE protocols, in the sense that the symmetric cipher is integrated into the model. In contrast to AKE protocols, where one requires *key indistinguishability*, we demand that a secure ACCE protocol allows to establish a 'secure communication channel' in the sense of stateful length-hiding authenticated encryption [22]. Loosely speaking, an ACCE channel guarantees that messages written to this channel are confidential (indistinguishable), and even the length of messages is concealed up to some granularity, and that a sequence of messages read from this channel corresponds exactly to the sequence of messages sent by the legitimate sender (of course up to dropping messages at the very end of the sequence, which is always possible). This captures exactly the properties expected from TLS-like protocols in practice. We prove that the combination of the TLS Handshake protocol with the TLS Record Layer forms a secure ACCE protocol, if the TLS Record Layer provides security in the sense of length-hiding authenticated encryption. Note that the latter was proven recently by Paterson *et al.* [22] for CBC-based Record Layer protocols.

The analyses of both truncated TLS-DHE (as an AKE protocol) and TLS-DHE (as an ACCE protocol) require, that the building blocks of TLS-DHE (digital signature scheme, Diffie-Hellman key exchange, symmetric cipher) meet certain security properties. The majority of these properties are standard assumptions, solely for the pseudo-random function we require an additional non-standard security assumption, which is a variant of the Oracle Diffie-Hellman assumption introduced by Abdalla, Bellare, and Rogaway [1]. We explain in Section 6 why such an assumption seems hard to avoid. Our proof is stated for *mutual authentication*, i.e., the client authenticates itself using a client certificate. This allows us to base our work on standard definitions for secure authenticated key exchange.

1.2 Interpretation

Our results show that the core cryptographic protocol of TLS-DHE is cryptographically sound, if its building blocks are suitably secure (the full version [18]

of this paper contains an analysis to what extent the concrete building blocks of TLS meet the required properties, here we can build upon previous work that considered particular components of TLS).

We note that TLS-DHE is much less used in practice than TLS with encrypted key transport (TLS-RSA). Moreover, we consider mutual authentication (that is, both the client and the server are in possession of a certified public key, which is used in the protocol to mutually authenticate each other), which is also rarely used in practice. We believe that our analysis of TLS-DHE is nevertheless of practical value, for the following reasons:

First, the TLS-DHE-based ciphersuite `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA` is mandatory for TLS 1.0 and 1.1, which are both still in widespread use. Only the most recent version TLS 1.2 prescribes TLS-RSA as mandatory. So one could theoretically configure a considerable amount of servers to use only TLS-DHE and benefit from the provable security guarantees of TLS-DHE as provided in our security analysis.

Second, we can show that TLS-DHE provides *perfect forward secrecy* – a very strong form of security, which basically states that future compromises of long-term secrets do no threaten past communication sessions. With encrypted key transport, as in TLS-RSA, this is not achievable, since an attacker that compromises the long-term key (the private decryption key) can easily obtain session keys from previous sessions by just decrypting recorded ciphertexts. To better protect users from the consequences of such key compromise attacks and offer better long-term security, service providers might therefore consider to switch to the (exclusive) use of TLS-DHE. Recently, Google has made a first step in that direction, by announcing that it will switch over to TLS-DHE as the default key exchange method for its services to provide (and push) perfect forward secrecy [2].

Third, it seems that giving a security proof of the actually most widespread option TLS-RSA is impossible in the standard model. Any approach we can think of would require IND-CCA-security of the encryption scheme used to transport the premaster secret from the client to the server, as otherwise we cannot simulate protocol executions while still being able to argue with indistinguishability of premaster secrets. But unfortunately it is well-known that the RSA-PKCS v1.5 scheme used in TLS is vulnerable to chosen-ciphertext attacks [7]. This problem was circumvented in previous work by either using an abstract public-key encryption scheme which is IND-CCA-secure [20], or by assuming PKCS#1 v2.0 (RSA-OAEP), which is not used in TLS, and omitting authentication [17].

Our work can also be seen as a ‘stepping stone’ towards a TLS version with a complete security proof in the standard model. Essentially, we identify certain security properties and prove that the TLS protocol framework yields a secure ACCE protocol under the assumption that the TLS building blocks satisfy these properties.

1.3 Related Work

Because of its eminent role, TLS and its building blocks have been subject to several security analyses. We mention only the works closely related to ours here, a more complete overview can be found in [18].

Gajek *et al.* [17] presented the first security analysis of the complete TLS protocol, combining Handshake and Record Layer, in the UC framework [9] for all three key exchange protocols static Diffie-Hellman, ephemeral signed Diffie-Hellman, and encrypted key transport. The ideal functionalities described in this paper are much weaker than the security guarantees we expect from TLS, since only unauthenticated key exchange is considered. The paper furthermore assumes that RSA-OAEP is used for encrypted key transport, which is not the case for current versions of TLS.

Morissey *et al.* [20] analysed, in a paper that is closest to our results, the security of the truncated TLS Handshake protocol in the random oracle model and provided a modular proof of security. They make extensive use of the random oracle model to separate the three layers in the TLS Handshake they define, and to switch from computational to indistinguishability based security models. The use of the random oracle model is justified by the authors of [20] since it seems impossible to prove the PKCS#1 v1.5 based ciphersuites of TLS secure in the standard model. This argumentation does not affect our work, since we consider Diffie-Hellman-based ciphersuites.

Paterson *et al.* [22] introduce the notion of length-hiding authenticated encryption, which captures the properties expected from the data encryption in the TLS Record Layer. Most importantly, they were able to show that CBC-based ciphersuites of TLS 1.1 and 1.2 meet this security notion. This work matches nicely our results on the TLS Handshake protocol, and is an important building block for our work.

Very recently, Brzuska *et al.* [8] proposed relaxed game-based security notions for key exchange. This approach may serve as an alternative to our ACCE-based approach to circumvent the impossibility of proving the TLS Handshake protocol secure in a key-indistinguishability-based security model.

1.4 Remark on Our Choice of the Security Model

Authenticated key exchange (AKE) is a basic building block in modern cryptography. However, since many different security models for different purposes exist [3,4,6,9,10,13,19,12], the choice of the right model is not an easy task, and must be considered carefully. We have to take into account that we cannot modify any detail in the TLS protocol, nor in the network protocols preceding it. We have chosen an enhanced variant of the first model of Bellare and Rogaway [4]. Variants of this model have also been studied by [12,6], and especially by [20]. Detailed reasons for our choice are given in the full version [18].

2 Preliminaries and Definitions

We denote with \emptyset the empty string, and with $[n] = \{1, \dots, n\} \subset \mathbb{N}$ the set of integers between 1 and n . If A is a set, then $a \xleftarrow{\$} A$ denotes the action of sampling a uniformly random element from A . If A is a probabilistic algorithm, then $a \xleftarrow{\$} A$ denotes that A is run with fresh random coins and returns a . In addition

to the complexity assumption described in the sequel, we need the standard security notions of digital signatures (EUF-CMA), pseudo-random functions, and the Decisional Diffie-Hellman (DDH) assumption. These are detailed in the full version [18].

The PRF-Oracle-Diffie-Hellman (PRF-ODH) Assumption. Let G be a group with generator g . Let PRF be a deterministic function $z = \text{PRF}(X, m)$, taking as input a key $X \in G$ and some bit string m , and returning a string $z \in \{0, 1\}^\mu$. Consider the following security experiment played between a challenger \mathcal{C} and an adversary \mathcal{A} .

1. The adversary \mathcal{A} outputs a value m .
2. The Challenger samples $u, v \xleftarrow{\$} [q]$, $z_1 \xleftarrow{\$} \{0, 1\}^\mu$ uniformly random and sets $z_0 := \text{PRF}(g^{uv}, m)$. Then it tosses a coin $b \in \{0, 1\}$ and returns z_b , g^u and g^v to the adversary.
3. The adversary may query a pair (X, m') with $X \neq g^u$ to the challenger. The challenger replies with $\text{PRF}(X^v, m')$.
4. Finally the adversary outputs a guess $b' \in \{0, 1\}$.

Definition 1. We say that the PRF-ODH problem is $(t, \epsilon_{\text{prfodh}})$ -hard with respect to G and PRF , if for all adversaries \mathcal{A} that run in time t it holds that

$$|\Pr[b = b'] - 1/2| \leq \epsilon_{\text{prfodh}}.$$

The PRF-Oracle-Diffie-Hellman (PRF-ODH) assumption is a variant of the ODH assumption introduced by Abdalla, Bellare and Rogaway in [1], adopted from hash functions to PRFs. In contrast to allowing a polynomial number of queries as in the original assumption [1], we allow only a single oracle query.

Stateful Length-Hiding Authenticated Encryption. The following description and security model was obtained from the authors of [22] via personal communication. See [22] for a detailed discussion and motivation of this security notion.

A stateful symmetric encryption scheme consists of two algorithms $\text{StE} = (\text{StE.Enc}, \text{StE.Dec})$. Algorithm $(C, st'_e) \xleftarrow{\$} \text{StE.Enc}(k, \text{len}, H, m, st_e)$ takes as input a secret key $k \in \{0, 1\}^\kappa$, an output ciphertext length $\text{len} \in \mathbb{N}$, some header data $H \in \{0, 1\}^*$, a plaintext $m \in \{0, 1\}^*$, and the current state $st_e \in \{0, 1\}^*$, and outputs either a ciphertext $C \in \{0, 1\}^{\text{len}}$ and an updated state st'_e or an error symbol \perp if for instance the output length len is not valid for the message m . Algorithm $(m', st'_d) = \text{StE.Dec}(k, H, C, st_d)$ takes as input a key k , header data H , a ciphertext C , and the current state $st_d \in \{0, 1\}^*$, and returns an updated state st'_d and a value m' which is either the message encrypted in C , or a distinguished error symbol \perp indicating that C is not a valid ciphertext. Both encryption state st_e and decryption state st_d are initialized to the empty string \emptyset . Algorithm StE.Enc may be probabilistic, while StE.Dec is always deterministic.

Definition 2. We say that a stateful symmetric encryption scheme $\text{StE} = (\text{StE.Init}, \text{StE.Enc}, \text{StE.Dec})$ is $(t, \epsilon_{\text{SLHAE}})$ -secure, if $\Pr[b = b'] \leq \epsilon_{\text{SLHAE}}$ for all adversaries \mathcal{A} running in time at most t in the following experiment.

<u>Encrypt(m_0, m_1, len, H):</u> $u := u + 1$ $(C^{(0)}, st_e^{(0)}) \xleftarrow{\$} \text{StE.Enc}(k, \text{len}, H, m_0, st_e)$ $(C^{(1)}, st_e^{(1)}) \xleftarrow{\$} \text{StE.Enc}(k, \text{len}, H, m_1, st_e)$ If $C^{(0)} = \perp$ or $C^{(1)} = \perp$ then return \perp $(C_u, st_e) := (C^{(b)}, st_e^{(b)})$ Return C_u	<u>Decrypt(C, H):</u> $v := v + 1$ If $b = 0$, then return \perp $(m, st_d) = \text{StE.Dec}(k, H, C, st_d)$ If $v > u$ or $C \neq C_v$, then phase := 1 If phase = 1 then return m Return \perp
--	---

Fig. 1. Encrypt and Decrypt oracles in the stateful LHAE security experiment

- Choose $b \xleftarrow{\$} \{0, 1\}$ and $k \xleftarrow{\$} \{0, 1\}^\kappa$, and set $st_e := \emptyset$ and $st_d := \emptyset$,
- run $b' \xleftarrow{\$} \mathcal{A}^{\text{Encrypt}, \text{Decrypt}}$.

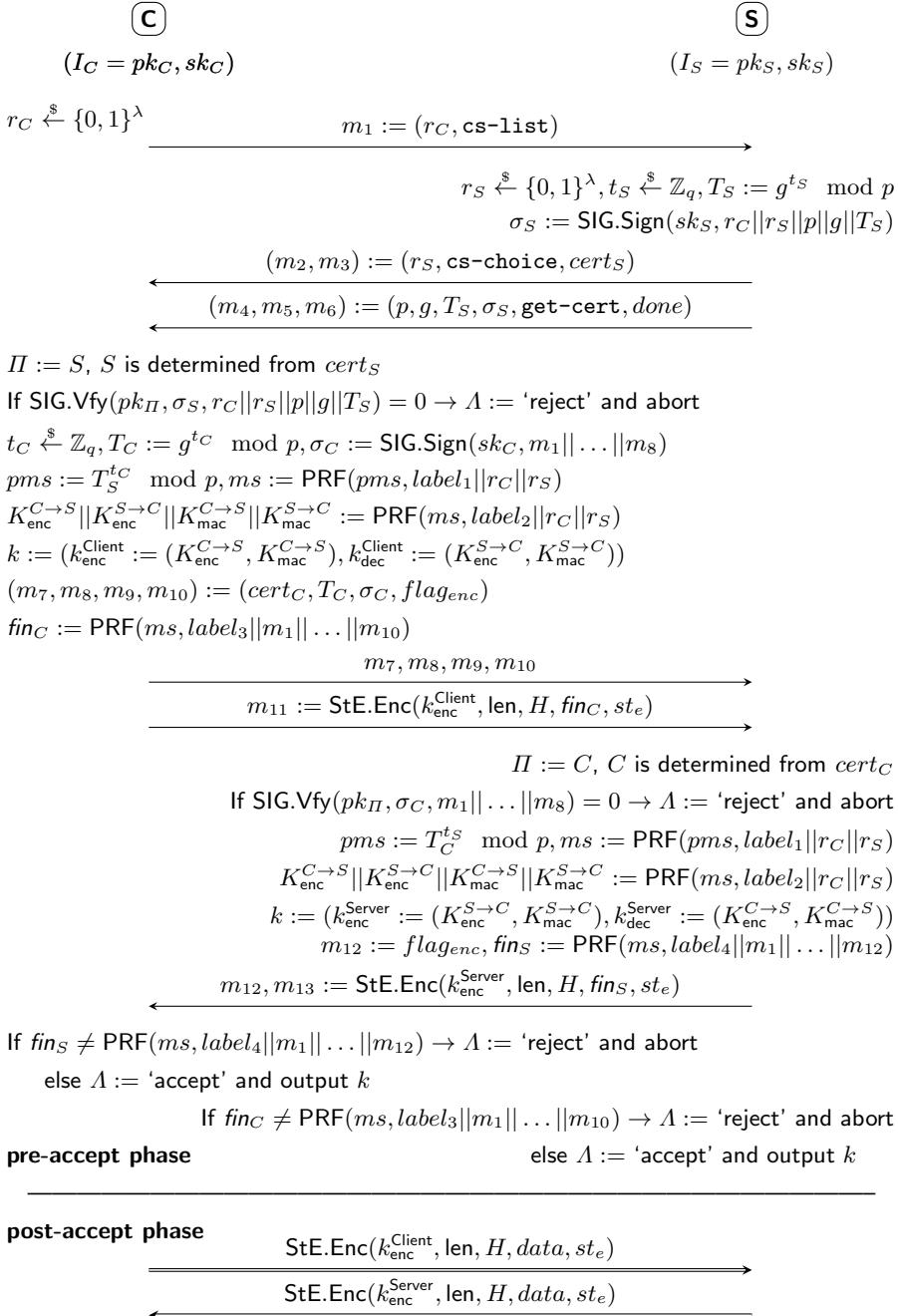
Here $\mathcal{A}^{\text{Encrypt}, \text{Decrypt}}$ denotes that \mathcal{A} has access to two oracles Encrypt and Decrypt. The encryption oracle $\text{Encrypt}(m_0, m_1, \text{len}, H)$ takes as input two messages m_0 and m_1 , length-parameter len and header data H . It maintains a counter u which is initialized to 0. Oracle $\text{Decrypt}(C, H)$ takes as input a ciphertext C and header H , and keeps a counter v and a variable **phase**, both are initialized to 0. Both oracles process a query as defined in Figure 1.

3 Transport Layer Security

The current version of TLS is 1.2 [16] coexists with its predecessors TLS 1.0 [14] and TLS 1.1 [15]. In the following we give a description of all messages sent during the TLS Handshake with ephemeral Diffie-Hellman key exchange and client authentication (i.e. for ciphersuites `TLS_DHE_*`). This description and its illustration in Figure 2 are valid for *all* TLS versions since v1.0. Our description makes use of several ‘state variables’ ($\Lambda, k, \Pi, \rho, st$). For instance, variable $\Lambda \in \{\text{accept}, \text{reject}\}$ determines whether one party ‘accepts’ or ‘rejects’ an execution of the protocol, or variable k stores the session key. These variables will also appear later in our security model (Section 4).

The TLS Handshake protocol consists of 13 messages, whose content ranges from constant byte values to tuples of cryptographic values. Not all messages are relevant for our security proof, we list them merely for completeness. All messages are prepended with a numeric tag that identifies the type of message, a length value, and the version number of TLS. All messages are sent through the TLS Record Layer, which at startup provides no encryption nor any other cryptographic transformations.

Message m_1 is the `Client Hello` message. It contains four values, two of which are optional. For our analysis the only important value is r_C , the random value chosen by the client. It consists of 32 bytes (256 Bits), where 4 Bytes are usually used to encode the local time of the client. The remaining 28 Bytes are chosen randomly by the client. This is followed by a list `cs-list` of *ciphersuites*, where each ciphersuite is a tuple of key exchange method, signing, encryption and MAC algorithms, coded as two bytes. Data compression is possible before encryption and is signaled by the inclusion of zero or more compression methods.

**Fig. 2.** Handshake protocol for ciphersuites **TLS_DHE_*** with client authentication

The **Server Hello** message m_2 has the same structure as **Client Hello**, with the only exception that at most one ciphersuite and one compression method can be present. Message m_3 may contain a certificate (or a chain of certificates, which is not considered in this paper) and the public key in the certificate must match the ciphersuite chosen by the server. For ephemeral Diffie-Hellman key exchange, the public key may be any key that can be used to sign messages. The Diffie-Hellman (DH) key exchange parameters are contained in the **Server Key Exchange** message m_4 , including information on the DH group (e.g. prime number p and generator g for a prime-order q subgroup of \mathbb{Z}_p^*), the DH share T_S , and a signature computed over these values plus the two random numbers r_C and r_S . The next two messages are very simple: the **Certificate Request** message m_5 only contains a list of certificate types that the client may use to authenticate itself, and the **Server Hello Done** message m_6 does not contain any data, but consists only of a constant tag with byte-value ‘14’ and a length value ‘0’.

Having received these messages, the signature σ_S is verified. If this fails, the client ‘rejects’ and aborts, otherwise the client completes the key exchange and computes the cryptographic keys. The **Client Certificate** message m_7 contains a signing certificate $cert_C$ with the public key pk_C of the client.¹ Message m_8 is called **Client Key Exchange**, and contains the Diffie-Hellman share T_C of the client. To authenticate the client, a signature σ_C is computed on a concatenation of all previous messages (up to m_8) and padded prefixes and sent in the **Certificate Verify** message m_9 .

The client is now also able to compute the *premaster secret* pms , from which all further secret values are derived. After computing the *master secret* ms , it is stored for the lifetime of the TLS session, and pms is erased from memory. The master secret ms is subsequently used, together with the two random nonces, to derive all encryption and MAC keys as well as the **Client Finished** message fin_C . More precisely, the key material is computed as

$$K_{\text{enc}}^{C \rightarrow S} || K_{\text{enc}}^{S \rightarrow C} || K_{\text{mac}}^{C \rightarrow S} || K_{\text{mac}}^{S \rightarrow C} := \text{PRF}(ms, \text{label}_2 || r_C || r_S). \quad (1)$$

After these computations have been completed, the keys are handed over to the TLS Record Layer of the client, which is now able to MAC and encrypt any data. To signal the ‘start of encryption’ to the server, a single message m_{10} (**Change Cipher Spec**) with byte value ‘1’ ($flag_{\text{enc}}$) is sent unencrypted to S . Then message m_{11} consists of an authenticated encryption of the **Client Finished** message fin_C . After the server has received messages m_7, m_8, m_9 , the server verifies the signature in m_9 . If this fails, the server ‘rejects’ (i.e. sets $\Lambda = \text{'reject'}$) and aborts. Otherwise it first determines pms and ms . From this the encryption and MAC keys are computed as in (1). It can then decrypt m_{11} and check fin_C by computing the pseudo-random value on the messages sent and received by the server. If this check fails, it ‘rejects’ and aborts. If the check is successful, it ‘accepts’ (i.e. sets $\Lambda = \text{'accept'}$), computes the **Server Finished**

¹ When either party receives a certificate $cert_X$, the partner id is set to $\Pi := X$.

message fin_S and sends messages m_{12} and m_{13} to the client. If the check of fin_S on the client side is successful, the client also ‘accepts’.

The obtained keys can now be used to transmit payload data in the TLS Record Layer using a stateful symmetric encryption scheme (StE.Enc, StE.Dec).

ABBREVIATED TLS HANDSHAKES, SIDE-CHANNELS AND CROSS-PROTOCOL ATTACKS. In our analysis, we do not consider the abbreviated TLS Handshake, but note that the server can always enforce an execution of the full protocol. Moreover, we do not consider attacks based on side-channels, such as error messages, or cross-protocol attacks like [24].

4 AKE Protocols

While the established security models for, say, encryption (e.g. IND-CPA or IND-CCA security), or digital signatures (e.g., EUF-CMA), are clean and simple, a more complex model is required to model the capabilities of active adversaries to define secure authenticated key-exchange. An important line of research [6,10,19,13] dates back to Bellare and Rogaway [4], where an adversary is provided with an ‘execution environment’, which emulates the real-world capabilities of an active adversary, which has full control over the communication network. In the sequel we describe a variant of this model, which captures adaptive corruptions, perfect forward secrecy, and security against key-compromise impersonation attacks in a public-key setting.

EXECUTION ENVIRONMENT. Consider a set of parties $\{P_1, \dots, P_\ell\}$, $\ell \in \mathbb{N}$, where each party $P_i \in \{P_1, \dots, P_\ell\}$ is a (potential) protocol participant and has a long-term key pair (pk_i, sk_i) . To model several sequential and parallel executions of the protocol, each party P_i is modeled by a collection of oracles π_i^1, \dots, π_i^d for $d \in \mathbb{N}$. Each oracle π_i^s represents a process that executes one single instance of the protocol. All oracles π_i^1, \dots, π_i^d representing party P_i have access to the same long-term key pair (pk_i, sk_i) of P_i and to all public keys pk_1, \dots, pk_ℓ . Moreover, each oracle π_i^s maintains as internal state the following variables:

- $\Lambda \in \{\text{accept}, \text{reject}\}$.
- $k \in \mathcal{K}$, where \mathcal{K} is the keyspace of the protocol.
- $\Pi \in \{1, \dots, \ell\}$ containing the intended communication partner, i.e., an index j that points to a public key pk_j used to perform authentication.²
- Variable $\rho \in \{\text{Client}, \text{Server}\}$.
- Some additional temporary state variable st (which may, for instance, be used to store ephemeral Diffie-Hellman exponents or a transcript of messages).

The internal state of each oracle is initialized to $(\Lambda, k, \Pi, \rho, st) = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$, where $V = \emptyset$ denotes that variable V is undefined. Furthermore, we will always

² We assume that each party P_i is uniquely identified by its public key pk_i . In practice, several keys may be assigned to one identity. Furthermore, there may be other ways to determine identities, for instance by using certificates. However, this is out of scope of this paper.

assume (for simplicity) that $k = \emptyset$ if an oracle has not reached `accept`-state (yet), and contains the computed key if an oracle is in `accept`-state, so that we have

$$k \neq \emptyset \iff \Lambda = \text{accept}. \quad (2)$$

An adversary may interact with these oracles by issuing the following queries.

- **Send**(π_i^s, m): The adversary can use this query to send message m to oracle π_i^s . The oracle will respond according to the protocol specification, depending on its internal state. If the attacker asks the first **Send**-query to oracle π_i^s , then the oracle checks whether $m = \top$ consists of a special ‘initialization’ symbol \top . If true, then it sets its internal variable $\rho := \text{Client}$ and responds with the first protocol message. Otherwise it sets $\rho := \text{Server}$ and responds as specified in the protocol.³ The variables Λ, k, Π, st are also set after certain **Send**-queries.⁴
- **Reveal**(π_i^s): Oracle π_i^s responds to a **Reveal**-query with the contents of variable k . Note that we have $k \neq \emptyset$ if and only if $\Lambda = \text{accept}$, see (2).
- **Corrupt**(P_i): Oracle π_i^1 responds with the long-term secret key sk_i of party P_i .⁵ If **Corrupt**(P_i) is the τ -th query issued by \mathcal{A} , then we say that P_i is *τ -corrupted*. For parties that are not corrupted we define $\tau := \infty$.
- **Test**(π_i^s): This query may be asked only once throughout the game. If π_i^s has state $\Lambda \neq \text{accept}$, then it returns some failure symbol \perp . Otherwise it flips a fair coin b , samples an independent key $k_0 \xleftarrow{\$} \mathcal{K}$, sets $k_1 = k$ to the ‘real’ key computed by π_i^s , and returns k_b .

SECURITY DEFINITION. Bellare and Rogaway [4] have introduced the notion of *matching conversations* in order to define correctness and security of an AKE protocol precisely. We denote with $T_{i,s}$ the sequence that consists of all messages sent and received by π_i^s in chronological order (not including the initialization-symbol \top). We also say that $T_{i,s}$ is the *transcript* of π_i^s . For two transcripts $T_{i,s}$ and $T_{j,t}$, we say that $T_{i,s}$ is a *prefix* of $T_{j,t}$, if $T_{i,s}$ contains at least one message, and the messages in $T_{i,s}$ are identical to and in the same order as the first $|T_{i,s}|$ messages of $T_{j,t}$.

Definition 3 (Matching conversations). *We say that π_i^s has a matching conversation to π_j^t , if*

- $T_{j,t}$ is a prefix of $T_{i,s}$ and π_i^s has sent the last message(s), or
- $T_{i,s}$ is a prefix of $T_{j,t}$ and π_j^t has sent the last message(s).

³ Note that we assume that learning identities of communication partners (which is necessary to determine the public-key used to perform authentication) is part of the protocol.

⁴ For details on when and how they are set in TLS, see the description in Section 3 and Figure 2.

⁵ Note, that the adversary does not ‘take control’ of oracles corresponding to a corrupted party. But he learns the long-term secret key, and can henceforth simulate these oracles.

Security of AKE protocols is now defined by requiring that (i) the protocol is a secure authentication protocol, and (ii) the protocol is a secure key-exchange protocol.

AKE Game. We formally capture this notion as a game, played between an adversary \mathcal{A} and a challenger \mathcal{C} . The challenger implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. At the beginning of the game, the challenger generates ℓ long-term key pairs (pk_i, sk_i) for all $i \in [\ell]$. The adversary receives the public keys pk_1, \dots, pk_ℓ as input. Now the adversary may start issuing **Send**, **Reveal** and **Corrupt** queries, as well as one **Test**-query. Finally, the adversary outputs a bit b' and terminates.

Definition 4. We say that an adversary (t, ϵ) -breaks an AKE protocol, if \mathcal{A} runs in time t , and at least one of the following two conditions holds:

1. When \mathcal{A} terminates, then with probability at least ϵ there exists an oracle π_i^s such that
 - π_i^s ‘accepts’ when \mathcal{A} issues its τ_0 -th query with partner $\Pi = j$, and
 - P_j is τ_j -corrupted with $\tau_0 < \tau_j$,⁶ and
 - there is no unique oracle π_j^t such that π_i^s has a matching conversation to π_j^t .

If an oracle π_i^s accepts in the above sense, then we say that π_i^s accepts maliciously.

2. When \mathcal{A} issues a **Test**-query to any oracle π_i^s and
 - \mathcal{A} does not issue a **Reveal**-query to π_i^s , nor to π_j^t such that π_i^s has a matching conversation to π_j^t (if such an oracle exists), and
 - π_i^s ‘accepts’ when \mathcal{A} issues its τ_0 -th query, and both parties P_i and P_j are τ_i - and τ_j -corrupted, respectively, with $\tau_0 < \tau_i, \tau_j$,⁷

then the probability that \mathcal{A} outputs b' which equals the bit b sampled by the **Test**-query satisfies

$$|\Pr[b = b'] - 1/2| \geq \epsilon.$$

We say that an AKE protocol is (t, ϵ) -secure, if there exists no adversary that (t, ϵ) -breaks it.

Remark 1. Note that the above definition even allows to corrupt oracles involved in the **Test**-session (of course only after the **Test**-oracle has reached **accept-state**, in order to exclude trivial attacks). Thus, protocols secure with respect to this definition provide *perfect forward secrecy*. Note also that we allow the ‘accepting’ oracle to be corrupted even *before* it reaches **accept-state**, which provides security against *key-compromise impersonation* attacks.

Now we can prove the security of a modified version of the TLS Handshake protocol. As discussed in the introduction, it is impossible to prove the full TLS Handshake protocol secure in any security model based on key-indistinguishability, like the model from Section 4, because the encryption and MAC of the

⁶ That is, P_j is not corrupted (i.e. τ -corrupted with $\tau = \infty$) when π_i^s ‘accepts’.

⁷ That is, neither party P_i nor P_j is corrupted when π_i^s ‘accepts’.

Finished messages provide a ‘check value’, that can be exploited by an adversary to determine the bit b chosen by the **Test**-query.

Therefore we consider a ‘truncated TLS’ protocol as in [20,21]. In this truncated version, we assume that the **Finished** messages are sent in clear, that is, neither encrypted nor authenticated by a MAC. More precisely, we modify the TLS protocol depicted in Figure 2 such that messages m_{11} and m_{13} contain only fin_H (instead of $\text{StE}.\text{Enc}(k_{\text{enc}}^H, \text{len}, H, \text{fin}_H, \text{st}_e)$), allowing us to prove security in the above model.

Theorem 1. *Let μ be the output length of PRF and let λ be the length of the nonces r_C and r_S . Assume that the pseudo-random function PRF is $(t, \epsilon_{\text{prf}})$ -secure, the signature scheme is $(t, \epsilon_{\text{sig}})$ -secure, the DDH-problem is $(t, \epsilon_{\text{ddh}})$ -hard in the group G used to compute the TLS premaster secret, and the PRF-ODH-problem is $(t, \epsilon_{\text{prfodh}})$ -hard with respect to G and PRF.*

Then for any adversary that $(t', \epsilon_{\text{ttls}})$ -breaks the truncated TLS-DHE protocol in the sense of Definition 4 with $t \approx t'$ holds that

$$\epsilon_{\text{ttls}} \leq 4 \cdot d\ell \left(\frac{d\ell}{2^\lambda} + \ell \cdot \epsilon_{\text{sig}} + \frac{5}{4} \cdot \epsilon_{\text{ddh}} + \frac{5}{2} \cdot \epsilon_{\text{prf}} + d\ell \left(\epsilon_{\text{prfodh}} + \epsilon_{\text{prf}} + \frac{1}{2^\mu} \right) \right).$$

Proof Sketch. Let us sketch the proof of Theorem 1, more details can be found in the full version [18]. We consider three types of adversaries:

1. Adversaries that succeed in making an oracle accept maliciously, such that the first oracle that does so is a **Client**-oracle (i.e., an oracle with $\rho = \text{Client}$). We call such an adversary a **Client**-adversary.
2. Adversaries that succeed in making an oracle accept maliciously, such that the first oracle that does so is a **Server**-oracle (i.e., an oracle with $\rho = \text{Server}$). We call such an adversary a **Server**-adversary.
3. Adversaries that do not succeed in making any oracle accept maliciously, but which answer the **Test**-challenge. We call such an adversary a **Test**-adversary.

We prove Theorem 1 by three lemmas. Lemma 1 bounds the probability ϵ_{client} that a **Client**-adversary succeeds, Lemma 2 bounds the probability ϵ_{server} that a **Server**-adversary succeeds, and Lemma 3 bounds the success probability ϵ_{ke} of a **Test**-adversary. Then we have $\epsilon_{\text{ttls}} \leq \epsilon_{\text{client}} + \epsilon_{\text{server}} + \epsilon_{\text{ke}}$.

Lemma 1. *For any adversary \mathcal{A} running in time $t' \approx t$, the probability that there exists an oracle π_i^s with $\rho = \text{Client}$ that accepts maliciously is at most*

$$\epsilon_{\text{client}} \leq d\ell \left(\frac{d\ell}{2^\lambda} + \ell \cdot \epsilon_{\text{sig}} + d\ell \left(\epsilon_{\text{prfodh}} + \epsilon_{\text{prf}} + \frac{1}{2^\mu} \right) \right)$$

where all quantities are defined as stated in Theorem 1.

Proof Sketch. We prove Lemma 1 in a sequence of games [5,23].

Game 0. This is the original security experiment.

Game 1. We add an abort condition. The challenger aborts, if throughout the game any oracle chooses a random nonce r_C or r_S which is not unique. Since nonces are chosen uniformly random, the collision probability is bounded by $(d\ell)^2 2^{-\mu}$. This abort condition ensures that any oracle that accepts with non-corrupted partner has a *unique* partner oracle.

Game 2. The challenger guesses an oracle $\pi_{i^*}^{s^*}$, and aborts if this oracle does not accept maliciously with $\rho = \text{Client}$. If there exists a maliciously accepting Client-oracle, then the guess is correct with probability $1/d\ell$.

Game 3. Next we want to ensure that $\pi_{i^*}^{s^*}$ receives as input exactly the Diffie-Hellman share T_S chosen by another oracle π_j^t (not by the adversary). Note that the respective party P_j must not be corrupted, as otherwise $\pi_{i^*}^{s^*}$ would not accept maliciously in the sense of Definition 4. The Diffie-Hellman share T_S is contained in the digital signature received by $\pi_{i^*}^{s^*}$, thus we can use the $(\epsilon_{\text{sig}}, t)$ -security of the signature scheme to ensure that the adversary can only forward T_S from π_j^t to $\pi_{i^*}^{s^*}$.

Game 4. In this game the challenger guesses upfront the oracle $\pi_{j^*}^{t^*}$ that chooses and signs the Diffie-Hellman share T_S received by $\pi_{i^*}^{s^*}$, and aborts if its guess is wrong. Again the guess is correct with probability at least $1/d\ell$.

Game 5. Now we are in a game where the challenger controls both Diffie-Hellman shares $T_C = g^{t_c}$ and $T_S = g^{t_s}$ chosen and received by $\pi_{i^*}^{s^*}$. A natural approach would be to use the DDH assumption now to replace the premaster-secret $pms = g^{t_c t_s}$ with an independent random value \widetilde{pms} , in order to be able to use the security of the $\text{PRF}(\widetilde{pms}, \cdot)$ as an argument in a following game to replace the master secret ms with an independent \widetilde{ms} .

However, unfortunately we cannot do this, as this would lead to a problem with the simulation of the fin_S -message sent by $\pi_{j^*}^{t^*}$ (we describe this issue in more detail in Section 6). Instead, we use the PRF-ODH-assumption to directly replace the master secret ms with an independent value \widetilde{ms} . We use the oracle provided by the PRF-ODH-assumption to simulate the fin_S message if necessary, which allows us to overcome the mentioned problem.

Game 6. In this game the challenger replaces the function $\text{PRF}(\widetilde{ms}, \cdot)$ with a truly random function. Note that \widetilde{ms} is an independent random value, thus we can use the security of the PRF to argue that this game is indistinguishable from Game 5.

Game 7. Finally, we use the fact that in this game a truly random function is used to verify the finished-message fin_S received by $\pi_{i^*}^{s^*}$, to conclude that $\pi_{i^*}^{s^*}$ accepts maliciously with probability at most $2^{-\mu}$.

Lemma 2. *For any adversary \mathcal{A} running in time $t' \approx t$, the probability that there exists an oracle π_i^s with $\rho = \text{Server}$ that accepts maliciously is at most*

$$\epsilon_{\text{server}} \leq d\ell \left(\frac{d\ell}{2^\lambda} + \ell \cdot \epsilon_{\text{sig}} + \epsilon_{\text{ddh}} + 2 \cdot \epsilon_{\text{prf}} + \frac{1}{2^\mu} \right)$$

where all quantities are defined as stated in Theorem 1.

Proof Sketch. The proof of Lemma 2 is very similar to the proof of Lemma 2, except that the problem with the simulation of the fin_S message does not occur in the case where we are dealing with Server-adversaries. Therefore we are able to base security in this case on the standard DDH assumption instead of the non-standard PRF-ODH assumption. (This is the reason why we consider Client- and Server-adversaries separately).

Lemma 3. *For any adversary \mathcal{A} running in time $t' \approx t$, the probability that \mathcal{A} answers the Test-challenge correctly is at most $1/2 + \epsilon_{\text{ke}}$ with*

$$\epsilon_{\text{ke}} \leq \epsilon_{\text{client}} + \epsilon_{\text{server}} + d\ell \cdot (\epsilon_{\text{ddh}} + 2 \cdot \epsilon_{\text{prf}}).$$

where $\epsilon_{\text{client}} + \epsilon_{\text{server}}$ is an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 4 (cf. Lemmas 1 and 2) and all other quantities are defined as stated in Theorem 1.

Proof Sketch. In order to prove Lemma 3, we first use the bounds derived in Lemmas 1 and 2 on the probability that there exists a Client- or Server-oracle that accepts maliciously. We then employ a very similar sequence of games as in the proofs of Lemmas 1 and 2. Recall that the keys in the real protocol are computed as

$$K_{\text{enc}}^{C \rightarrow S} \| K_{\text{enc}}^{S \rightarrow C} \| K_{\text{mac}}^{C \rightarrow S} \| K_{\text{mac}}^{S \rightarrow C} := \text{PRF}(ms, \text{label}_2 \| r_C \| r_S),$$

and in the proofs of Lemmas 1 and 2 we have first replaced ms with an independent value \tilde{ms} , and then the function $\text{PRF}(\tilde{ms}, \cdot)$ with a truly random function. Once we have reached this game, the adversary will *always* receive an independent key vector $K_{\text{enc}}^{C \rightarrow S} \| K_{\text{enc}}^{S \rightarrow C} \| K_{\text{mac}}^{C \rightarrow S} \| K_{\text{mac}}^{S \rightarrow C}$ as input, regardless of the bit b sampled for the Test-query. Thus, the adversary outputs its guess b' without receiving any information about b . This allows us to bound the success probability of the adversary in this final game as $\Pr[b' = b] = 1/2$.

Summing up probabilities from Lemmas 1 to 3, we obtain that

$$\begin{aligned} \epsilon_{\text{ttls}} &\leq \epsilon_{\text{client}} + \epsilon_{\text{server}} + \epsilon_{\text{ke}} \leq 2 \cdot (\epsilon_{\text{client}} + \epsilon_{\text{server}}) + d\ell \cdot (\epsilon_{\text{ddh}} + 2 \cdot \epsilon_{\text{prf}}) \\ &\leq 4 \cdot \max\{\epsilon_{\text{client}}, \epsilon_{\text{server}}\} + d\ell \cdot (\epsilon_{\text{ddh}} + 2 \cdot \epsilon_{\text{prf}}) \\ &\leq 4 \cdot d\ell \left(\frac{d\ell}{2^\lambda} + \ell \cdot \epsilon_{\text{sig}} + \epsilon_{\text{ddh}} + 2 \cdot \epsilon_{\text{prf}} + d\ell \left(\epsilon_{\text{prfodh}} + \epsilon_{\text{prf}} + \frac{1}{2^\mu} \right) \right) \\ &\quad + d\ell (\epsilon_{\text{ddh}} + 2 \cdot \epsilon_{\text{prf}}) \\ &= 4 \cdot d\ell \left(\frac{d\ell}{2^\lambda} + \ell \cdot \epsilon_{\text{sig}} + \frac{5}{4} \cdot \epsilon_{\text{ddh}} + \frac{5}{2} \cdot \epsilon_{\text{prf}} + d\ell \left(\epsilon_{\text{prfodh}} + \epsilon_{\text{prf}} + \frac{1}{2^\mu} \right) \right). \end{aligned}$$

5 ACCE Protocols

An *authenticated and confidential channel establishment* (ACCE) protocol is a protocol executed between two parties. The protocol consists of two phases, called the ‘pre-accept’ phase and the ‘post-accept’ phase.

Pre-accept phase. In this phase a ‘handshake protocol’ is executed. In terms of functionality this protocol is an AKE protocol as in Section 4, that is, both communication partners are mutually authenticated, and a session key k is established. However, it need not necessarily meet the security definition for AKE protocols (Definition 4). This phase ends, when both communication partners reach an `accept`-state.

Post-accept phase. This phase is entered, when both communication partners reach `accept`-state. In this phase data can be transmitted, encrypted and authenticated with key k .

The prime example for an ACCE protocol is TLS. Here, the pre-accept phase consists of the TLS Handshake protocol. In the post-accept phase encrypted and authenticated data is transmitted over the TLS Record Layer.

To define security of ACCE protocols, we combine the security model for authenticated key exchange from Section 4 with stateful length-hiding encryption in the sense of [22]. Technically, we provide a slightly modified execution environment that extends the types of queries an adversary may issue.

EXECUTION ENVIRONMENT. The execution environment is very similar to the model from Section 4, except for a few simple modifications. We extend the model such that in the post-accept phase an adversary is also able to ‘inject’ chosen-plaintexts by making an `Encrypt`-query, and chosen-ciphertexts by making a `Decrypt`-query. Moreover, each oracle π_i^s keeps as additional internal state a (randomly chosen) bit $b_i^s \leftarrow \{0, 1\}$, two counters u and v required for the security definition, and two state variables st_e and st_d for encryption and decryption with a stateful symmetric cipher. In the sequel we will furthermore assume that the key k consists of two different keys $k = (k_{\text{enc}}^\rho, k_{\text{dec}}^\rho)$ for encryption and decryption. Their order depends on the role $\rho \in \{\text{Client}, \text{Server}\}$ of oracle π_i^s . This is the case for TLS (see Section 3).

An adversary issue the following queries to the provided oracles.

- $\text{Send}^{\text{pre}}(\pi_i^s, m)$: This query is identical to the `Send`-query in the AKE model from Section 4, except that it replies with an error symbol \perp if oracle π_i^s has state $\Lambda = \text{accept}$. (`Send`-queries in `accept`-state are handled by the `Decrypt`-query below).
- $\text{Reveal}(\pi_i^s)$ and $\text{Corrupt}(P_i)$: These queries are identical to the corresponding queries in the AKE model from Section 4.
- $\text{Encrypt}(\pi_i^s, m_0, m_1, \text{len}, H)$: This query takes as input two messages m_0 and m_1 , length parameter len , and header data H . If $\Lambda \neq \text{accept}$ then π_i^s returns \perp . Otherwise, it proceeds as $\text{Encrypt}(m_0, m_1, \text{len}, H)$ depicted in Figure 1 with $k = k_{\text{enc}}^\rho$ and $b = b_i^s$ depending on the internal state of π_i^s .
- $\text{Decrypt}(\pi_i^s, C, H)$: This query takes as input a ciphertext C and header data H . If $\Lambda \neq \text{accept}$ then π_i^s returns \perp . Otherwise, it proceeds as $\text{Decrypt}(C, H)$ depicted in Figure 1 with $k = k_{\text{dec}}^\rho$ and $b = b_i^s$ depending on the internal state of π_i^s .

SECURITY DEFINITION. Security of ACCE protocols is defined by requiring that (i) the protocol is a secure authentication protocol (ii) in the post-accept phase

data is transmitted over an authenticated and confidential channel in the sense of Definition 2.

Again this notion is captured by a game, played between an adversary \mathcal{A} and a challenger \mathcal{C} . The challenger implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. At the beginning of the game, the challenger generates ℓ long-term key pairs (pk_i, sk_i) for all $i \in [\ell]$. The adversary receives the public keys pk_1, \dots, pk_ℓ as input. Now the adversary may start issuing `Send`, `Reveal`, `Corrupt`, `Encrypt`, and `Decrypt` queries. Finally, the adversary outputs a triple (i, s, b') and terminates.

Definition 5. We say that an adversary (t, ϵ) -breaks an ACCE protocol, if \mathcal{A} runs in time t , and at least one of the following two conditions holds:

1. When \mathcal{A} terminates, then with probability at least ϵ there exists an oracle π_i^s such that
 - π_i^s ‘accepts’ when \mathcal{A} issues its τ_0 -th query with partner $\Pi = j$, and
 - P_j is τ_j -corrupted with $\tau_0 < \tau_j$,⁸ and
 - there is no unique oracle π_j^t such that π_i^s has a matching conversation to π_j^t .
2. When \mathcal{A} terminates and outputs a triple (i, s, b') such that
 - π_i^s ‘accepts’ when \mathcal{A} issues its τ_0 -th query with intended partner $\Pi = j$, and P_j is τ_j -corrupted with $\tau_0 < \tau_j$,
 - \mathcal{A} did not issue a `Reveal`-query to π_i^s , nor to π_j^t such that π_i^s has a matching conversation to π_j^t (if such an oracle exists), and
then the probability that b' equals b_i^s is bounded by

$$|\Pr[b_i^s = b'] - 1/2| \geq \epsilon.$$

If an adversary \mathcal{A} outputs (i, s, b') such that $b' = b_i^s$ and the above conditions are met, then we say that \mathcal{A} answers the encryption-challenge correctly.

We say that an ACCE protocol is (t, ϵ) -secure, if there exists no adversary that (t, ϵ) -breaks it.

Remark 2. Note that the above definition even allows to corrupt the oracle π_i^s whose internal secret bit the attacker tries to determine. Of course this is only allowed after π_i^s has reached `accept`-state, in order to exclude trivial attacks. Thus, protocols secure with respect to this definition provide *perfect forward secrecy*. Note also that again we allow the ‘accepting’ oracle to be corrupted even before it reaches `accept`-state, which provides security against *key-compromise impersonation* attacks.

Relation to the AKE Security Definition from Section 4. Note that an ACCE protocol can be constructed in a two-step approach.

1. (AKE part) First an authenticated key-exchange (AKE) protocol is executed. This protocol guarantees the authenticity of the communication partner, and provides a cryptographically ‘good’ (i.e., for the attacker indistinguishable from random) session key.

⁸ That is, P_j is not corrupted (i.e. τ -corrupted with $\tau = \infty$) when π_i^s ‘accepts’.

2. (Symmetric part) The session key is then used in a symmetric encryption scheme providing integrity and confidentiality.

This modular approach is simple and generic, and therefore appealing. It can be shown formally that this two-step approach yields a secure ACCE protocol, if the ‘AKE part’ meets the security in the sense of Definition 4, and the ‘symmetric part’ consists of a suitable authenticated symmetric encryption scheme (e.g. secure according to Definition 2).

However, if the purpose of the protocol is the establishment of an authenticated confidential channel, then it is not necessary that the ‘AKE-part’ of the protocol provides full indistinguishability of *session keys*. It actually would suffice if *encrypted messages* are indistinguishable, and *cannot be altered* by an adversary. These requirements are strictly weaker than indistinguishability of keys in the sense of Definition 4, and thus easier to achieve (possibly from weaker hardness assumptions, or by more efficient protocols).

We stress that our ACCE definition is mainly motivated by the fact that security models based on key indistinguishability do not allow for a security analysis of full TLS, as detailed in the introduction. We do not want to propose ACCE as a new security notion for key exchange protocols, since it is very complex and the modular two-step approach approach seems more useful in general.

Theorem 2. *Let μ be the output length of PRF and let λ be the length of the nonces r_C and r_S . Assume that the pseudo-random function PRF is $(t, \epsilon_{\text{prf}})$ -secure, the signature scheme is $(t, \epsilon_{\text{sig}})$ -secure, the DDH-problem is $(t, \epsilon_{\text{ddh}})$ -hard in the group G used to compute the TLS premaster secret, and the PRF-ODH-problem is $(t, \epsilon_{\text{prfodh}})$ -hard with respect to G and PRF. Suppose that the stateful symmetric encryption scheme is $(t, \epsilon_{\text{sLHAE}})$ -secure.*

Then for any adversary that $(t', \epsilon_{\text{tls}})$ -breaks the TLS-DHE protocol in the sense of Definition 5 with $t \approx t'$ holds that

$$\epsilon_{\text{tls}} \leq 4d\ell \left(\frac{d\ell}{2^\lambda} + \ell\epsilon_{\text{sig}} + \frac{5}{4}\epsilon_{\text{ddh}} + \frac{5}{2}\epsilon_{\text{prf}} + \frac{1}{4}\epsilon_{\text{sLHAE}} + d\ell \left(\epsilon_{\text{prfodh}} + \epsilon_{\text{prf}} + \frac{1}{2^\mu} \right) \right).$$

Proof Sketch. The proof of Theorem 2 is very similar to the proof of Theorem 1. Instead of proving indistinguishability of keys, as in Lemma 3 from the proof of Theorem 1, we now have to consider indistinguishability of encrypted messages and authenticity of ciphertexts. We do this by employing the same sequence of games as in the proof of Lemma 3, except that we extend the proof by one game-hop at the end of the sequence of games.

Recall that in the proof of Lemma 3 the keys $K_{\text{enc}}^{C \rightarrow S} || K_{\text{enc}}^{S \rightarrow C} || K_{\text{mac}}^{C \rightarrow S} || K_{\text{mac}}^{S \rightarrow C}$, which determine the encryption and decryption keys $(k_{\text{enc}}^{\rho}, k_{\text{dec}}^{\rho})$ of the stateful encryption scheme, were replaced with independent random values. This allows us to extend the sequence of games by one final game, where the security of TLS-DHE is reduced to the sLHAE-security (in the sense of Definition 2) of the underlying stateful encryption scheme.

6 Security of TLS-DHE from Standard Assumptions

In this section we sketch why we had to make the PRF-ODH-assumption in the proof of Lemma 1 (and thus in Theorems 1 and 2), and why it seems unlikely that one can prove security based on standard DDH and a standard assumption on the PRF, if a security model allowing active adversaries and user corruptions is considered.

Suppose we are given a Client-adversary, that is, an adversary which always makes Client-oracle $C := \pi_i^s$ (i.e., π_i^s with $\rho = \text{Client}$) accept maliciously with intended partner $H = S$. Suppose we want to argue that the adversary is not able to forge the fin_S -message received by C (which we would have to, since the fin_S -message is the only message that cryptographically protects all messages previously received by π_i^s , and thus is required to ensure that π_i^s has a matching conversation to some other oracle), and that we want to assume only that the PRF is secure in the standard sense (see [18, Definition 3]). Then at some point in the proof we would have to replace the premaster secret computed by π_i^s as $pms = T_S^{t_C} = g^{t_C t_S}$ with an independent random value.

Note that in order to do so and to argue in the proof with indistinguishability, we must not know any of the exponents t_C and t_S in $T_C = g^{t_C}$ and $T_S = g^{t_S}$, as otherwise we can trivially distinguish the real $pms = g^{t_C t_S}$ from a random pms' . The problematic property of TLS-DHE is now, that an adversary may test whether the challenger ‘knows’ t_S , and then make Client-oracle π_i^s accept maliciously only if this holds. This works as follows.

1. The adversary establishes a communication between two oracles π_i^s (representing the client C) and π_j^t (representing the server S) by simply forwarding the messages m_1 and (m_2, \dots, m_6) between C and S .
2. Then C will respond with $(m_7, \dots, m_{11}) = (\text{cert}_C, T_C, \sigma_C, \text{flag}_{\text{enc}}, \text{fin}_C)$.⁹ This message is not forwarded.
3. Instead, the adversary corrupts some party $P^* \notin \{P_i, P_j\}$, and obtains the secret key sk^* of this party. Then it computes
 - (a) $T^* := g^{t^*} \bmod p$ for random $t^* \xleftarrow{\$} \mathbb{Z}_q$,
 - (b) $\sigma^* := \text{SIG.Sign}(sk^*; (r_C, r_S, T_S, T^*))$ using the corrupted key sk^* ,
 - (c) $ms^* := \text{PRF}(T_S^{t^*}, \text{label}_1 || r_C || r_S)$ using knowledge of t^* , and
 - (d) $\text{fin}_C^* := \text{PRF}(ms^*, m_1 || m_2 || (T^*, \sigma^*))$.

and sends $(m_7, \dots, m_{11}) = (\text{cert}_{C^*}, T^*, \sigma^*, \text{flag}_{\text{enc}}, \text{fin}_C^*)$ to S . Note that S cannot determine that its communication partner has changed, because any messages previously received by S were perfectly anonymous.

4. If S responds with a correct fin_S^* -message (note that the adversary is able to compute the key $pms^* := T_S^{t^*}$, since it ‘knows’ t^* , and thus is able to verify the validity of fin_S^*), then adversary concludes that the challenger ‘knows’ t_S and forges the required fin_S -message to make π_i^s accept without matching conversations. Otherwise the adversary aborts.

⁹ We consider truncated TLS-DHE here for simplicity, the same argument applies to TLS-DHE with encrypted **Finished**-messages, too.

Note that the above adversary is a valid, successful adversary in the real security experiment. It does not issue any **Reveal**-query and only one **Corrupt**-query to an unrelated party, such that the intended communication partner $\Pi = S$ of $C = \pi_i^s$ remains uncorrupted, but still it makes $C = \pi_i^s$ ‘accept’ and there is no oracle that C has a matching conversation to.

However, we explain why we will not be able to use this adversary in a simulated security experiment, where the challenger does not know the exponent t_S of $T_S = g^{t_S}$. Intuitively, the reason is that in this case the challenger would *first* have to compute the **Finished**-message fin_S^* , where

$$fin_S^* = \text{PRF}(ms, m_1 || \dots || m_3) \quad \text{and} \quad ms = \text{PRF}(T_S^{t^*}, \text{label}_1 || r_C || rs),$$

but ‘knowing’ neither $t_S = \log T_S$, nor $t^* = \log T^*$. This is the technical problem we are faced with, if we want to prove security under a standard assumption like DDH. Under the PRF-ODH-assumption, we can however use the given oracle to compute first ms , and from this the **Finished**-message fin_S^* .

Interestingly, the above technical problem does not appear if we consider only **Server**-adversaries (i.e., adversaries that make an oracle π_i^s accept maliciously with $\rho = \text{Server}$) instead. This is due to an asymmetry of the TLS-DHE Handshake protocol, see [18] for details.

One can circumvent the above problem, and thus base the security proof on the standard DDH assumption instead of PRF-ODH, if one considers a weaker security model where no **Corrupt**-queries are allowed (which however seems not adequate for the way how TLS-DHE is used on the Internet).

In [11] Canetti and Krawczyk describe a protocol called Σ_0 , which exhibits many similarities to the TLS-DHE Handshake protocol, but is provably secure under standard assumptions (in particular under DDH instead of PRF-ODH). We discuss why the subtle differences between Σ_0 and TLS-DHE are crucial in [18, Section 8]. We also note that one could, *in principle*, make TLS-DHE provably secure under standard assumptions, if one would modify it such that it becomes more similar to Σ_0 , which would allow to carry the security analysis of Σ_0 from [11] over to TLS-DHE. Of course it seems unrealistic that such substantial changes become accepted in practice.

7 Conclusion

We have shown that the core cryptographic protocol underlying TLS-DHE provides a secure establishment of confidential and authenticated channels. We can avoid the random oracle model, if we make a suitable assumption on the pseudo-random function. The goal of this work is to analyse TLS-DHE on the protocol layer. As common in cryptographic protocol analyses, we therefore have ignored implementational issues like error messages, which of course might also be used to break the security of the protocol. We leave it as an interesting open question to find an adequate approach for modeling such side-channels in complex scenarios like AKE involving many parties and parallel, sequential, and concurrent executions.

The whole TLS protocol suite is much more complex than the cryptographic protocol underlying TLS-DHE. It is very flexible, as it allows to negotiate cipher-suites at the beginning of the protocol, or to resume sessions using an abbreviated TLS Handshake. So clearly the security analysis of TLS is not finished yet, there are still many open questions. However, we consider this work as a strong indicator for the soundness of the TLS protocol framework. We believe that future revisions of the TLS standard should be guided by provable security – ideally in the standard model.

Acknowledgements. We would like to thank Dennis Hofheinz, Kenny Paterson, Zheng Yang, and the anonymous referees for helpful comments and discussions.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Langley, A., Google Security Team: Protecting data for the long term with forward secrecy, <http://googleonlinesecurity.blogspot.co.uk/2011/11/protecting-data-for-long-term-with.html>
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Pneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
4. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
5. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
6. Blake-Wilson, S., Johnson, D., Menezes, A.: Key Agreement Protocols and their Security Analysis. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997)
7. Bleichenbacher, D.: Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
8. Brzuska, C., Fischlin, M., Smart, N.P., Warinschi, B., Williams, S.: Less is more: Relaxed yet composable security notions for key exchange. Cryptology ePrint Archive, Report 2012/242 (2012), <http://eprint.iacr.org/>
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, pp. 136–145. IEEE Computer Society Press (October 2001)
10. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
11. Canetti, R., Krawczyk, H.: Security Analysis of IKE’s Signature-Based Key-Exchange Protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002), <http://eprint.iacr.org/2002/120/>

12. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
13. Cremers, C.J.F.: Session-state Reveal Is Stronger Than Ephemeral Key Reveal: Attacking the NAXOS Authenticated Key Exchange Protocol. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 20–33. Springer, Heidelberg (2009)
14. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard) (January 1999); Obsoleted by RFC 4346, updated by RFCs 3546, 5746
15. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard) (April 2006); Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746
16. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (August 2008); Updated by RFCs 5746, 5878
17. Gajek, S., Manulis, M., Pereira, O., Sadeghi, A.-R., Schwenk, J.: Universally Composable Security Analysis of TLS. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 313–327. Springer, Heidelberg (2008)
18. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the Standard Model (full version). Cryptology ePrint Archive, Report 2011/219 (2011) (revised 2012), <http://eprint.iacr.org/2011/219>
19. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
20. Morrissey, P., Smart, N.P., Warinschi, B.: A Modular Security Analysis of the TLS Handshake Protocol. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 55–73. Springer, Heidelberg (2008)
21. Morrissey, P., Smart, N.P., Warinschi, B.: The TLS Handshake protocol: A modular analysis. Journal of Cryptology 23(2), 187–223 (2010)
22. Paterson, K.G., Ristenpart, T., Shrimpton, T.: Tag Size *Does* Matter: Attacks and Proofs for the TLS Record Protocol. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 372–389. Springer, Heidelberg (2011)
23. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (November 2004)
24. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 protocol. In: Proceedings of the Second USENIX Workshop on Electronic Commerce, pp. 29–40. USENIX Association (1996)

Semantic Security for the Wiretap Channel

Mihir Bellare¹, Stefano Tessaro², and Alexander Vardy³

¹ Department of Computer Science & Engineering,
University of California San Diego
cseweb.ucsd.edu/~mihir/

² CSAIL, Massachusetts Institute of Technology
people.csail.mit.edu/tessaro/

³ Department of Electrical & Computer Engineering,
University of California San Diego
<http://www.ece.ucsd.edu/~avardy/>

Abstract. The wiretap channel is a setting where one aims to provide information-theoretic privacy of communicated data based solely on the assumption that the channel from sender to adversary is “noisier” than the channel from sender to receiver. It has developed in the Information and Coding (I&C) community over the last 30 years largely divorced from the parallel development of modern cryptography. This paper aims to bridge the gap with a cryptographic treatment involving advances on two fronts, namely definitions and schemes. On the first front (definitions), we explain that the mis-r definition in current use is weak and propose two alternatives: mis (based on mutual information) and ss (based on the classical notion of semantic security). We prove them equivalent, thereby connecting two fundamentally different ways of defining privacy and providing a new, strong and well-founded target for constructions. On the second front (schemes), we provide the first explicit scheme with all the following characteristics: it is proven to achieve both security (ss and mis, not just mis-r) and decodability; it has optimal rate; and both the encryption and decryption algorithms are proven to be polynomial-time.

1 Introduction

The wiretap channel is a setting where one aims to obtain information-theoretic privacy under the sole assumption that the channel from sender to adversary is “noisier” than the channel from sender to receiver. Introduced by Wyner, Csiszár and Körner in the late seventies [41,14], it has developed in the Information and Coding (I&C) community largely divorced from the parallel development of modern cryptography. This paper aims to bridge the gap with a cryptographic treatment involving advances on two fronts.

The first is *definitions*. We explain that the security definition in current use, that we call mis-r (mutual-information security for random messages) is weak and insufficient to provide security of applications. We suggest strong, new definitions. One, that we call mis (mutual-information security), is an extension of mis-r and thus rooted in the I&C tradition and intuition. Another, semantic

security (ss), adapts the cryptographic “gold standard” emanating from [19] and universally accepted in the cryptographic community. We prove the two equivalent, thereby connecting two fundamentally different ways of defining privacy and providing a new, strong and well-founded target for constructions.

The second is *schemes*. Classically, the focus of the I&C community has been proofs of existence of mis-r schemes of optimal rate. The proofs are non-constructive so that the schemes may not even be explicit let alone polynomial time. Recently, there has been progress towards explicit mis-r schemes [30,29,21]. We take this effort to its full culmination by providing the first explicit construction of a scheme with all the following characteristics: it is proven to achieve security (not just mis-r but ss and mis) over the adversary channel; it is proven to achieve decodability over the receiver channel; it has optimal rate; and both the encryption and decryption algorithms are proven to be polynomial-time.

Today the possibility of realizing the wiretap setting in wireless networks is receiving practical attention and fueling a fresh look at this area. Our work helps guide the choice of security goals and schemes. Let us now look at all this in more detail.

THE WIRETAP MODEL. The setting is depicted in Fig. 1. The sender applies to her message M a randomized encryption algorithm $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ to get what we call the *sender ciphertext* $X \leftarrow_s \mathcal{E}(M)$. This is transmitted to the receiver over the receiver channel ChR so that the latter gets a *receiver ciphertext* $Y \leftarrow_s ChR(X)$ which he decrypts via algorithm \mathcal{D} to recover the message. The adversary’s wiretap is modeled as another channel ChA and she accordingly gets an *adversary ciphertext* $Z \leftarrow_s ChA(X)$ from which she tries to glean whatever she can about the message.

A *channel* is a randomized function specified by a transition probability matrix W where $W[x, y]$ is the probability that input x results in output y . Here x, y are strings. The canonical example is the Binary Symmetric Channel BSC_p with crossover probability $p \leq 1/2$ taking a binary string x of any length and returning the string y of the same length formed by flipping each bit of x independently with probability p . For concreteness and simplicity of exposition we will often phrase discussions in the setting where ChR, ChA are BSCs with crossover probabilities $p_R, p_A \leq 1/2$ respectively, but our results apply in much greater generality. In this case the assumption that ChA is “noisier” than ChR corresponds to the assumption that $p_R < p_A$. This is the only assumption made: the adversary is computationally unbounded, and the scheme is keyless, meaning sender and receiver are not assumed to a priori share any information not known to the adversary.

The setting now has a literature encompassing hundreds of papers. (See the survey [28] or the book [6].) Schemes must satisfy two conditions, namely *decoding* and *security*. The *decoding* condition asks that the scheme provide error-correction over the receiver channel, namely the limit as $k \rightarrow \infty$ of the maximum, over all M of length m , of $\Pr[\mathcal{D}(ChR(\mathcal{E}(M))) \neq M]$, is 0, where k is an underlying security parameter of which m, c are functions. The original security condition of [41] was that $\lim_{k \rightarrow \infty} \mathbf{I}(M; ChA(\mathcal{E}(M))/m = 0$ where the message random variable M is uniformly distributed over $\{0, 1\}^m$ and $\mathbf{I}(M; Z) = H(M) - H(M | Z)$ is

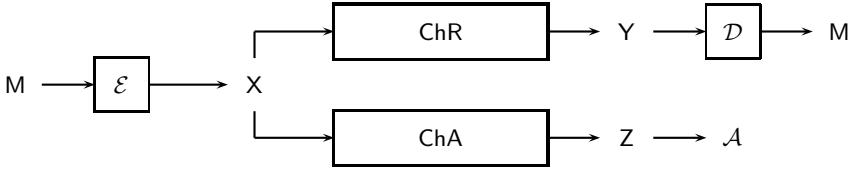


Fig. 1. Wiretap channel model. See text for explanations.

the mutual information. This was critiqued by [31,32] who put forth the stronger security condition now in use, namely that $\lim_{k \rightarrow \infty} I(M; ChA(\mathcal{E}(M))) = 0$, the random variable M continuing to be uniformly distributed over $\{0, 1\}^m$. The *rate* $\text{Rate}(\mathcal{E})$ of a scheme is m/c .

The literature has focused on determining the maximum possible rate. (We stress that the maximum is over all possible schemes, not just ones that are explicitly given or polynomial time.) Shannon's seminal result [37] says that if we ignore security and merely consider achieving the decoding condition then this optimal rate is the receiver channel capacity, which in the BSC case is $1 - h_2(p_R)$ where h_2 is the binary entropy function $h_2(p) = -p \lg(p) - (1-p) \lg(1-p)$. He gave non-constructive proofs of existence of schemes meeting capacity. Coming in with this background and the added security condition, it was natural for the wiretap community to follow Shannon's lead and begin by asking what is the maximum achievable rate subject to both the security and decoding conditions. This optimal rate is called the secrecy capacity and, in the case of BSCs, equals the difference $(1 - h_2(p_R)) - (1 - h_2(p_A)) = h_2(p_A) - h_2(p_R)$ in capacities of the receiver and adversary channels. Non-constructive proofs of the existence of schemes with this optimal rate were given in [41,14,7]. Efforts to obtain explicit, secure schemes of optimal rate followed [40,33,30,29,21].

CONTEXT. Practical interest in the wiretap setting is escalating. Its proponents note two striking benefits over conventional cryptography: (1) no computational assumptions, and (2) no keys and hence no key distribution. Item (1) is attractive to governments who are concerned with long-term security and worried about quantum computing. Item (2) is attractive in a world where vulnerable, low-power devices are proliferating and key-distribution and key-management are unsurmountable obstacles to security. The practical challenge is to realize a secrecy capacity, meaning ensure by physical means that the adversary channel is noisier than the receiver one. The degradation with distance of radio communication signal quality is the basis of several approaches being investigated for wireless settings. Government-sponsored Ziva Corporation [42] is using optical techniques to build a receiver channel in such a way that wiretapping results in a degraded channel. A program called Physical Layer Security aimed at practical realization of the wiretap channel is the subject of books [6] and conferences [24]. All this activity means that schemes are being sought for implementation. If so, we need privacy definitions that yield security in applications, and we need constructive results yielding practical schemes achieving privacy under these definitions. This is what we aim to supply.

DEFINITIONS. A security *metric* xs associates to encryption function $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ and adversary channel ChA a number $\mathbf{Adv}^{\text{xs}}(\mathcal{E}; \text{ChA})$ that measures the maximum “advantage” of an adversary in breaking the scheme under metric xs . For example, the metric underlying the current, above-mentioned security condition is $\mathbf{Adv}^{\text{mis-r}}(\mathcal{E}; \text{ChA}) = \mathbf{I}(M; \text{ChA}(\mathcal{E}(M)))$ where M is uniformly distributed over $\{0, 1\}^m$. We call this the mis-r (mutual-information security for random messages) metric because messages are assumed to be random. From the cryptographic perspective, this is extraordinarily weak, for we know that real messages are not random. (They may be files, votes or any type of structured data, often with low entropy. Contrary to a view in the I&C community, compression does *not* render data random, as can be seen from the case of votes, where the message space has very low entropy.) This leads us to suggest a stronger metric that we call *mutual-information security*, defined via $\mathbf{Adv}^{\text{mis}}(\mathcal{E}; \text{ChA}) = \max_M \mathbf{I}(M; \text{ChA}(\mathcal{E}(M)))$ where the maximum is over all random variables M over $\{0, 1\}^m$, regardless of their distribution.

At this point, we have a legitimate metric, but how does it capture privacy? The intuition is that it is measuring the difference in the number of bits required to encode the message before and after seeing the ciphertext. This intuition is alien to cryptographers, whose metrics are based on much more direct and usage-driven privacy requirements. Cryptographers understand since [19] that encryption must hide all partial information about the message, meaning the adversary should have little advantage in computing a function of the message given the ciphertext. (Examples of partial information about a message include its first bit or even the XOR of the first and second bits.) The mis-r and mis metrics ask for nothing like this and are based on entirely different intuition. We extend Goldwasser and Micali’s semantic security [19] definition to the wiretap setting, defining $\mathbf{Adv}^{\text{ss}}(\mathcal{E}; \text{ChA})$ as

$$\max_{f, M} \left(\max_{\mathcal{A}} \Pr[\mathcal{A}(\text{ChA}(\mathcal{E}(M))) = f(M)] - \max_{\mathcal{S}} \Pr[\mathcal{S}(m) = f(M)] \right).$$

Within the parentheses is the maximum probability that an adversary \mathcal{A} , given the adversary ciphertext, can compute the result of function f on the message, minus the maximum probability that a simulator \mathcal{S} can do the same given only the length of the message. We also define a distinguishing security (ds) metric as an analog of indistinguishability [19] via

$$\mathbf{Adv}^{\text{ds}}(\mathcal{E}; \text{ChA}) = \max_{\mathcal{A}, M_0, M_1} 2 \Pr[\mathcal{A}(M_0, M_1, \text{ChA}(\mathcal{E}(M_b))) = b] - 1$$

where challenge bit b is uniformly distributed over $\{0, 1\}$ and the maximum is over all m -bit messages M_0, M_1 and all adversaries \mathcal{A} . For any metric xs , we say \mathcal{E} provides XS-security over ChA if $\lim_{k \rightarrow \infty} \mathbf{Adv}^{\text{xs}}(\mathcal{E}; \text{ChA}) = 0$.

RELATIONS. The mutual information between message and ciphertext, as measured by mis, is, as noted above, the change in the number of bits needed to encode the message created by seeing the ciphertext. It is not clear why this should measure privacy in the sense of semantic security. Yet we are able to show that

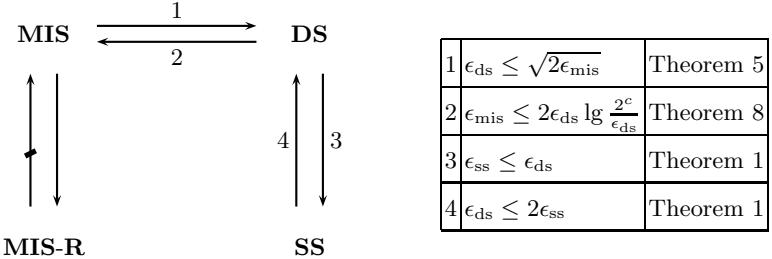


Fig. 2. Relations between notions. An arrow $\mathbf{A} \rightarrow \mathbf{B}$ is an implication, meaning every scheme that is \mathbf{A} -secure is also \mathbf{B} -secure, while a barred arrow $\mathbf{A} \not\rightarrow \mathbf{B}$ is a separation, meaning that there is a \mathbf{A} -secure scheme that is not \mathbf{B} -secure. If $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ is the encryption algorithm and ChA the adversary channel, we let $\epsilon_{xs} = \mathbf{Adv}^{\text{xs}}(\mathcal{E}; \text{ChA})$. The table then shows the quantitative bounds underlying the annotated implications.

mutual-information security and semantic security are equivalent, meaning an encryption scheme is MIS-secure if and only if it is SS-secure. Fig. 2 summarizes this and other relations we establish that between them settle all possible relations. The equivalence between SS and DS is the information-theoretic analogue of the corresponding well-known equivalence in the computational setting [19,3]. As there, however, it brings the important benefit that we can now work with the technically simpler DS. We then show that MIS implies DS by reducing to Pinsker's inequality. We show conversely that DS implies MIS via a general relation between mutual information and statistical distance. As Fig. 2 indicates, the asymptotic relations are all underlain by concrete quantitative and polynomial relations between the advantages. On the other hand, we show that in general MIS-R does not imply MIS, meaning the former is strictly weaker than the latter. We do this by exhibiting an encryption function \mathcal{E} and channel ChA such that \mathcal{E} is MIS-R-secure relative to ChA but MIS-insecure relative to ChA . Furthermore we do this for the case that ChA is a BSC.

OUR SCHEME. We provide the first explicit scheme with all the following characteristics over a large class of adversary and receiver channels including BSCs: (1) It is DS (hence SS, MIS and MIS-R) secure (2) It is proven to satisfy the decoding condition (3) It has optimal rate (4) It is fully polynomial time, meaning both encryption and decryption algorithms run in polynomial time (5) the errors in the security and decoding conditions do not just vanish in the limit but at an exponential rate. Our scheme is based on three main ideas: (1) the use of *invertible* extractors (2) analysis via smooth min-entropy, and (3) a (surprising) result saying that for certain types of schemes, security on random messages implies security on all messages.

Recall that the secrecy capacity is the optimal rate for MIS-R schemes meeting the decoding condition and in the case of BSCs it equals $h_2(p_A) - h_2(p_R)$. Since DS-security is stronger than MIS-R security, the optimal rate could in principle be smaller. Perhaps surprisingly, it isn't: for a broad class of channels including

symmetric channels, the optimal rate is the same for DS and MIS-R security. This follows by applying our above-mentioned result showing that MIS-R implies MIS for certain types of schemes and channels, to known results on achieving the secrecy capacity for MIS-R. Thus, when we say, above, that our scheme achieves optimal rate, this rate is in fact the secrecy capacity.

A common misconception is to think that privacy and error-correction may be completely de-coupled, meaning one would first build a scheme that is secure when the receiver channel is noiseless and then add an ECC on top to meet the decoding condition with a noisy receiver channel. This does not work because the error-correction helps the adversary by reducing the noise over the adversary channel. The two requirements do need to be considered together. Nonetheless, our approach is modular, combining (invertible) extractors with existing ECCs in a blackbox way. As a consequence, any ECC of sufficient rate may be used. This is an attractive feature of our scheme from the practical perspective. In addition our scheme is simple and efficient.

Our claims (proven DS-security and decoding with optimal rate) hold not only for BSCs but for a wide range of receiver and adversary channels.

A CONCRETE INSTANTIATION. As a consequence of our general paradigm, we prove that the following scheme achieves secrecy capacity for the BSC setting with $p_R < p_A \leq 1/2$. For any ECC $\mathsf{E}: \{0,1\}^k \rightarrow \{0,1\}^n$ such that $k \approx (1 - h_2(p_R)) \cdot n$ (such ECCs can be built e.g. from polar codes [2] or from concatenated codes [18]) and a parameter $t \geq 1$, our encryption function \mathcal{E} , on input an m -bit message M , where $m = b \cdot t$ and $b \approx (h_2(p_A) - h_2(p_R)) \cdot n$, first selects a random k -bit string $A \neq 0^k$ and t random $(k-b)$ -bit strings $R[1], \dots, R[t]$. It then splits M into t b -bit blocks $M[1], \dots, M[t]$, and outputs

$$\mathcal{E}(M) = \mathsf{E}(A) \parallel \mathsf{E}(A \odot (M[1] \parallel R[1])) \parallel \cdots \parallel \mathsf{E}(A \odot (M[t] \parallel R[t])),$$

where \odot is multiplication of k -bit strings interpreted as elements of the extension field $\text{GF}(2^k)$.

RELATED WORK. This paper was formed by merging [5,4] which together function as the full version of this paper. We refer there for all proofs omitted from this paper and also for full and comprehensive surveys of the large body of work related to wiretap security, and more broadly, to information-theoretically secure communication in a noisy setup. Here we discuss the most related work.

Relations between entropy- and distance-based security metrics have been explored in settings other than the wiretap one, using techniques similar to ours [13,7,15], the last in the context of statistically-private commitment. Iwamoto and Ohta [25] relate different notions of indistinguishability for statistically secure symmetric encryption. In the context of key-agreement in the wiretap setting (a simpler problem than ours) Csiszár [13] relates MIS-R and RDS, the latter being DS for random messages.

Wyner's syndrome coding approach [41] and extensions by Cohen and Zémor [9,10] only provide weak security. Hayashi and Matsumoto [21] replace the matrix-multiplication in these schemes by a universal hash function and show MIS-R security of their scheme. Their result could be used to obtain an alternative

to the proof of RDS security used in our scheme for the common case where the extractor is obtained from a universal hash function. However, the obtained security bound is not explicit, making their result unsuitable to applications. Moreover, our proof also yields as a special case a cleaner proof for the result of [21].

Syndrome coding could be viewed as a special case of coset coding which is based on an inner code and outer code. Instantiations of this approach have been considered in [40,33,38] using LDPC codes, but polynomial-time decoding is possible only if the adversary channel is a binary erasure channel or the receiver channel is noiseless.

Mahdavifar and Vardy [29] and Hof and Shamai [23] (similar ideas also appeared in [26,1]) use polar codes [2] to build encryption schemes for the wiretap setting with binary-input symmetric channels. However, these schemes only provide weak security. The full version [30] of [29] provides a variant of the scheme achieving MIS-R security. They use results of the present paper (namely our above-mentioned result that MIS-R implies MIS for certain schemes, whose proof they re-produce for completeness), to conclude that their scheme is also MIS-secure. However there is no proof that decryption (decoding) is possible in their scheme, even in principle let alone in polynomial time. Also efficient generation of polar codes is an open research direction with first results only now appearing [39], and hence relying on this specific code family may be somewhat problematic. Our solution, in contrast, works for arbitrary codes.

As explained in [5], fuzzy extractors [17] can be used to build a DS-secure scheme with polynomial-time encoding and decoding, but the rate of this scheme is far from optimal and the approach is inherently limited to low-rate schemes. We note that (seedless) invertible extractors were previously used in [8] within schemes for the “wiretap channel II” model [34], where the adversary (adaptively) erases ciphertext bits. In contrast to our work, only random-message security was considered in [8].

2 Preliminaries

BASIC NOTATION AND DEFINITIONS. “PT” stands for “polynomial-time.” If s is a binary string then $s[i]$ denotes its i -th bit and $|s|$ denotes its length. If S is a set then $|S|$ denotes its size. If x is a real number then $|x|$ denotes its absolute value. A function $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$ is *linear* if $f(x \oplus y) = f(x) \oplus f(y)$ for all $x, y \in \{0, 1\}^m$. A probability distribution is a function P that associates to each x a probability $P(x) \in [0, 1]$. The support $\text{SUPP}(P)$ is the set of all x such that $P(x) > 0$. All probability distributions in this paper are discrete. Associate to random variable X and event E the probability distributions $P_X, P_{X|E}$ defined for all x by $P_X(x) = \Pr[X = x]$ and $P_{X|E}(x) = \Pr[X = x | E]$. We denote by $\lg(\cdot)$ the logarithm in base two, and by $\ln(\cdot)$ the natural logarithm. We adopt standard conventions such as $0 \lg 0 = 0 \lg \infty = 0$ and $\Pr[E_1|E_2] = 0$ when $\Pr[E_2] = 0$. The function $h : [0, 1] \rightarrow [0, 1]$ is defined by $h(x) = -x \lg x$. The (Shannon) entropy of probability distribution P

is defined by $\mathbf{H}(P) = \sum_x h(P(x))$ and the statistical difference between probability distributions P, Q is defined by $\mathbf{SD}(P; Q) = 0.5 \cdot \sum_x |P(x) - Q(x)|$. If X, Y are random variables the (Shannon) entropy is defined by $\mathbf{H}(X) = \mathbf{H}(P_X) = \sum_x h(P_X(x))$. The conditional entropy is defined via $\mathbf{H}(X|Y = y) = \sum_x h(P_{X|Y=y}(x))$ and $\mathbf{H}(X|Y) = \sum_y P_Y(y) \cdot \mathbf{H}(X|Y = y)$. The mutual information is defined by $\mathbf{I}(X; Y) = \mathbf{H}(X) - \mathbf{H}(X|Y)$. The statistical or variational distance between random variables X_1, X_2 is $\mathbf{SD}(X_1; X_2) = \mathbf{SD}(P_{X_1}; P_{X_2}) = 0.5 \cdot \sum_x |\Pr[X_1 = x] - \Pr[X_2 = x]|$. The min-entropy of random variable X is $\mathbf{H}_\infty(X) = -\lg(\max_x \Pr[X = x])$ and if Z is also a random variable the conditional min-entropy is $\mathbf{H}_\infty(X|Z) = -\lg(\sum_z \Pr[Z = z] \max_x \Pr[X = x|Z = z])$.

TRANSFORMS, CHANNELS AND ALGORITHMS. We say that T is a transform with domain D and range R , written $T: D \rightarrow R$, if $T(x)$ is a random variable over R for every $x \in D$. Thus, T is fully specified by a sequence $P = \{P_x\}_{x \in D}$ of probability distributions over R , where $P_x(y) = \Pr[T(x) = y]$ for all $x \in D$ and $y \in R$. We call P the distribution associated to T . This distribution can be specified by a $|D|$ by $|R|$ transition probability matrix W defined by $W[x, y] = P_x(y)$. A channel is simply a transform. This is a very general notion of a channel but it does mean channels are memoryless in the sense that two successive transmissions over the same channel are independent random variables. The transition probability matrix representation is the most common one in this case. A (randomized) algorithm is also a transform. Finally, an adversary too is a transform, and so is a simulator.

If $T: \{0, 1\} \rightarrow R$ is a transform we may apply it to inputs of any length. The understanding is that T is applied independently to each bit of the input. For example BSC_p , classically defined as a 1-bit channel, is here viewed as taking inputs of arbitrary length and flipping each bit independently with probability p . Similarly, we apply a transform $T: \{0, 1\}^l \rightarrow R$ to any input whose length is divisible by l . We say that a transform $T: D \rightarrow R$ with transition matrix W is *symmetric* if there exists a partition of the range as $R = R_1 \cup \dots \cup R_n$ such that for all i the sub-matrix $W_i = W[\cdot, R_i]$ induced by the columns in R_i is strongly symmetric, i.e., all rows of W_i are permutations of each other, and all columns of W_i are permutations of each other.

3 Security Metrics and Relations

ENCRYPTION FUNCTIONS AND SCHEMES. An *encryption function* is a transform $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ where m is the message length and c is the sender ciphertext length. The *rate* of \mathcal{E} is $\mathbf{Rate}(\mathcal{E}) = m/c$. If $\text{ChR}: \{0, 1\}^c \rightarrow \{0, 1\}^d$ is a receiver channel then a *decryption function* for \mathcal{E} over ChR is a transform $\mathcal{D}: \{0, 1\}^d \rightarrow \{0, 1\}^m$ whose decryption error $\mathbf{DE}(\mathcal{E}; \mathcal{D}; \text{ChR})$ is defined as the maximum, over all $M \in \{0, 1\}^m$, of $\Pr[\mathcal{D}(\text{ChR}(\mathcal{E}(M))) \neq M]$.

An *encryption scheme* $\bar{\mathcal{E}} = \{\mathcal{E}_k\}_{k \in \mathbb{N}}$ is a family of encryption functions where $\mathcal{E}_k: \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{c(k)}$ for functions $m, c: \mathbb{N} \rightarrow \mathbb{N}$ called the message length and sender ciphertext length of the scheme. The *rate* of $\bar{\mathcal{E}}$ is the

function $\mathbf{Rate}_{\overline{\mathcal{E}}}: \mathbb{N} \rightarrow \mathbb{R}$ defined by $\mathbf{Rate}_{\overline{\mathcal{E}}}(k) = \mathbf{Rate}(\mathcal{E}_k)$ for all $k \in \mathbb{N}$. Suppose $\overline{\text{ChR}} = \{\text{ChR}_k\}_{k \in \mathbb{N}}$ is a family of receiver channels where $\text{ChR}_k: \{0, 1\}^{c(k)} \rightarrow \{0, 1\}^{d(k)}$. Then a *decryption scheme* for $\overline{\mathcal{E}}$ over $\overline{\text{ChR}}$ is a family $\overline{\mathcal{D}} = \{\mathcal{D}_k\}_{k \in \mathbb{N}}$ where $\mathcal{D}_k: \{0, 1\}^{d(k)} \rightarrow \{0, 1\}^{m(k)}$ is a decryption function for \mathcal{E}_k over ChR_k . We say that $\overline{\mathcal{D}}$ is a correct decryption scheme for $\overline{\mathcal{E}}$ relative to $\overline{\text{ChR}}$ if the limit as $k \rightarrow \infty$ of $\mathbf{DE}(\mathcal{E}_k; \mathcal{D}_k; \text{ChR}_k)$ is 0. We say that encryption scheme $\overline{\mathcal{E}}$ is decryptable relative to ChR if there exists a correct decryption scheme for $\overline{\mathcal{E}}$ relative to $\overline{\text{ChR}}$. A family $\{\mathcal{S}_k\}_{k \in \mathbb{N}}$ (eg. an encryption or decryption scheme) is PT if there is a polynomial time computable function which on input 1^k (the unary representation of k) and x returns $\mathcal{S}_k(x)$. Our constructs will provide PT encryption and decryption schemes.

SECURITY METRICS. Let $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ be an encryption function and let $\text{ChA}: \{0, 1\}^c \rightarrow \{0, 1\}^d$ be an adversary channel. Security depends only on these, not on the receiver channel. We now define semantic security (ss), distinguishing security (ds), random mutual-information security (mis-r) and mutual information security (mis). For each type of security $\text{xs} \in \{\text{ss}, \text{ds}, \text{mis-r}, \text{mis}\}$, we associate to $\mathcal{E}; \text{ChA}$ a real number $\mathbf{Adv}^{\text{xs}}(\mathcal{E}; \text{ChA})$. The smaller this number, the more secure is $\mathcal{E}; \text{ChA}$ according to the metric in question. The security of an encryption function is quantitative, as captured by the advantage. We might measure it in bits, saying that $\mathcal{E}; \text{ChA}$ has s bits of xs -security if $\mathbf{Adv}^{\text{xs}}(\mathcal{E}; \text{ChA}) \leq 2^{-s}$. A qualitative definition of “secure,” meaning one under which something is secure or not secure, may only be made asymptotically, meaning for schemes. We say encryption scheme $\overline{\mathcal{E}} = \{\mathcal{E}_k\}_{k \in \mathbb{N}}$ is XS-secure relative to $\overline{\text{ChA}} = \{\text{ChA}_k\}_{k \in \mathbb{N}}$ if $\lim_{k \rightarrow \infty} \mathbf{Adv}^{\text{xs}}(\mathcal{E}_k; \text{ChA}_k) = 0$. This does not mandate any particular rate at which the advantage should vanish, but in our constructions this rate is exponentially vanishing with k . We define the ss advantage $\mathbf{Adv}^{\text{ss}}(\mathcal{E}; \text{ChA})$ as

$$\max_{f, \mathbf{M}} \left(\max_{\mathcal{A}} \Pr[\mathcal{A}(\text{ChA}(\mathcal{E}(\mathbf{M}))) = f(\mathbf{M})] - \max_{\mathcal{S}} \Pr[\mathcal{S}(\mathbf{m}) = f(\mathbf{M})] \right). \quad (1)$$

Here f is a transform with domain $\{0, 1\}^m$ that represents partial information about the message. Examples include $f(M) = M$ or $f(M) = M[1]$ or $f(M) = M[1] \oplus \dots \oplus M[m]$, where $M[i]$ is the i -th bit of M . But f could be a much more complex function, and could even be randomized. The adversary’s goal is to compute $f(M)$ given an adversary ciphertext $\text{ChA}(\mathcal{E}(\mathbf{M}))$ formed by encrypting message \mathbf{M} . The probability that it does this is $\Pr[\mathcal{A}(\text{ChA}(\mathcal{E}(\mathbf{M}))) = f(\mathbf{M})]$, then maximized over all adversaries \mathcal{A} to achieve strategy independence. We then subtract the a priori probability of success, meaning the maximum possible probability of computing $f(M)$ if you are not given the adversary ciphertext. Finally, the outer max over all f, \mathbf{M} ensures that the metric measures the extent to which *any* partial information leaks regardless of message distribution. We define the distinguishing advantage via

$$\mathbf{Adv}^{\text{ds}}(\mathcal{E}; \text{ChA}) = \max_{\mathcal{A}, M_0, M_1} 2 \Pr[\mathcal{A}(M_0, M_1, \text{ChA}(\mathcal{E}(M_b))) = b] - 1 \quad (2)$$

$$= \max_{M_0, M_1} \mathbf{SD}(\text{ChA}(\mathcal{E}(M_0)); \text{ChA}(\mathcal{E}(M_1))). \quad (3)$$

In Eq. (2), $\Pr[\mathcal{A}(M_0, M_1, \text{ChA}(\mathcal{E}(M_b))) = b]$ is the probability that adversary \mathcal{A} , given m -bit messages M_0, M_1 and an adversary ciphertext emanating from M_b , correctly identifies the random challenge bit b . The a priori success probability being $1/2$, the advantage is appropriately scaled. This advantage is equal to the statistical distance between the random variables $\text{ChA}(\mathcal{E}(M_0))$ and $\text{ChA}(\mathcal{E}(M_1))$. The mutual-information security advantages are defined via

$$\mathbf{Adv}^{\text{mir-r}}(\mathcal{E}; \text{ChA}) = I(U; \text{ChA}(\mathcal{E}(U))) \quad (4)$$

$$\mathbf{Adv}^{\text{mis}}(\mathcal{E}; \text{ChA}) = \max_M I(M; \text{ChA}(\mathcal{E}(M))) \quad (5)$$

where the random variable U is uniformly distributed over $\{0, 1\}^m$.

DS IS EQUIVALENT TO SS. Theorem 1 below says that SS and DS are equivalent up to a small constant factor in the advantage. This is helpful because DS is more analytically tractable than SS. The proof is an extension of the classical ones in computational cryptography and is given in [5].

Theorem 1. [DS \leftrightarrow SS] Let $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ be an encryption function and ChA an adversary channel. Then $\mathbf{Adv}^{\text{ss}}(\mathcal{E}; \text{ChA}) \leq \mathbf{Adv}^{\text{ds}}(\mathcal{E}; \text{ChA}) \leq 2 \cdot \mathbf{Adv}^{\text{ss}}(\mathcal{E}; \text{ChA})$. ■

MIS IMPLIES DS. The KL divergence is a distance measure for probability distributions P, Q defined by $\mathbf{D}(P; Q) = \sum_x P(x) \lg P(x)/Q(x)$. Let M, C be random variables. Probability distributions $J_{M,C}, I_{M,C}$ are defined for all M, C by $J_{M,C}(M, C) = \Pr[M = M, C = C]$ and $I_{M,C}(M, C) = \Pr[M = M] \cdot \Pr[C = C]$. Thus $J_{M,C}$ is the joint distribution of M and C , while $I_{M,C}$ is the “independent” or product distribution. The following is standard:

Lemma 2. Let M, C be random variables. Then $I(M; C) = \mathbf{D}(J_{M,C}; I_{M,C})$. ■

Pinsker’s inequality—from [35] with the tight constant from [12]—lower bounds the KL divergence between two distributions in terms of their statistical distance:

Lemma 3. Let P, Q be probability distributions. Then $\mathbf{D}(P; Q) \geq 2 \cdot \mathbf{SD}(P; Q)^2$. ■

To use the above we need the following, whose proof is in [5]:

Lemma 4. Let M be uniformly distributed over $\{M_0, M_1\} \subseteq \{0, 1\}^m$. Let $g: \{0, 1\}^m \rightarrow \{0, 1\}^c$ be a transform and let $C = g(M)$. Then $\mathbf{SD}(J_{M,C}; I_{M,C})$ equals $\mathbf{SD}(g(M_0); g(M_1))/2$. ■

Combining the lemmas, we show the following in [5]:

Theorem 5. [MIS \rightarrow DS] Let $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ be an encryption function and ChA an adversary channel. Then $\mathbf{Adv}^{\text{ds}}(\mathcal{E}; \text{ChA}) \leq \sqrt{2 \cdot \mathbf{Adv}^{\text{mis}}(\mathcal{E}; \text{ChA})}$. ■

DS IMPLIES MIS. The following general lemma from [5] bounds the difference in entropy between two distributions in terms of their statistical distance. It is a slight strengthening of [11, Theorem 16.3.2]. Similar bounds are provided in [22].

Lemma 6. Let P, Q be probability distributions. Let $N = |\text{SUPP}(P) \cup \text{SUPP}(Q)|$ and $\epsilon = \mathbf{SD}(P; Q)$. Then $\mathbf{H}(P) - \mathbf{H}(Q) \leq 2\epsilon \cdot \lg(N/\epsilon)$. ■

To exploit this, we define the *pairwise statistical distance* $\mathbf{PSD}(\mathbf{M}; \mathbf{C})$ between random variables \mathbf{M}, \mathbf{C} as the maximum, over all messages $M_0, M_1 \in \text{SUPP}(P_{\mathbf{M}})$, of $\mathbf{SD}(P_{\mathbf{C}|\mathbf{M}=M_0}; P_{\mathbf{C}|\mathbf{M}=M_1})$. The proof of the following is in [5].

Lemma 7. Let \mathbf{M}, \mathbf{C} be random variables. Then $\mathbf{SD}(P_{\mathbf{C}}; P_{\mathbf{C}|\mathbf{M}=M}) \leq \mathbf{PSD}(\mathbf{M}; \mathbf{C})$ for any M . ■

From this we conclude in [5] that DS implies MIS:

Theorem 8. [DS \rightarrow MIS] Let $\mathcal{E}: \{0, 1\}^m \rightarrow \{0, 1\}^c$ be an encryption function and ChA an adversary channel. Let $\epsilon = \mathbf{Adv}^{\text{ds}}(\mathcal{E}; \text{ChA})$. Then $\mathbf{Adv}^{\text{mis}}(\mathcal{E}; \text{ChA}) \leq 2\epsilon \cdot \lg(2^c/\epsilon)$. ■

The somewhat strange-looking form of the bound of Theorem 8 naturally raises the question of whether Lemma 6 is tight. The following says that it is up to a constant factor of 4. The proof is in [5].

Proposition 9. Let $n > k \geq 1$ be integers. Let $\epsilon = 2^{-k}$ and $N = 1 + \epsilon 2^n$. Then there are distributions P, Q with $|\text{SUPP}(P) \cup \text{SUPP}(Q)| = N$ and $\mathbf{SD}(P; Q) = \epsilon$ and $\mathbf{H}(P) - \mathbf{H}(Q) \geq 0.5 \cdot \epsilon \cdot \lg(N/\epsilon)$. ■

OTHER RELATIONS. We have now justified all the numbered implication arrows in Fig. 2. The un-numbered implication **MIS** \rightarrow **MIS-R** is trivial. The intuition for the separation **MIS-R** $\not\rightarrow$ **MIS** is simple. Let \mathcal{E} be the identity function. Let ChA faithfully transmit inputs 0^m and 1^m and be very noisy on other inputs. Then **MIS** fails because the adversary has high advantage when the message takes on only values $0^m, 1^m$ but **MIS-R**-security holds since these messages are unlikely. This example may seem artificial. In [5] we give a more complex example where ChA is a BSC and the encryption function is no longer trivial.

4 DS-Secure Encryption Achieving Secrecy Capacity

This section presents our main technical result, an encryption scheme achieving DS-security. Its rate, for a large set of adversary channels, is optimal.

HIGH-LEVEL APPROACH. We start by considering an extension of the usual setting where sender and receiver share a *public* random value S , i.e., known to the adversary, and which we call the *seed*. We will call an encryption function in this setting a *seeded encryption function*. For simplicity, this discussion will focus on the case where ChR and ChA are BSCs with respective crossover probabilities $p_R < p_A \leq 1/2$, and we assume that sender and receiver only want to agree on a joint secret key. If we let S be the seed of an extractor $\text{Ext}: \text{Sds} \times \{0, 1\}^k \rightarrow \{0, 1\}^m$ and given an error-correcting code $\mathsf{E}: \{0, 1\}^k \rightarrow \{0, 1\}^n$ for reliable communication over BSC_{p_R} , a natural approach consists of the sender sending $\mathsf{E}(R)$, for a random k -bit R , to the receiver, and both parties now derive the key as $K = \text{Ext}(S, R)$, since the receiver can recover R with very high probability.

The achievable key length is at most $\mathbf{H}_\infty(R|Z)$, where $Z = \text{BSC}_{p_A}(\mathsf{E}(R))$. Yet, it is not hard to see that the most likely outcome, when $Z = z$, is that R equals the unique r such that $\mathsf{E}(r) = z$, and that hence $\mathbf{H}_\infty(R|Z) = n \cdot \lg(1/(1 - p_A))$, falling short of achieving capacity $h(p_A) - h(p_R)$. To overcome this, we will observe the following: We can think of BSC_{p_A} as adding an n -bit vector E to its input $\mathsf{E}(R)$, where each bit $E[i]$ of the noise vector takes value one with probability p_A . With overwhelming probability, E is (roughly) uniformly distributed on the set of n -bit vectors with hamming weight (approximately) $p_A \cdot n$ and there are (approximately) $2^{n \cdot h_2(p_A)}$ such vectors. Therefore, choosing the noise uniformly from such vectors does not change the experiment much, and moreover, in this new experiment, one can show that roughly $\mathbf{H}_\infty(R|Z) \geq k - n \cdot (1 - h_2(p_A))$, which yields optimal rate using an optimal code with $k \approx (1 - h(p_R)) \cdot n$. We will make this precise for a general class of symmetric channels via the notion of *smooth min-entropy* [36].

But recall that our goal is way more ambitious: Alice wants to send an *arbitrary message of her choice*. The obvious approach is obtain a key K as above and then send an error-corrected version of $K \oplus M$. But this at least halves the rate, which becomes far from optimal. Our approach instead is to use an extractor Ext that is *invertible*, in the sense that given M and S , we can sample a random R such that $\text{Ext}(S, R) = M$. We then encrypt a message M as $\mathsf{E}(R)$, where R is a random preimage of M under $\text{Ext}(S, \cdot)$. However, the above argument only yields, at best, security for randomly chosen messages. In contrast, showing DS-security accounts to proving, for any two messages M_0 and M_1 , that $\text{BSC}_{p_A}(\mathsf{E}(R_0))$ and $\text{BSC}_{p_A}(\mathsf{E}(R_1))$ are statistically close, where R_i is uniform such that $\text{Ext}(S, R_i) = M_i$. To make things even worse, we allow the messages M_0 and M_1 are allowed to depend on the seed. The main challenge is that such proof appears to require detailed knowledge of the combinatorial structure of E and Ext , as the actual ciphertext distribution depends on them.

Instead, we will take a completely different approach: We show that any seeded encryption function with appropriate linearity properties is DS-secure whenever it is secure for random messages. This result is surprising, as random-message security does *not*, in general, imply chosen-message security. A careful choice of the extractor to satisfy these requirements, combined with the above idea, yields a DS-secure seeded encryption function. The final step is to remove the seed, which is done by transmitting it (error-corrected) and amortizing out its impact on the rate to essentially zero by re-using it with the above seeded encryption function across blocks of the message. A hybrid argument is used to bound the decoding error and loss in security.

SEEDED ENCRYPTION. A *seeded encryption function* \mathcal{SE} : $\text{SDS} \times \{0, 1\}^b \rightarrow \{0, 1\}^n$ takes a seed $S \in \text{SDS}$ and message $M \in \{0, 1\}^b$ to return a sender ciphertext denoted $\mathcal{SE}(S, M)$ or $\mathcal{SE}_S(M)$; each seed S defines an encryption function $\mathcal{SE}_S: \{0, 1\}^b \rightarrow \{0, 1\}^n$. There must be a corresponding seeded decryption function \mathcal{SD} : $\text{SDS} \times \{0, 1\}^n \rightarrow \{0, 1\}^b$ such that $\mathcal{SD}(S, \mathcal{SE}(S, M)) = M$ for all S, M . We consider an extension of the standard wiretap setting where a seed $S \leftarrow \text{s}$ SDS is a public parameter, available to sender, receiver and adversary. We extend

transform $\mathcal{SE}(S, M)$: $// S \in \text{SDS}, M \in \{0, 1\}^b$ $R \leftarrow \$_S \{0, 1\}^r ; \text{Ret } \mathbf{E}(\text{Inv}(S, R, M)) .$ transform $\mathcal{E}(M)$: $// M \in \{0, 1\}^m$ $S \leftarrow \$_S \text{SDS} ; S[1], \dots, S[c] \xleftarrow{k} S$ $M[1], \dots, M[t] \xleftarrow{b} M$ $\text{For } i = 1 \text{ to } t \text{ do } C[i] \leftarrow \mathcal{SE}(S, M[i])$ $\text{Ret } \mathbf{E}(S[1]) \parallel \dots \parallel \mathbf{E}(S[c]) \parallel C[1] \parallel \dots \parallel C[t] .$	transform $\mathcal{D}(C)$: // $C \in \text{OUTR}^{(c+t)n}$ $C[1], \dots, C[c+t] \xleftarrow{n} C$ $S \leftarrow \mathbf{D}(C[1]) \parallel \dots \parallel \mathbf{D}(C[c])$ $\text{For } i = 1 \text{ to } t \text{ do}$ $X[i] \leftarrow \mathbf{D}(C[c+i])$ $M[i] \leftarrow \mathbf{Ext}(S, X[i])$ $\text{Ret } M[1] \parallel \dots \parallel M[t] .$
--	---

Fig. 3. Encryption function $\mathcal{E} = \mathbf{RItE}_t[\text{Inv}, \mathbf{E}]$ using $\mathcal{SE} = \mathbf{ItE}[\text{Inv}, \mathbf{E}]$ and decryption function \mathcal{D} . By $X[1], \dots, X[c] \xleftarrow{b} X$ we mean that b -bit string X is split into b -bit blocks.

DS-security to this setting by letting $\mathbf{Adv}^{\text{ds}}(\mathcal{SE}; \text{ChA})$ be the expectation, over S drawn at random from SDS, of $\mathbf{Adv}^{\text{ds}}(\mathcal{SE}_S; \text{ChA})$. The rate of \mathcal{SE} is defined as b/n , meaning the seed is ignored.

EXTRACTORS. A function $\mathbf{Ext} : \text{SDS} \times \{0, 1\}^k \rightarrow \{0, 1\}^b$ is an (h, α) -extractor if $\mathbf{SD}((\mathbf{Ext}(S, X), Z, S); (U, Z, S)) \leq \alpha$ for all pairs of (correlated) random variables (X, Z) over $\{0, 1\}^k \times \{0, 1\}^*$ with $\mathbf{H}_\infty(X|Z) \geq h$, where additionally S and U are uniform on SDS and $\{0, 1\}^b$, respectively. (This is a strong, average case extractor in the terminology of [16].) We will say that \mathbf{Ext} is *regular* if for all $S \in \text{SDS}$, the function $\mathbf{Ext}(S, \cdot)$ is regular, meaning every point in the range has the same number of preimages.

INVERTING EXTRACTORS. We say that a function $\mathbf{Inv} : \text{SDS} \times \{0, 1\}^r \times \{0, 1\}^b \rightarrow \{0, 1\}^k$ is an *inverter* for an extractor $\mathbf{Ext} : \text{SDS} \times \{0, 1\}^k \rightarrow \{0, 1\}^b$ if for all $S \in \text{SDS}$ and $Y \in \{0, 1\}^b$, and for R uniform over $\{0, 1\}^k$, the random variable $\mathbf{Inv}(S, R, Y)$ is uniformly distributed on $\{X \in \{0, 1\}^k : \mathbf{Ext}(S, X) = Y\}$, the set of preimages of Y under $\mathbf{Ext}(S, \cdot)$. To make this concrete we give an example of an extractor with an efficiently computable inverter. Recall that k -bit strings can be interpreted as elements of the finite field $\text{GF}(2^k)$, allowing us to define a multiplication operator \odot on k -bit strings. Then, for $\text{SDS} = \{0, 1\}^k \setminus 0^k$, we consider the function $\mathbf{Ext} : \text{SDS} \times \{0, 1\}^k \rightarrow \{0, 1\}^b$ which, on inputs $S \in \text{SDS}$ and $X \in \{0, 1\}^k$, outputs the first b bits of $X \odot S$. It is easy to see that \mathbf{Ext} is regular, as 0^k is not in the set of seeds. In [4] we prove the following using the average-case version of the Leftover Hash Lemma of [20], due to [16].

Lemma 10. *For all $\alpha \in (0, 1]$ and all $b \leq k - 2\lg(1/\alpha) + 2$ the function \mathbf{Ext} is a $(b + 2\lg(1/\alpha) - 2, \alpha)$ -extractor.*

An efficient inverter $\mathbf{Inv} : \text{SDS} \times \{0, 1\}^{k-b} \times \{0, 1\}^b \rightarrow \{0, 1\}^k$ is obtained via $\mathbf{Inv}(S, R, M) = S^{-1} \odot (M \parallel R)$ where S^{-1} is the inverse of S with respect to multiplication in $\text{GF}(2^k)$.

THE RITE CONSTRUCTION. Let $\mathbf{Ext} : \text{SDS} \times \{0, 1\}^k \rightarrow \{0, 1\}^b$ be a regular extractor with inverter $\mathbf{Inv} : \text{SDS} \times \{0, 1\}^r \times \{0, 1\}^b \rightarrow \{0, 1\}^k$. Also let $\mathbf{E} : \{0, 1\}^k \rightarrow \{0, 1\}^n$ be an injective function with $k \leq n$, later to be instantiated

via an ECC. Assume without loss of generality that for some $c \geq 1$, we have $|S| = c \cdot k$ for all $S \in \text{SDS}$. The encryption function \mathcal{E} is described in Fig. 3 and is obtained via the construction **RItE_t** (Repeat Invert-Encode), where $t \geq 1$ is a parameter: As its main component, it relies on the construction **ItE** (Invert-Encode) of a seeded encryption function $\text{ItE}[\text{Inv}, E] : \text{SDS} \times \{0, 1\}^b \rightarrow \{0, 1\}^n$ which applies the inverter Inv to the message, and then applies E to the result. The final, seed-free, encryption function $\text{RItE}_t[\text{Inv}, E]$ then takes an input $M \in \{0, 1\}^m$, where $m = t \cdot b$, splits it into t b -bit blocks $M[1], \dots, M[t]$, chooses a random seed S , and combines an encoding of S with the encryptions of the blocks using \mathcal{SE}_S for $\mathcal{SE} = \text{ItE}[\text{Inv}, E]$.

DECRYPTABILITY. Given a channel $\text{ChR} : \{0, 1\} \rightarrow \text{OUTR}$, a decoder for E over ChR is a function $D : \text{OUTR}^n \rightarrow \{0, 1\}^k$. Its decoding error is defined as $\mathbf{DE}(E; D; \text{ChR}) = \max_{M \in \{0, 1\}^k} \Pr[D(\text{ChR}(E(M))) \neq M]$. Therefore, for any output alphabet OUTR and function $D : \text{OUTR}^n \rightarrow \{0, 1\}^b$, we define the corresponding decryption function for \mathcal{E} over ChR as in Fig. 3. The following lemma summarizes the relation between its decryption error and the one of D .

Lemma 11. [Correct decryption] *Let $\text{ChR} : \{0, 1\} \rightarrow \text{OUTR}$ be a channel, and let \mathcal{E} , D , E , and D be as above. Then, $\mathbf{DE}(\mathcal{E}; D; \text{ChR}) \leq (c + t) \cdot \mathbf{DE}(E; D; \text{ChR})$.* ■

STEP I: FROM RITE TO ItE. We reduce security of **RItE** to that of **ItE**. The proof of the following [4] uses a hybrid argument.

Lemma 12. *Let $t \geq 1$, $\mathcal{E} = \text{RItE}_t[\text{Inv}, E]$ and $\mathcal{SE} = \text{ItE}[\text{Inv}, E]$. For all $\text{ChA} : \{0, 1\}^n \rightarrow \text{OUTA}$ we have $\mathbf{Adv}^{\text{ds}}(\mathcal{E}; \text{ChA}) \leq t \cdot \mathbf{Adv}^{\text{ds}}(\mathcal{SE}; \text{ChA})$.* ■

STEP II: RDS-SECURITY OF ItE. Towards determining the DS-security of **ItE** we first address the seemingly simpler question of proving security under *random* messages. Specifically, for a seeded encryption function $\mathcal{SE} : \text{SDS} \times \{0, 1\}^b \rightarrow \{0, 1\}^n$, we define the rds advantage $\mathbf{Adv}^{\text{rds}}(\mathcal{SE}; \text{ChA})$ as the expectation of $\mathbf{SD}((\text{ChA}(\mathcal{SE}(S, U)), U); (\text{ChA}(\mathcal{SE}(S, U')), U))$ where U and U' are uniformly chosen and independent b -bit messages, and the expectation is taken over the choice of the seed S . Exploiting the notion of ϵ -smooth min-entropy [36], the following, proven in [4], establishes RDS-security of **ItE**:

Lemma 13. [RDS-security of ItE] *Let $\delta > 0$, let $\text{ChA} : \{0, 1\} \rightarrow \text{OUTA}$ be a symmetric channel, let $\text{Inv} : \text{SDS} \times \{0, 1\}^r \times \{0, 1\}^b \rightarrow \{0, 1\}^k$ be the inverter of a regular $(k - n \cdot (\lg(|\text{OUTA}|) - \mathbf{H}(\text{ChA}) + \delta), \alpha)$ -extractor, and let $E : \{0, 1\}^k \rightarrow \{0, 1\}^n$ be injective. Then, for $\mathcal{SE} = \text{ItE}[\text{Inv}, E]$, we have*

$$\mathbf{Adv}^{\text{rds}}(\mathcal{SE}; \text{ChA}) \leq 2 \cdot 2^{-\frac{n\delta^2}{2\lg^2(|\text{OUTA}|+3)}} + \alpha .$$

STEP III: FROM RDS- TO DS-SECURITY. In contrast to RDS-security, proving DS-security of **ItE** seems to require a better grasp of the combinatorial structure of E and Inv . More concretely, think of any randomized (seeded) encryption function $\mathcal{SE} : \text{SDS} \times \{0, 1\}^b \rightarrow \{0, 1\}^n$ as a deterministic map $\mathcal{SE} : \text{SDS} \times \{0, 1\}^r \times$

$\{0, 1\}^b \rightarrow \{0, 1\}^n$ (for some r), where the second argument takes the role of the random coins. We call \mathcal{SE} *separable* if $\mathcal{SE}(S, R, M) = \mathcal{SE}(S, R, 0^b) \oplus \mathcal{SE}(S, 0^r, M)$ for all $S \in \text{SDS}$, $R \in \{0, 1\}^r$, and $M \in \{0, 1\}^b$. Also, it is *message linear* if $\mathcal{SE}(S, 0^r, \cdot)$ is linear for all $S \in \text{SDS}$. The following is true for encryption functions with both these properties, and is proven in [4].

Lemma 14. [RDS \Rightarrow DS] *Let $\text{ChA} : \{0, 1\} \rightarrow \text{OUTA}$ be symmetric. If \mathcal{SE} is separable and message linear, then $\mathbf{Adv}^{\text{ds}}(\mathcal{SE}; \text{ChA}) \leq 2 \cdot \mathbf{Adv}^{\text{rds}}(\mathcal{SE}; \text{ChA})$.* ■

Coming back to **ItE**, we say that $\text{Inv} : \text{SDS} \times \{0, 1\}^r \times \{0, 1\}^b \rightarrow \{0, 1\}^k$ is *output linear* if $\text{Inv}(S, 0^r, \cdot)$ is linear for all $S \in \text{SDS}$. Moreover, it is *separable* if $\text{Inv}(S, R, M) = \text{Inv}(S, R, 0^b) \oplus \text{Inv}(S, 0^r, M)$ for all $S \in \text{SDS}$, $R \in \{0, 1\}^r$, and $M \in \{0, 1\}^b$. For example, the inverter for the above extractor based on finite-field multiplication is easily seen to be output linear and separable, by the linearity of the map $M \mapsto S^{-1} \odot M$.

SECURITY. If we instantiate **ItE**[Inv, E] so that Inv is both output linear and separable, and we let E be linear, the encryption function \mathcal{SE} is easily seen to be message linear and separable. The following theorem now follows immediately by combining Lemma 12, Lemma 14, and Lemma 13.

Theorem 15. [DS-security of RItE] *Let $\delta > 0$ and $t \geq 1$. Also, let $\text{ChA} : \{0, 1\} \rightarrow \text{OUTA}$ be a symmetric channel, let $\text{Inv} : \text{SDS} \times \{0, 1\}^r \times \{0, 1\}^b \rightarrow \{0, 1\}^k$ be the output-linear and separable inverter of a regular $(k - n \cdot (\lg(|\text{OUTA}|) - \mathbf{H}(\text{ChA}) + \delta), \alpha)$ -extractor, and let $\mathsf{E} : \{0, 1\}^k \rightarrow \{0, 1\}^n$ be linear and injective. Then, for $\mathcal{E} = \text{RItE}_t[\text{Inv}, \mathsf{E}]$, we have*

$$\mathbf{Adv}^{\text{ds}}(\mathcal{E}; \text{ChA}) \leq 2t \cdot \left(2 \cdot 2^{-\frac{n\delta^2}{2\lg^2(|\text{OUTA}|+3)}} + \alpha \right).$$
 ■

INSTANTIATING THE SCHEME. Recall that if $\text{ChA} : \{0, 1\}^l \rightarrow \text{OUTA}$ and $\text{ChR} : \{0, 1\}^l \rightarrow \text{OUTR}$ are symmetric channels, their secrecy capacity equals [27] $(\mathbf{H}(\mathbf{U}|\text{ChA}(\mathbf{U})) - \mathbf{H}(\mathbf{U}|\text{ChR}(\mathbf{U}))) / l$, for a uniform l -bit \mathbf{U} . Also, for a channel ChR , we denote its (Shannon) capacity as $\mathbf{C}(\text{ChR}) = \max_{\mathbf{X}} \mathbf{I}(\mathbf{X}; \text{ChR}(\mathbf{X})) / l$. We will need the following result (cf. e.g. [18] for a proof).

Lemma 16. [18] *For any $l \in \mathbb{N}$ and any channel $\text{ChR} : \{0, 1\}^l \rightarrow \text{OUTR}$, there is a family $\mathsf{E} = \{\mathsf{E}_s\}_{s \in \mathbb{N}}$ of linear encoding functions $\mathsf{E}_s : \{0, 1\}^{k(s)} \rightarrow \{0, 1\}^{n(s)}$ (where $n(s)$ is a multiple of l), with corresponding decoding functions $\mathsf{D}_s : \text{OUTR}^{n(s)/l} \rightarrow \{0, 1\}^{k(s)}$, such that (i) $\mathbf{DE}(\mathsf{E}_s; \mathsf{D}_s; \text{ChR}) = 2^{-\Theta(k(s))}$, (ii) $\lim_{s \rightarrow \infty} k(s)/n(s) = \mathbf{C}(\text{ChR})$, and (iii) E and D are PT computable.* ■

We now derive a scheme $\overline{\mathcal{E}} = \{\mathcal{E}_s\}_{s \in \mathbb{N}}$ achieving secrecy capacity for the most common case $\text{ChR} = \text{BSC}_{p_R}$ and $\text{ChA} = \text{BSC}_{p_A}$, where $0 \leq p_R < p_A \leq \frac{1}{2}$. We start with a family of codes $\{\mathsf{E}_s\}_{s \in \mathbb{N}}$ for BSC_{p_R} guaranteed to exist by Lemma 16, where $\mathsf{E}_s : \{0, 1\}^{k(s)} \rightarrow \{0, 1\}^{n(s)}$ and $\lim_{s \rightarrow \infty} k(s)/n(s) = 1 - h_2(p_R)$, or, equivalently, there exists ν such that $\nu(s) = o(1)$ and $k(s) = (1 - h_2(p_R) - \nu(s)) \cdot n(s)$. Then, we let $\delta(s) = (2 \lg^2(5))^{1/2} \cdot n(s)^{-1/4}$ and $\alpha(s) = 2^{-n(s)^{1/2}}$, and use

the finite-field based extractor $\text{Ext}_s : \{0,1\}^{k(s)} \times \{0,1\}^{k(s)} \rightarrow \{0,1\}^{b(s)}$, where $b(s) = k(s) - n(s) \cdot (1 - h_2(p_A) + \delta(s)) + 2\lg(\alpha) = (h_2(p_A) - h_2(p_R) - \nu(s) - \delta(s) - 2 \cdot n(s)^{-1/2}) \cdot n(s)$. We note that the resulting scheme is equivalent to the one described in the introduction (with $A = S^{-1}$). With these parameters,

$$\mathbf{Adv}^{\text{ds}}(\mathcal{E}_s; \text{BSC}_{p_A}) \leq 6 \cdot t(s) \cdot 2^{-\sqrt{n}}, \quad \mathbf{DE}(\mathcal{E}_s; \mathcal{D}_s; \text{BSC}_{p_R}) \leq (t(s) + 1) \cdot 2^{-\Theta(k(s))}$$

by Theorem 15 and Lemma 11, respectively. The rate of \mathcal{E}_s is

$$\mathbf{Rate}(\mathcal{E}_s) = \frac{t(s)}{t(s) + 1} \cdot \left(h_2(p_A) - h_2(p_R) - \nu(s) - \delta(s) - \frac{2}{\sqrt{n(s)}} \right).$$

Setting $t(s) = \lg(k(s))$ yields $\lim_{s \rightarrow \infty} \mathbf{Rate}(\mathcal{E}_s) = h_2(p_A) - h_2(p_R)$.

EXTENSIONS. The proof applies also for any pair of symmetric channels ChR and ChA, and the resulting rate is the secrecy capacity if the capacity of ChA : $\{0,1\} \rightarrow \text{OUTA}$ is $\lg(|\text{OUTA}|) - \mathbf{H}(\text{ChA})$, which is the case if and only if a uniform input to ChA produces a uniform output. For other channels, such as *erasure channels* (where each bit is left unchanged with probability δ and mapped to an erasure symbol with probability $1 - \delta$) our technique still yields good schemes which, however, may fall short of achieving capacity. We also remark that the above presentation is constrained to single input-bit base channels for simplicity only. Our results can be extended to discrete memoryless channels with l -bit inputs for $l > 1$. For example, Lemma 13 extends to arbitrary symmetric channels ChA : $\{0,1\}^l \rightarrow \text{OUTA}$, at the price of replacing n by n/l in the security bound and in the extractor's entropy requirement. In contrast, we do not know whether Lemma 14 applies to arbitrary symmetric channels with l -bit inputs, but it does, for instance, extend to any channel of the form $\text{ChA}(X) = X \oplus E$, where E is an l -bit string sampled according to an input-independent noise distribution.

Acknowledgments. Bellare was supported in part by NSF grants CCF-0915675, CNS-0904380 and CNS-1116800. Tessaro was supported in part by NSF grants CCF-0915675, CCF-1018064, and DARPA contracts FA8750-11-C-0096, FA8750-11-2-0225.

References

1. Andersson, M., Rathi, V., Thobaben, R., Kliewer, J., Skoglund, M.: Nested polar codes for wiretap and relay channels. Available at arxiv.org/abs/1006.3573 (2010)
2. Arikan, E.: Channel polarization: A method for constructing capacity achieving codes for symmetric binary-input memoryless channels. IEEE Transactions on Information Theory 55(7), 3051–3073 (2009)
3. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS, pp. 394–403. IEEE Computer Society Press (October 1997)
4. Bellare, M., Tessaro, S.: Polynomial-time, semantically-secure encryption achieving the secrecy capacity. Available as arxiv.org/abs/1201.3160 and Cryptology Eprint Archive Report 2012/022 (January 2012)

5. Bellare, M., Tessaro, S., Vardy, A.: A cryptographic treatment of the wiretap channel. Available as arxiv.org/abs/1201.2205 and Cryptology Eprint Archive Report 2012/15 (January 2012)
6. Bloch, M., Barros, J.: Physical-Layer Security: From Information Theory to Security Engineering. Cambridge Academic Press (2011)
7. Bloch, M., Laneman, J.N.: On the secrecy capacity of arbitrary wiretap channels. In: Proceedings of the 46th Allerton Conference on Communications, Control, and Computing, pp. 818–825 (September 2008)
8. Cheraghchi, M., Didier, F., Shokrollahi, A.: Invertible extractors and wiretap protocols. *IEEE Transactions on Information Theory* 58(2), 1254–1274 (2012)
9. Cohen, G., Zémor, G.: The wiretap channel applied to biometrics. In: Proc. of the International Symposium on Information Theory and Applications (2004)
10. Cohen, G., Zémor, G.: Syndrome coding for the wire-tap channel revisited. In: Proc. of the IEEE Information Theory Workshop (ITW 2006), pp. 33–36. IEEE (2006)
11. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley and Sons (1991)
12. Csiszár, I.: Information-type measures of difference of probability distributions and indirect observations. *Studia Scientiarum Mathematicarum Hungarica* 2, 299–318 (1967)
13. Csiszár, I.: Almost independence and secrecy capacity. *Problems of Information Transmission* 32(1), 40–47 (1996)
14. Csiszár, I., Körner, J.: Broadcast channels with confidential messages. *IEEE Transactions on Information Theory* 24(3), 339–348 (1978)
15. Damgård, I., Pedersen, T., Pfitzmann, B.: Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory* 44(3), 1143–1151 (1998)
16. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing* 38(1), 97–139 (2008)
17. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
18. Dumer, I.: Concatenated codes and their multilevel generalizations. In: The Handbook of Coding Theory, pp. 1191–1988. Elsevier (1998)
19. Goldwasser, S., Micali, S.: Probabilistic encryption. *Journal of Computer and System Sciences* 28(2), 270–299 (1984)
20. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. *SIAM Journal on Computing* 28(4), 1364–1396 (1999)
21. Hayashi, M., Matsumoto, R.: Construction of wiretap codes from ordinary channel codes. In: Proceedings of the 2010 IEEE International Symposium on Information Theory (ISIT 2010), pp. 2538–2542. IEEE (2010)
22. Ho, S., Yeung, R.: The interplay between entropy and variational distance. *IEEE Transactions on Information Theory* 56(12), 5906–5929 (2010)
23. Hof, E., Shamai, S.: Secrecy-achieving polar-coding. In: Proceedings of the IEEE Information Theory Workshop (ITW 2010). IEEE (2010)
24. ICC 2011 workshop on physical-layer security, Kyoto, Japan (June 2011)
25. Iwamoto, M., Ohta, K.: Security notions for information theoretically secure encryptions. In: Proceedings of the 2011 IEEE International Symposium on Information Theory (ISIT 2011), pp. 1777–1781. IEEE (2011)

26. Koayluoglu, O., ElGamal, H.: Polar coding for secure transmission. In: Proceedings of the IEEE International Symposium on Personal Indoor and Mobile Radio Communication, pp. 2698–2703 (2010)
27. Leung-Yan-Cheong, S.: On a special class of wire-tap channels. *IEEE Transactions on Information Theory* 23(5), 625–627 (1977)
28. Liang, Y., Poor, H., Shamai, S.: Information theoretic security. *Foundations and Trends in Communications and Information Theory* 5(4), 355–580 (2008)
29. Mahdavifar, H., Vardy, A.: Achieving the secrecy capacity of wiretap channels using polar codes. In: Proceedings of the 2010 IEEE International Symposium on Information Theory (ISIT 2010), pp. 913–917. IEEE (2010)
30. Mahdavifar, H., Vardy, A.: Achieving the secrecy capacity of wiretap channels using polar codes. *IEEE Transactions on Information Theory* 57(10), 6428–6443 (2011)
31. Maurer, U.: The strong secret key rate of discrete random triples. In: Blahut, R.E. (ed.) *Communication and Cryptography – Two Sides of One Tapestry*, pp. 271–285. Kluwer (1994)
32. Maurer, U.M., Wolf, S.: Information-Theoretic Key Agreement: From Weak to Strong Secrecy for Free. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 351–368. Springer, Heidelberg (2000)
33. Muramatsu, J., Miyake, S.: Construction of wiretap channel codes by using sparse matrices. In: Proc. of the IEEE Information Theory Workshop (ITW 2009), pp. 105–109. IEEE (2009)
34. Ozarow, L.H., Wyner, A.D.: Wire-Tap Channel II. In: Beth, T., Cot, N., Ingemansson, I. (eds.) *EUROCRYPT 1984*. LNCS, vol. 209, pp. 33–50. Springer, Heidelberg (1985)
35. Pinsker, M.S.: Information and information stability of random variables and processes. Holden Day, San Francisco (1964)
36. Renner, R., Wolf, S.: Simple and Tight Bounds for Information Reconciliation and Privacy Amplification. In: Roy, B. (ed.) *ASIACRYPT 2005*. LNCS, vol. 3788, pp. 199–216. Springer, Heidelberg (2005)
37. Shannon, C.E.: A mathematical theory of communication. *The Bell System Technical Journal* 27, 379–423, 623–656 (1948)
38. Suresh, A., Subramanian, A., Thangaraj, A., Bloch, M., McLaughlin, S.W.: Strong secrecy for erasure wiretap channels. In: Proc. of the IEEE Information Theory Workshop (ITW 2010). IEEE (2010)
39. Tal, I., Vardy, A.: How to construct polar codes. In: Proc. of the IEEE Information Theory Workshop (ITW 2010). IEEE (2010)
40. Thangaraj, A., Dihidar, S., Calderbank, A., McLaughlin, S., Merolla, J.: Applications of LDPC codes to the wiretap channel. *IEEE Transactions on Information Theory* 53(8), 2933–2945 (2007)
41. Wyner, A.D.: The wire-tap channel. *Bell Systems Tech. Journal* 54(8), 1355–1387 (1975)
42. Ziva corporation, <http://www.ziva-corp.com/>

Multi-instance Security and Its Application to Password-Based Cryptography

Mihir Bellare¹, Thomas Ristenpart², and Stefano Tessaro³

¹ Department of Computer Science & Engineering, University of California San Diego
cseweb.ucsd.edu/~mihir/

² Department of Computer Sciences, University of Wisconsin - Madison
pages.cs.wisc.edu/~rist/

³ CSAIL, Massachusetts Institute of Technology
people.csail.mit.edu/tessaro/

Abstract. This paper develops a theory of multi-instance (mi) security and applies it to provide the first proof-based support for the classical practice of salting in password-based cryptography. Mi-security comes into play in settings (like password-based cryptography) where it is computationally feasible to compromise a single instance, and provides a second line of defense, aiming to ensure (in the case of passwords, via salting) that the effort to compromise all of some large number m of instances grows linearly with m . The first challenge is definitions, where we suggest LORX-security as a good metric for mi security of encryption and support this claim by showing it implies other natural metrics, illustrating in the process that even lifting simple results from the si setting to the mi one calls for new techniques. Next we provide a composition-based framework to transfer standard single-instance (si) security to mi-security with the aid of a key-derivation function. Analyzing password-based KDFs from the PKCS#5 standard to show that they meet our indistinguishability-style mi-security definition for KDFs, we are able to conclude with the first proof that per password salts amplify mi-security as hoped in practice. We believe that mi-security is of interest in other domains and that this work provides the foundation for its further theoretical development and practical application.

1 Introduction

This paper develops a theory of *multi-instance security* and applies it to support practices in password-based cryptography.

BACKGROUND. Password-based encryption (PBE) in practice is based on the PKCS#5 (equivalently, RFC 2898) standard [32]. It encrypts a message M under a password pw by picking a random s -bit *salt* sa , deriving a key $L \leftarrow \text{KD}(pw\|sa)$ and returning $C' \leftarrow C\|sa$ where $C \leftarrow \mathcal{E}(L, M)$. Here \mathcal{E} is a symmetric encryption scheme, typically an IND-CPA AES mode of operation, and key-derivation function (KDF) $\text{KD}: \{0,1\}^* \rightarrow \{0,1\}^n$ is the c -fold iteration $\text{KD} = H^c$ of a cryptographic hash function $H: \{0,1\}^* \rightarrow \{0,1\}^n$. However, passwords are often poorly chosen [29], falling within a set D called a “dictionary” that is small

enough to exhaust. A brute-force attack now recovers the target password pw (thereby breaking the ind-cpa security of the encryption) using cN hashes where $N = |D|$ is the size of the dictionary. Increasing c increases this effort, explaining the role of this iteration count, but c cannot be made too large without adversely impacting the performance of PBE.

Consider now m users, the i -th with password pw_i . If the salt is absent ($s = 0$), the number of hashes for the brute force attack to recover all m passwords remains around cN , but if s is large enough that salts are usually distinct, it rises to mcN , becoming prohibitive for large m . Salting, thus, aims to make the effort to compromise m target passwords scale linearly in m . (It has no effect on the security of encryption under any one, particular target password.)

NEW DIRECTIONS. This practice, in our view, opens a new vista in theoretical cryptography, namely to look at the multi-instance (mi) security of a scheme. We would seek metrics of security under which an adversary wins when it breaks all of m instances *but not if it breaks fewer*. This means that the mi security could potentially be much higher than the traditional single-instance (si) security. We would have security amplification.

Why do this? As the above discussion of password-based cryptography shows, there are settings where the computational effort t needed to compromise a single instance is feasible. Rather than give up, we provide a second line of defense. We limit the *scale* of the damage, ensuring (in the case of passwords, via the mechanism of salting) that the computational effort to compromise all of m instances is (around) tm and thus prohibitive for large m . We can't prevent the occasional illness, but we can prevent an epidemic.

We initiate the study of multi-instance security with a foundational treatment in two parts. The first part is agnostic to whether the setting is password-based or not, providing definitions for different kinds of mi-security of encryption and establishing relations between them, concluding with the message that what we call LORX-security is a good choice. The second part of our treatment focuses on password-based cryptography, providing a modular framework that proves mi-security of password-based primitives by viewing them as obtained by the composition of a mi-secure KDF with a si-secure primitive, and yielding in particular the first proof that salting works as expected to increase multi-instance security under a strong and formal metric for the latter.

Multi-instance security turns out to be challenging both definitionally (providing metrics where the adversary wins on breaking all instances but not fewer) and technically (reductions need to preserve tiny advantages and standard hybrid arguments no longer work). It also connects in interesting ways to security amplification via direct products and xor lemmas, eg. [37,16,19,30,13,27,34,28,35]. (We import some of their methods and export some novel viewpoints.) We believe there are many fruitful directions for future work, both theoretical (pursuing the connection with security amplification) and applied (mi security could be valuable in public-key cryptography where steadily improving attacks are making current security parameters look uncomfortably close to the edge for single-instance security). Let us now look at all this in some more detail.

LORX. We consider a setting with m independent target keys K_1, \dots, K_m . (They may, but need not, be passwords.) In order to show that mi-security grows with m we want a metric (definition) where the adversary wins if it breaks all m instances of the encryption but does not win if it breaks strictly fewer. If “breaking” is interpreted as recovery of the key then such a metric is easily given: it is the probability that the adversary recovers all m target keys. We refer to this as the UKU (Universal Key Unrecoverability) metric. But we know very well that key-recovery is a weak metric of encryption security. We want instead a mi analog of ind-cpa. The first thing that might come to mind is multi-user security [3,2]. But in the latter the adversary wins (gets an advantage of one) even if it breaks just one instance so the mu-advantage of an adversary can never be less than its si (ind-cpa) advantage. We, in contrast, cannot “give up” once a single instance is broken. Something radically different is needed.

Our answer is LORX (left-or-right xor indistinguishability). Our game picks m independent challenge bits b_1, \dots, b_m and gives the adversary an oracle **Enc**(\cdot, \cdot, \cdot) which given i, M_0, M_1 returns an encryption of M_{b_i} under K_i . The adversary outputs a bit b' and its advantage is $2\Pr[b' = b_1 \oplus \dots \oplus b_m] - 1$.¹ Why xor? Its well-known “sensitivity” means that even if the adversary figures out $m-1$ of the challenge bits, it will have low advantage unless it also figures out the last. This intuitive and historical support is strengthened by the relations, discussed below, that show that LORX implies security under other natural metrics.

RELATIONS. The novelty of multi-instance security prompts us to step back and consider a broad choice of definitions. Besides UKU and LORX, we define RORX (real-or-random xor indistinguishability, a mi-adaptation of the si ROR notion of [4]) and a natural AND metric where the challenge bits b_1, \dots, b_m and oracle **Enc**(\cdot, \cdot, \cdot) are as in the LORX game but the adversary output is a vector (b'_1, \dots, b'_m) and its advantage is $\Pr[(b'_1, \dots, b'_m) = (b_1, \dots, b_m)] - 2^{-m}$. The relations we provide, summarized in Figure 1, show that LORX emerges as the best choice because it implies all the others with tight reductions. Beyond that, they illustrate that the mi terrain differs from the si one in perhaps surprising ways, both in terms of relations and the techniques needed to establish them.

Thus, in the si setting, LOR and ROR are easily shown equivalent up to a factor 2 in the advantages [4]. It continues to be true that LORX easily implies RORX but the hybrid argument used to prove that ROR implies LOR [4] does not easily extend to the mi setting and the proof that RORX implies LORX is not only more involved but incurs a factor 2^m loss.² In the si setting, both

¹ This is a simplification of our actual definition, which allows the adversary to adaptively corrupt instances to reveal the underlying keys and challenge bits. This capability means that LORX-security implies threshold security where the adversary wins if it predicts the xor of the challenge bits of some subset of the instances of its choice. See Section 2 for further justification for this feature of the model.

² This (exponential) 2^m factor loss is a natural consequence of the factor of 2 loss in the si case, our bound is tight, and the loss in applications is usually small because advantages are already exponentially vanishing in m . Nonetheless it is not always negligible and makes LORX preferable to RORX.

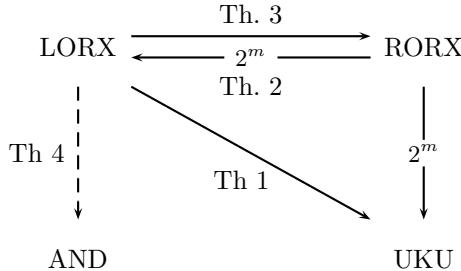


Fig. 1. Notions of multi-instance security for encryption and their relations. LORX (left-or-right xor indistinguishability) emerges as the strongest, tightly implying RORX (real-or-random xor indistinguishability) and UKU (universal key-unrecoverability). The dashed line indicates that under some (mild, usually met) conditions LORX also implies AND. RORX implies LORX and UKU but with a 2^m loss in advantage where m is the number of instances, making LORX a better choice.

LOR and ROR are easily shown to imply KU (key unrecoverability). Showing LORX implies UKU is more involved, needing a boosting argument to ensure preservation of exponentially-vanishing advantages. This reduction is tight but, interestingly, the reduction showing RORX implies UKU is not, incurring a 2^m -factor loss, again indicating that LORX is a better choice. We show that LORX usually implies AND by exploiting a direct product theorem by Unger [35], evidencing the connections with this area. Another natural metric of mi-security is a threshold one, but our incorporation of corruptions means that LORX implies security under this metric.

MI-SECURITY OF PBE. Under the LORX metric, we prove that the advantage ϵ' obtained by a time t adversary against m instances of the above PBE scheme \mathcal{E}' is at most $\epsilon + (q/mcN)^m$ (we are dropping negligible terms) where q is the number of adversary queries to RO H and ϵ is the advantage of a time t ind-cpa (si) adversary against \mathcal{E} . This is the desired result saying that salting works to provide a second line of defense under a strong mi security metric, amplifying security linearly in the number of instances.

FRAMEWORK. This result for PBE is established in a modular (rather than ad hoc) way, via a framework that yields corresponding results for any password-based primitive. This means not only ones like password-based message authentication (also covered in PKCS#5) or password-based authenticated encryption (WinZip) but public-key primitives like password-based digital signatures, where the signing key is derived from a password. We view a password-based scheme for a goal as derived by composing a key-derivation function (KDF) with a standard (si) scheme for the same goal. The framework then has the following components. (1) We provide a definition of mi-security for KDFs. (2) We provide composition theorems, showing that composing a mi-secure KDF with a si-secure scheme for a goal results in a mi-secure scheme for that goal. (We will illustrate this for the case of encryption but similar results may be shown for other primitives.) (3) We analyze the iterated hash KDF of PKCS#5 and establish its mi security.

The statements above are qualitative. The quantitative security aspect is crucial. The definition of mi-security of KDFs must permit showing mi-security much higher than si-security. The reductions in the composition theorems must preserve exponentially vanishing mi-advantages. And the analysis of the PKCS#5 KDF must prove that the adversary advantage in q queries to the RO H grows as $(q/cmN)^m$, not merely q/cN . These quantitative constraints represent important technical challenges.

Mi-SECURITY OF KDFs. We expand on item (1) above. The definition of mi-security we provide for KDFs is a simulation-based one inspired by the indistinguishability framework [26,11]. The attacker must distinguish between the real world and an ideal counterpart. In both, target passwords pw_1, \dots, pw_m and salts sa_1, \dots, sa_m are randomly chosen. In the real world, the adversary gets input $(pw_1, sa_1, \text{KD}(pw_1\|sa_1)), \dots, (pw_m, sa_m, \text{KD}(pw_m\|sa_1))$ and also gets an oracle for the RO hash function H used by KD. In the ideal world, the input is $(pw_1, sa_1, L_1), \dots, (pw_m, sa_m, L_m)$ where the keys L_1, \dots, L_m are randomly chosen, and the oracle is a simulator. The simulator itself has access to a **Test** oracle that will take a guess for a password and tell the simulator whether or not it matches one of the target passwords. Crucially, we require that when the number of queries made by the adversary to the simulator is q , the number of queries made by the simulator to its **Test** oracle is only q/c . This restriction is critical to our proof of security amplification and a source of challenges therein.

RELATED WORK. Previous work which aimed at providing proof-based assurances for password-based key-derivation has focused on the single-instance case and the role of iteration as represented by the iteration count c . Our work focuses on the multi-instance case and the roles of both salting and iteration.

The UNIX password hashing algorithm maps a password pw to $E_{pw}^c(0)$ where E is a blockcipher and 0 is a constant. Luby and Rackoff [24] show this is a one-way function when $c = 1$ and pw is a random blockcipher key. (So their result does not really cover passwords.) Wagner and Goldberg [36] treat the more general case of arbitrary c and keys that are passwords, but the goal continues to be to establish one-wayness and no security amplification (meaning increase in security with c) is shown. Boyen [8,9] suggests various ways to enhance security, including letting users pick their own iteration counts.

Yao and Yin [38] give a natural pseudorandomness definition of a KDF in which the attacker gets (K, sa) where K is either $H^c(pw\|sa)$ or a random string of the same length and must determine which. Modeling H as a random oracle (RO) [7] to which the adversary makes q queries, they claim to prove that the adversary's advantage is at most q/cN plus a negligible term. This would establish single-instance security amplification by showing that iteration works as expected to increase attacker effort.³ However, even though salts are considered,

³ Unfortunately, we point in [6] to a bug in the proof of [38, Lemma 2.2] and explain why the bound claimed by [38, Theorem 1] is wrong. Beyond this, the proof makes some rather large and not fully justified jumps. The special case $m = 1$ of our treatment will fill these gaps and recover the main claim of [38].

this does not consider multi-instance security let alone establish multi-instance security amplification, and their definition of KDF security does not adapt to allow this. (We use, as indicated above, an indifferentiability-style definition.) In fact the KDF definition of [38] is not even sufficient to establish si security of password-based encryption in the case the latter, as specified in PKCS#5, picks a fresh salt for each message encrypted. Kelsey, Schneier, Hall and Wagner [21] look into the time for password-recovery attacks for different choices of KDFs.

KDFs are for use in non-interactive settings like encryption with WinZip. The issues and questions we consider do not arise with password authenticated key exchange (PAKE) [5,10,14] where definitions already guarantee that the session key may be safely used for encryption. There are no salts and no amplification issues. Abadi and Warinschi [1] provide a si, key-recovery definition for PBE security and connect this with symbolic notions. They do not consider mi security. Dodis, Gennaro, Håstad, Krawczyk and Rabin [12] treat statistically-secure key derivation using hash functions and block ciphers. As discussed in-depth by Kraczyk [23], these results and techniques aren't useful for password-based KDFs because passwords aren't large enough, let alone have the sufficient amount of min-entropy. Kraczyk [23] also notes that his two-stage KDF approach could be used to build password-based KDFs by replacing the extraction stage with a key-stretching operation. Our general framework may be used to analyze the mi-security of this construction.

Work on direct product theorems and XOR lemmas (eg. [37,15,18,13,27]) has considered the problem of breaking multiple instances of a cryptographic primitive, in general as an intermediate step to amplifying security in the single-instance setting. Mi-Xor-security is used in this way in [13,27].

2 The Multi-instance Terrain

This section defines metrics of mi-secure encryption and explores the relations between them to establish the notions and results summarized in Figure 1. Our treatment intends to show that the mi terrain is different from the si one in fundamental ways, leading to new definitions, challenges and connections.

SYNTAX. Recall that a symmetric encryption scheme is a triple of algorithms $\mathbf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation algorithm \mathcal{K} outputs a key. The encryption algorithm \mathcal{E} takes a key K and a message M and outputs a ciphertext $C \leftarrow_{\$} \mathcal{E}(K, M)$. The deterministic decryption algorithm \mathcal{D} takes K and a ciphertext C to return either a string or \perp . Correctness requires that $\mathcal{D}(K, \mathcal{E}(K, M)) = M$ for all M with probability 1 over $K \leftarrow_{\$} \mathcal{K}$ and the coins of \mathcal{E} .

To illustrate the issues and choices in defining mi security, we start with key unrecoverability which is simple because it is underlain by a computational game and its mi counterpart is easily and uncontroversially defined. When we move to stronger notions underlain by decisional games, definitions will get more difficult and more contentious as more choices will emerge.

UKU. Single-instance key unrecoverability is formalized via the game $\mathbf{KU}_{\mathbf{SE}}$ where a target key $K \leftarrow_{\$} \mathcal{K}$ is initially sampled, and the adversary \mathcal{A} is given an

<u>main</u> $\text{UKU}_{\text{SE},m}^{\mathcal{A}}$	<u>proc.</u> $\text{Enc}(i, M)$	<u>proc.</u> $\text{Cor}(i)$
$\mathbf{K}[1], \dots, \mathbf{K}[m] \leftarrow \mathcal{K}; \mathbf{K}' \leftarrow \mathcal{A}^{\text{Enc}}$ Ret $\mathbf{K}' = \mathbf{K}$	Ret $\mathcal{E}(\mathbf{K}[i], M)$	Ret $\mathbf{K}[i]$
<u>main</u> $\text{LORX}_{\text{SE},m}^{\mathcal{A}}$	<u>main</u> $\text{AND}_{\text{SE},m}^{\mathcal{A}}$	<u>proc.</u> $\text{Enc}(i, M_0, M_1)$
$\mathbf{K}[1], \dots, \mathbf{K}[m] \leftarrow \mathcal{K}$ $\mathbf{b} \leftarrow \{0, 1\}^m$ $b' \leftarrow \mathcal{A}^{\text{Enc}}$ Ret $(b' = \oplus_i \mathbf{b}[i])$	$\mathbf{K}[1], \dots, \mathbf{K}[m] \leftarrow \mathcal{K}$ $\mathbf{b} \leftarrow \{0, 1\}^m$ $b' \leftarrow \mathcal{A}^{\text{Enc}}$ Ret $(\mathbf{b}' = \mathbf{b})$	If $ M_0 \neq M_1 $ then Ret \perp $C \leftarrow \mathcal{E}(\mathbf{K}[i], M_{\mathbf{b}[i]})$ Ret C
<u>main</u> $\text{RORX}_{\text{SE},m}^{\mathcal{A}}$	<u>proc.</u> $\text{Enc}(i, M)$	<u>proc.</u> $\text{Cor}(i)$
$\mathbf{K}[1], \dots, \mathbf{K}[m] \leftarrow \{0, 1\}^k$ $\mathbf{b} \leftarrow \{0, 1\}^m; b' \leftarrow \mathcal{A}^{\text{Enc}}$ Ret $(b' = \oplus_i \mathbf{b}[i])$	$C_1 \leftarrow \mathcal{E}(\mathbf{K}[i], M)$ $M_0 \leftarrow \{0, 1\}^{ M }; C_0 \leftarrow \mathcal{E}(\mathbf{K}[i], M_0)$ Ret $C_{\mathbf{b}[i]}$	Ret $(\mathbf{K}[i], \mathbf{b}[i])$

Fig. 2. Multi instance security notions for encryption

oracle **Enc** which, on input M , returns $\mathcal{E}(K, M)$. Finally, the adversary is asked to output a guess K' for the key, and the game returns **true** if $K = K'$, and **false** otherwise. An mi version of the game, $\text{UKU}_{\text{SE},m}$, is depicted in Figure 2. It picks an m -vector \mathbf{K} of target keys and the oracle **Enc** now takes i, M to return $\mathcal{E}(\mathbf{K}[i], M)$. The **Cor** oracle gives the adversary the capability of corrupting a user to obtain its target key. The adversary's output guess is also a m -vector \mathbf{K}' and the game returns the boolean ($\mathbf{K} = \mathbf{K}'$), meaning the adversary wins only if it recovers *all* the target keys. (The “U” in “UKU” reflects this, standing for “Universal.”) The advantage of adversary \mathcal{A} is $\text{Adv}_{\text{SE},m}^{\text{uku}}(\mathcal{A}) = \Pr[\text{UKU}_{\text{SE},m}^{\mathcal{A}} \Rightarrow \text{true}]$. Naturally, this advantage depends on the adversary's resources. (It could be 1 if the adversary corrupts all instances.) We say that \mathcal{A} is a (t, \mathbf{q}, q_c) -adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\text{Enc}(i, \cdot)$ and makes at most q_c corruption queries. Then we let $\text{Adv}_{\text{SE},m}^{\text{uku}}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \text{Adv}_{\text{SE},m}^{\text{uku}}(\mathcal{A})$ where the maximum is over all (t, \mathbf{q}, q_c) -adversaries.

AND. Single-instance indistinguishability for symmetric encryption is usually formalized via left-or-right security [4]. A random bit b and key $K \leftarrow \mathcal{K}$ are chosen, and an adversary \mathcal{A} is given access to an oracle **Enc** that given equal-length messages M_0, M_1 returns $\mathcal{E}(K, M_b)$. The adversary outputs a bit b' and its advantage is $2 \Pr[b = b'] - 1$. There are several ways one might consider creating an mi analog. Let us first consider a natural AND-based metric based on game $\text{AND}_{\text{SE},m}$ of Figure 2. It picks at random a vector $\mathbf{b} \leftarrow \{0, 1\}^m$ of challenge bits as well as a vector $\mathbf{K}[1], \dots, \mathbf{K}[m]$ of keys, and the adversary is given access to oracle **Enc** that on input i, M_0, M_1 , where $|M_0| = |M_1|$, returns $\mathcal{E}(\mathbf{K}[i], M_{\mathbf{b}[i]})$. Additionally, the corruption oracle **Cor** takes i and returns the pair $(\mathbf{K}[i], \mathbf{b}[i])$. The adversary finally outputs a bit vector \mathbf{b}' , and wins if and only if $\mathbf{b} = \mathbf{b}'$. (It is equivalent to test that $\mathbf{b}[i] = \mathbf{b}'[i]$ for all uncorrupted i .) The advantage of adversary \mathcal{A} is $\text{Adv}_{\text{SE},m}^{\text{and}}(\mathcal{A}) = \Pr[\text{AND}_{\text{SE},m}^{\mathcal{A}} \Rightarrow \text{true}] - 2^{-m}$. We say that \mathcal{A}

is a (t, \mathbf{q}, q_c) -adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i, \cdot, \cdot)$ and makes at most q_c corruption queries. Then we let $\mathbf{Adv}_{\mathsf{SE},m}^{\text{and}}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \mathbf{Adv}_{\mathsf{SE},m}^{\text{and}}(\mathcal{A})$ where the maximum is over all (t, \mathbf{q}, q_c) -adversaries.

This metric has many points in its favor. By (later) showing that security under it is implied by security under our preferred LORX metric, we automatically garner whatever value it offers. But the AND metric also has weaknesses that in our view make it inadequate as the primary choice. Namely, it does not capture the hardness of breaking *all* the uncorrupted instances. For example, an adversary that corrupts instances $1, \dots, m-1$ to get $\mathbf{b}[1], \dots, \mathbf{b}[m-1]$, makes a random guess g for $\mathbf{b}[m]$ and returns $(\mathbf{b}[1], \dots, \mathbf{b}[m-1], g)$ has the high advantage $0.5 - 2^{-m}$ without breaking all instances. We prefer a metric where this adversary's advantage is close to 0.

LORX. To overcome the above issue with the AND advantage, we introduce the XOR advantage measure and use it to define LORX. Game $\text{LORX}_{\mathsf{SE},m}$ of Figure 2 makes its initial choices the same way as game $\text{AND}_{\mathsf{SE},m}$ and provides the adversary with the same oracles. However, rather than a vector, the adversary must output a bit b' , and wins if this equals $\mathbf{b}[1] \oplus \dots \oplus \mathbf{b}[m]$. (It is equivalent to test that $b' = \oplus_{i \in S} \mathbf{b}[i]$ where S is the uncorrupted set.) The advantage of adversary \mathcal{A} is $\mathbf{Adv}_{\mathsf{SE},m}^{\text{lorx}}(\mathcal{A}) = 2 \Pr[\text{LORX}_{\mathsf{SE},m}^{\mathcal{A}} \Rightarrow \text{true}] - 1$. We say that \mathcal{A} is a (t, \mathbf{q}, q_c) -adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i, \cdot, \cdot)$ and makes at most q_c corruption queries. Then we let $\mathbf{Adv}_{\mathsf{SE},m}^{\text{lorx}}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \mathbf{Adv}_{\mathsf{SE},m}^{\text{lorx}}(\mathcal{A})$ where the maximum is over all (t, \mathbf{q}, q_c) -adversaries. In the example we gave for AND, if an adversary corrupts the first $m-1$ instances to get back $\mathbf{b}[1], \dots, \mathbf{b}[m-1]$, makes a random guess g for $\mathbf{b}[m]$ and outputs $b' = \mathbf{b}[1] \oplus \dots \oplus \mathbf{b}[m-1] \oplus g$, it will have advantage 0.

RORX. A variant of the si LOR notion, ROR, was given in [4]. Here the adversary must distinguish between an encryption of a message M it provides and the encryption of a random message of length $|M|$. This was shown equivalent to LOR up to a factor 2 in the advantages [4]. This leads us to define the mi analog RORX and ask how it relates to LORX. Game $\text{RORX}_{\mathsf{SE},m}$ of Figure 2 makes its initial choices the same way as game $\text{LORX}_{\mathsf{SE},m}$. The adversary is given access to oracle \mathbf{Enc} that on input i, M , returns $\mathcal{E}(\mathbf{K}[i], M)$ if $\mathbf{b}[i] = 1$ and otherwise returns $\mathcal{E}(\mathbf{K}[i], M_1)$ where $M_1 \leftarrow \{0, 1\}^{|M|}$. It also gets the usual \mathbf{Cor} oracle. It outputs a bit b' and wins if this equals $\mathbf{b}[1] \oplus \dots \oplus \mathbf{b}[m]$. The advantage of adversary \mathcal{A} is $\mathbf{Adv}_{\mathsf{SE},m}^{\text{rорx}}(\mathcal{A}) = 2 \Pr[\text{RORX}_{\mathsf{SE},m}^{\mathcal{A}} \Rightarrow \text{true}] - 1$. We say that \mathcal{A} is a (t, \mathbf{q}, q_c) -adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i, \cdot)$ and makes at most q_c corruption queries. Then we let $\mathbf{Adv}_{\mathsf{SE},m}^{\text{rорx}}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \mathbf{Adv}_{\mathsf{SE},m}^{\text{rорx}}(\mathcal{A})$ where the maximum is over all (t, \mathbf{q}, q_c) -adversaries.

DISCUSSION. The multi-user security goal from [3] gives rise to a version of the above games without corruptions and where all instances share the same challenge bit b , which the adversary tries to guess. But this does *not* measure mi security, since recovering a single key suffices to learn b .

The above approach extends naturally to providing a mi counterpart to any security definition based on a decisional game, where the adversary needs to guess a bit b . For example we may similarly create mi metrics of CCA security.

Why does the model include corruptions? The following example may help illustrate. Suppose SE is entirely insecure when the key has first bit 0 and highly secure otherwise. (From the si perspective, it is insecure.) In the LORX game, an adversary will be able to figure out around half the challenge bits. If we disallow corruptions, it would still have very low advantage. From the application point of view, this seems to send the wrong message. We want LORX-security to mean that the probability of “large scale” damage is low. But breaking half the instances is pretty large scale. Allowing corruptions removes this defect because the adversary could corrupt the instances it could not break and then, having corrupted only around half the instances, get a very high advantage, breaking LORX-security. In this way, we may conceptually keep the focus on an adversary goal of breaking *all* instances, yet cover the case of breaking some threshold number via the corruption capability.

An alternative way to address the above issue without corruptions is to define threshold metrics where the adversary wins by outputting a dynamically chosen set S and predicting the xor of the challenge bits for the indexes in S . This, again, has much going for it as a metric. But LORX with corruptions, as we define it, will imply security under this metric.

LORX IMPLIES UKU. In the si setting, it is easy to see that LOR security implies KU security. The LOR adversary simply runs the KU adversary. When the latter makes oracle query M , the LOR adversary queries its own oracle with M, M and returns the outcome to the KU adversary. When the latter returns a key K' , the LOR adversary submits a last oracle query consisting of a pair M_0, M_1 of random messages to get back a challenge ciphertext C , returning 1 if $\mathcal{D}(K', C) = M_1$ and 0 otherwise. A similar but slightly more involved proof shows that ROR implies KU.

It is important to establish analogs of these basic results in the mi setting, for they function as “tests” for the validity of our mi notions. The following shows that LORX security implies UKU. Interestingly, it is not as simple to establish in the mi case as in the si case. Also, as we will see later, the proof that RORX implies UKU is not only even more involved but incurs a factor 2^m loss, making LORX a better choice as the metric to target in designs.

Theorem 1. [LORX \Rightarrow UKU] *Let $\text{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with message space \mathcal{M} , and let ℓ be such that $\{0, 1\}^\ell \subseteq \mathcal{M}$. Then, for all t , q_c , and \mathbf{q} , and for all $k \geq 1$,*

$$\mathbf{Adv}_{\text{SE}, m}^{\text{uku}}(t, \mathbf{q}, q_c) \leq \mathbf{Adv}_{\text{SE}, m}^{\text{lorx}}(t', \mathbf{q}', q_c) + m \cdot \left(\frac{1}{2^\ell - 1} \right)^k,$$

where $t' = t + O(m \cdot k)$, and $\mathbf{q}'[i] = \mathbf{q}[i] + k$ for all $i = 1, \dots, m$. ■

The proof is given in [6]. Here, let us stress Theorem 1 surfaces yet another subtlety of the mi setting. At first, it would seem that proving the case $k = 1$

of the theorem is sufficient (this is what usually done in the si case). However, it is crucial to remark that $\mathbf{Adv}_{\mathbf{SE},m}^{\text{lorx}}(t', \mathbf{q}', q_c)$ may be *very* small. For example, it is not unreasonable to expect $2^{-128 \cdot m}$ if \mathbf{SE} is secure in the single-instance setting. Yet, assume that \mathcal{E} encrypts 128-bit messages, then we are only able to set $\ell = 128$, in turn making $m/(2^\ell - 1) \approx m \cdot 2^{-128}$ by far the leading term on the right-hand side. The parameter k hence opens the door to fine tuning of the additive extra term at the cost of an additive complexity loss in the reduction. Also note that the reduction in the proof of Theorem 1 is not immediate, as an adversary guessing all the keys in the UKU game with probability ϵ only yields an adversary recovering all the bits $\mathbf{b}[1], \dots, \mathbf{b}[m]$ in the LORX game with probability ϵ . Just outputting the xor of these bits is not sufficient, as we have to boost the success probability to $\frac{1+\epsilon}{2}$ in order to obtain the desired relation between the two advantage measures.

In analogy to the si setting, UKU does not imply LORX. Just take a scheme $\mathbf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ encrypting n -bit messages which is UKU-secure, and modify it into a scheme $\mathbf{SE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ where $\mathcal{K} = \mathcal{K}'$ and $\mathcal{E}'(K, M) = \mathcal{E}'(K, M) \parallel M[0]$, with $M[0]$ being the first bit of M . Clearly, \mathbf{SE}' is still UKU-secure but not LORX-secure.

As indicated above, a proof that RORX implies UKU is much more involved and incurs a factor 2^m loss. Roughly speaking, this is because in the si case, in the reduction needed to prove that ROR implies KU, the ROR adversary can only simulate the execution of the KU adversary correctly in the case where the bit is 1, i.e., the encryption oracle returns the actual encryption of a message. This results in a factor two loss in terms of advantage. Upon translating this technique to the mi case, the factor 2 becomes 2^m , as *all* bits need to be 1 for the UKU adversary to output the right keys with some guaranteed probability. However, we will not follow this route for the proof of this result. Instead, we can obtain the same result by combining Theorem 2 and Theorem 1.

LORX VERSUS RORX. In the si setting, LOR and ROR are the same up to a factor 2 in the advantage [4]. The LOR implies ROR implication is trivial and ROR implies LOR is a simple hybrid argument. We now discuss the relation between the mi counterparts, namely RORX and LORX, which is both more complex and more challenging to establish.

Theorem 2. [RORX \Rightarrow LORX] *Let $\mathbf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For all $m, t, q_c > 0$, and all vectors \mathbf{q} we have $\mathbf{Adv}_{\mathbf{SE},m}^{\text{lorx}}(t, \mathbf{q}, q_c) \leq 2^m \cdot \mathbf{Adv}_{\mathbf{SE},m}^{\text{rорx}}(t', \mathbf{q}, q_c)$, where $t' = t + \mathcal{O}(1)$.* ■

As discussed in Section 1, the multiplicative factor 2^m is often of no harm because advantages are already exponentially small in m . The factor is natural, being the mi analogue of the factor 2 appearing in the traditional si proof, and examples can be given showing that the bound is tight. The proof of the above is in [6]. The difficulty is adapting the hybrid argument technique to the mi setting. We omit the much simpler proof of the converse:

Theorem 3. [LORX \Rightarrow RORX] Let $\text{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For all $m, t, q_c > 0$, and all vectors \mathbf{q} we have $\text{Adv}_{\text{SE},m}^{\text{rora}}(t, \mathbf{q}, q_c) \leq \text{Adv}_{\text{SE},m}^{\text{lora}}(t', \mathbf{q}, q_c)$, where $t' = t + \mathcal{O}(1)$. ■

LORX IMPLIES AND. Intuitively, one might expect AND security to be a stronger requirement than LORX security, as the former seems easier to break than the latter. However we show that under a fairly minimal requirement, LORX implies AND. This brings another argument in support of LORX: Even if an application requires AND security, it turns out that proving LORX security is generally sufficient. The following theorem is to be interpreted as follows: In general, if we only know that $\text{Adv}_{\text{SE},m}^{\text{lora}}(t, \mathbf{q}, q_c)$ is small, we do not know how to prove $\text{Adv}_{\text{SE},m}^{\text{and}}(t', \mathbf{q}, q_c)$ is also small (for $t' \approx t$), or whether this is true at all. As we sketched above, the reason is that we do not know how to use an adversary \mathcal{A} for which the $\text{AND}_{\text{SE},m}$ advantage is large to construct an adversary for which the $\text{LORX}_{\text{SE},m}$ advantage is large. Still, one would expect that such an adversary *might* more easily yield one for which the $\text{LORX}_{\text{SE},k}$ advantage is sufficiently large, for *some* $k \leq m$. The following theorem uses a probabilistic lemma due to Unger [35] to confirm this intuition.

Theorem 4. Let $\text{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Further, let m, t, \mathbf{q} , and q_c be given, and assume that there exist C, ϵ , and γ such that for all $1 \leq i \leq m$,

$$\max_{S \subseteq \{1, \dots, m\}, |S|=i} \text{Adv}_{\text{SE},i}^{\text{lora}}(t_S^*, \mathbf{q}[S], q_c) \leq C \cdot \epsilon^i + \gamma,$$

where $\mathbf{q}[S]$ is the projection of \mathbf{q} on the components in S , and $t_S^* = t + \mathcal{O}(t_\mathcal{E} \cdot \sum_{i \notin S} \mathbf{q}[i])$, with $t_\mathcal{E}$ denoting the running time needed for one encryption with \mathcal{E} . Then, $\text{Adv}_{\text{SE},m}^{\text{and}}(t, \mathbf{q}, q_c) \leq \gamma + C \cdot \prod_{i=1}^m (1 + \epsilon_i)/2$. ■

We are not able to prove that the converse (AND implies LORX) is true in general, but in the absence of corruptions one can upper bound $\text{Adv}_{\text{SE},m}^{\text{lora}}(t, \mathbf{q}, 0)$ in terms of $\text{Adv}_{\text{SE},m'}^{\text{and}}(t', \mathbf{q}', 0)$ for $m' \approx 2m$ and t' and \mathbf{q}' being much larger than t, \mathbf{q} . The proof, which we omit, follows the lines of the proof of the XOR Lemma from the Direct Product Theorem given by [18], and relies on the Goldreich-Levin theorem [17]. As the loss in concrete security in this reduction is very large, and it only holds in the corruption-free case, we find this an additional argument to support the usage of the LORX metric.

3 Password-Based Encryption via KDFs

We now turn to our main motivating application, that of password based encryption (PBE) as specified in PKCS#5 [32]. The schemes specified there combine a conventional mode of operation (e.g., CBC mode) with a password-based key derivation function (KDF). We start with formalizing the latter.

PASSWORD-BASED KDFs. Formally, a (k, s, c) -KDF is a deterministic map $\text{KD} : \{0, 1\}^* \times \{0, 1\}^s \rightarrow \{0, 1\}^k$ that may make use of an underlying ideal primitive.

Here c is the iteration count, which specifies the multiplicative increase in work that should slow down brute force attacks.

PKCS#5 describes two KDFs [32]. We treat the first in detail and discuss the second in [6]. Let $\text{KD1}^H(pw, sa) = H^c(pw \parallel sa)$ where H^c is the function that composes H with itself c times. To generalize beyond concatenation, we can define a function $\text{Encode}(pw, sa)$ that describes how to encode its inputs onto $\{0, 1\}^*$ with efficiently computable inverse $\text{Decode}(W)$.

PBE SCHEMES. A PBE scheme is just a symmetric encryption scheme where we view the keys as passwords and key generation as a password sampling algorithm. To highlight when we are thinking of key generation as password sampling we will use \mathcal{P} to denote key generation (instead of \mathcal{K}). We will also write pw for a key that we think of as a password. Let KD be a (k, s, c) -KDF and let $\text{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme with \mathcal{K} outputting uniformly selected k -bit keys. Then we define the PBE scheme $\mathcal{SE}[\text{KD}, \text{SE}] = (\mathcal{P}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ as follows. Encryption $\overline{\mathcal{E}}(pw, M)$ is done via $sa \leftarrow \{0, 1\}^s; K \leftarrow \text{KD}(pw, sa); C \leftarrow \mathcal{E}(K, M)$, returning (sa, C) as the ciphertext. Decryption recomputes the key K by reapplying the KDF and then applies \mathcal{D} . If the KDF is KD1 and the encryption scheme is CBC mode, then one obtains the first PBE scheme from PKCS#5 [32].

PASSWORD GUESSING. We aim to show that security of the above constructions holds up to the amount of work required to brute-force the passwords output by \mathcal{P} . This begs the question of how we measure the strength of a password sampler. We will formalize the hardness of guessing passwords output by some sampler \mathcal{P} via an adaptive guessing game: It challenges an adversary with guessing passwords adaptively in a setting where the attacker may, also, adaptively learn some passwords via a corruption oracle. Concretely, let $\text{GUESS}_{\mathcal{P}, m}$ be the game defined in Figure 3. A (q_t, q_c) -guessing adversary is one that makes at most q_t queries to **Test** and q_c queries to **Cor**. An adversary \mathcal{B} 's guessing advantage is $\text{Adv}_{\mathcal{P}, m}^{\text{guess}}(\mathcal{B}) = \Pr[\text{GUESS}_{\mathcal{P}, m}^{\mathcal{B}} \Rightarrow \text{true}]$. We assume without loss of generality that \mathcal{A} does not make any *pointless queries*: (1) repeated queries to **Cor** on the same value; (2) a query **Test**(i, \cdot) following a query of **Cor**(i); and (3) a query **Cor**(i) after a query **Test**(i, pw) that returned true. We also define a variant of the above guessing game that includes salts and allows an attacker to test password-salt pairs against all m instances simultaneously. This will be useful as an intermediate step when reducing to guessing advantage. The game $\text{saGUESS}_{\mathcal{P}, m, \rho}$ is shown in Figure 3 and we define advantage via $\text{Adv}_{\mathcal{P}, m}^{\text{sa-guess}}(\mathcal{B}) = \Pr[\text{saGUESS}_{\mathcal{P}, m}^{\mathcal{B}} \Rightarrow \text{true}]$. An easy argument proves the following lemma.

Lemma 5. *Let $m, \rho > 0$, let \mathcal{P} be a password sampler and let \mathcal{A} be an (q_t, q_c) -guessing $\text{GUESS}_{\mathcal{P}, m}$ adversary. Then there is a (q_t, q_c) -guessing $\text{saGUESS}_{\mathcal{P}, m, \rho}$ adversary \mathcal{B} such that $\text{Adv}_{\mathcal{P}, m, \rho}^{\text{sa-guess}}(\mathcal{A}) \leq \text{Adv}_{\mathcal{P}, m}^{\text{guess}}(\mathcal{B}) + m^2 \rho^2 / 2^s$. \square*

SAMPLERS WITH HIGH MIN-ENTROPY. Even though the guessing advantage precisely quantifies strength of password samplers, good upper bounds in terms of the adversary's complexity and of some simpler relevant parameters of a

<u>main</u> GUESS $_{\mathcal{P},m}$	<u>proc.</u> Test(i, pw)	<u>proc.</u> Cor(i)
$pw[1], \dots, pw[m] \leftarrow_s \mathcal{P}$	If ($pw = pw[i]$) then Ret true	Ret $pw[i]$
$pw' \leftarrow_s \mathcal{B}^{\text{Test}, \text{Cor}}$	Ret \perp	
Ret $\bigwedge_{i=1}^m (pw'[i] = pw[i])$		
<u>main</u> saGUESS $_{\mathcal{P},m,\rho}$	<u>proc.</u> Test(pw, sa)	<u>proc.</u> Cor(i)
$pw[1], \dots, pw[m] \leftarrow_s \mathcal{P}$	For $i = 1$ to m do	Ret $pw[i]$
For $i = 1$ to m do	For $j = 1$ to ρ do	
For $j = 1$ to ρ do	If (pw, sa) = ($pw[i], sa[i, j]$) then	
$sa[i, j] \leftarrow_s \{0, 1\}^s$	Ret (i, j)	
$pw' \leftarrow_s \mathcal{B}^{\text{Test}, \text{Cor}}(sa)$	Ret (\perp, \perp)	
Ret $\bigwedge_{i=1}^m (pw'[i] = pw[i])$		

Fig. 3. An adaptive password-guessing game

password sampler are desirable. One interesting case is samplers with high min-entropy. Formally, we say that \mathcal{P} has min-entropy μ if for all pw' it holds that $\Pr[pw = pw'] \leq 2^{-\mu}$ over the coins used in choosing $pw \leftarrow_s \mathcal{P}$.

Theorem 6. Fix $m \geq q_c \geq 0$ and a password sampler \mathcal{P} with min-entropy μ . Let \mathcal{B} be a (q_t, q_c) -adversary for GUESS $_{\mathcal{P},m}$ making q_i queries of the form $\text{Test}(i, \cdot)$ with $q_t = q_1 + \dots + q_m$. Let $\delta = q_t/(m2^\mu)$ and let $\gamma = (m - q_c)/m$. Then $\mathbf{Adv}_{\mathcal{P},m}^{\text{guess}}(\mathcal{B}) \leq e^{-m\Delta(\gamma, \delta)}$ where $\Delta(\gamma, \delta) = \gamma \ln(\frac{\gamma}{\delta}) + (1 - \gamma) \ln(\frac{1-\gamma}{1-\delta})$. \square

Using $\Delta(\gamma, \delta) \geq 2(\gamma - \delta)^2$, we see that to win the guessing game for q_c corruptions, $q_t \approx (m - q_c) \cdot 2^\mu$ **Test** queries are necessary, and the brute-force attack is optimal. Note that the above bound is the best we expect to prove: Indeed, assume for a moment that we restrict ourselves to adversaries that want to recover a subset of $m - q_c$ passwords, without corruptions, and make q_t/m queries $\text{Test}(i, \cdot)$, for each i , which are independent from queries $\text{Test}(j, \cdot)$ for other $j \neq i$. Then, each individual password is found, independently, with probability at most $q_t/(m \cdot 2^\mu)$, and if one applies the Chernoff bound, the probability that a subset of size $m - q_c$ of the passwords are retrieved is upper bounded by $e^{-m\Delta(\gamma, \delta)}$. In our case, we have additional challenges: Foremost, queries for each i are not independent. Also, the number of queries may not be the same for each index i . And finally, we allow for corruption queries.

The full proof of Theorem 6 is given in [6]. At a high level, it begins by showing how to move to a simpler setting in which the adversary wins by recovering a subset of the passwords without the aid of a corrupt oracle. The resulting setting is an example of a threshold direct product game. This allows us to apply a *generalized* Chernoff bound due to Panconesi and Srinivasan [31] (see also [20]) that reduces threshold direct product games to (non-threshold) direct product games. Finally, we apply an amplification lemma due to Maurer, Pietrzak, and Renner [25] that yields a direct product theorem for the password guessing game. Let us also note that using the same technique, the better bound $\mathbf{Adv}_{\mathcal{P},m}^{\text{guess}}(\mathcal{B}) \leq (q_t/m2^\mu)^m$ can be proven for the special case of $(q_t, 0)$ -adversaries.

CORRELATED PASSWORDS. By taking independent samples from \mathcal{P} we have captured only the setting of independent passwords. In practice, of course, passwords may be correlated across users or, at least, user accounts. Our results extend to the setting of jointly selecting a vector of m passwords, except of course the analysis of the guessing advantage (whose proof fundamentally relies upon independence). This last only limits our ability to measure, in terms of simpler metrics like min-entropy, the difficulty of a guessing game against correlated passwords. This does not decrease the security proven, as the simulation-based paradigm we introduce below allows one to reduce to the full difficulty of the guessing game.

SIMULATION-BASED SECURITY FOR KDFs. We define an ideal-functionality style notion of security for KDFs. Figure 4 depicts two games. A message sampler \mathcal{M} is an algorithm that takes input a number r and outputs a pair of vectors $(\mathbf{pw}, \mathbf{sa})$ each having r elements and with $|\mathbf{sa}[i]| = s$ for $1 \leq i \leq r$. A simulator S is a randomized, stateful procedure. It expects oracle access to a procedure **Test** to which it can query a message. Game $\text{Real}_{\text{KD}, \mathcal{M}, r}^{\mathcal{D}}$ gives a distinguisher \mathcal{D} the messages and associated derived keys. Also, \mathcal{D} can adaptively query the ideal primitive H underlying KD. Game $\text{Ideal}_{\mathcal{S}, \mathcal{M}, r}^{\mathcal{D}}$ gives \mathcal{D} the messages and keys chosen uniformly at random. Now \mathcal{D} can adaptively query a primitive oracle implemented by a simulator S that, itself, has access to a **Test** oracle. Then we define KDF advantage by

$$\mathbf{Adv}_{\text{KD}, \mathcal{M}, r}^{\text{kdf}}(\mathcal{D}, S) = \Pr \left[\text{Real}_{\text{KD}, \mathcal{M}, r}^{\mathcal{D}} \Rightarrow 1 \right] - \Pr \left[\text{Ideal}_{\mathcal{S}, \mathcal{M}, r}^{\mathcal{D}} \Rightarrow 1 \right].$$

To be useful, we will require proving that there exists a simulator S such that for any \mathcal{D}, \mathcal{M} pair the KDF advantage is “small”.

This notion is equivalent to applying the indistinguishability framework [26] to a particular ideal KDF functionality. That functionality chooses messages according to an algorithm \mathcal{M} and outputs on its honest interface the messages and uniform keys associated to them. On the adversarial interface is the test routine which allows the simulator to learn keys associated to messages. This raises the question of why not just use indistinguishability from a RO as our target security notion. The reasons are two-fold. First, it is not clear that H^c is indistinguishable from a random oracle. Second, even if it were, a proof would seem to require a simulator that makes at least the same number of queries to the RO as it receives from the distinguisher. This rules out showing security amplification due to the iteration count c . Our approach solves both issues, since we will show KDF security for simulators that make one call to **Test** for every c made to it. For example, our simulator for KD1 will only query **Test** if a chain of c hashes leads to the being-queried point X and this chain is not a continuation of some longer chain. We formally capture this property of simulators next.

c -AMPLIFYING SIMULATORS. Let $\tau = (X_1, Y_1), \dots, (X_q, Y_q)$ be a (possibly partial) transcript of **Prim** queries and responses. We restrict attention to (k, s, c) -KDFs for which we can define a predicate $\text{final}_{\text{KD}}(X_i, \tau)$ which evaluates to true if there exists exactly one sequence of c indices $j_1 < \dots < j_c$ such that (1) $j_c = i$, (2) there exist unique $(\mathbf{pw}, \mathbf{sa})$ such that evaluating $\text{KD}^H(\mathbf{pw}, \mathbf{sa})$ when H is such

main $\text{Real}_{\text{KD}, \mathcal{M}, r}$ $(\text{pw}, \text{sa}) \leftarrow \mathcal{M}(r)$ For $i = 1$ to r do $\mathbf{K}[i] \leftarrow \text{KD}^H(\text{pw}[i], \text{sa}[i])$ $b' \leftarrow \mathcal{D}^{\text{Prim}}(\text{pw}, \text{sa}, \mathbf{K})$ Ret b' proc. $\text{Prim}(X)$ Ret $H(X)$	main $\text{Ideal}_{S, \mathcal{M}, r}$ $(\text{pw}, \text{sa}) \leftarrow \mathcal{M}(r)$ For $i = 1$ to r do $\mathbf{K}[i] \leftarrow \{0, 1\}^k$ $b' \leftarrow \mathcal{D}^{\text{Prim}}(\text{pw}, \text{sa}, \mathbf{K})$ Ret \perp Ret b' proc. $\text{Prim}(X)$ Ret $S^{\text{Test}}(X)$	sub. $\text{Test}(\text{pw}, \text{sa})$ For $i = 1$ to r do If $(\text{pw}[i], \text{sa}[i]) = (\text{pw}, \text{sa})$ then Ret $\mathbf{K}[i, j]$ Ret \perp
--	--	---

Fig. 4. Games for the simulation-based security notion for KDFs

that $Y_j = H(X_j)$ for $1 \leq j \leq i$ results exactly in the queries X_{j_1}, \dots, X_{j_c} in any order where X_i is the last query, and (3) $\text{final}_{\text{KD}}(X_{j_r}, \tau) = \text{false}$ for all $r < c$.

Our simulators only query **Test** on queries X_i for which $\text{final}_{\text{KD}}(X_i, \tau) = \text{true}$; we call such queries **KD-completion queries** and simulators satisfying this are called *c-amplifying*. Note that (3) implies that there are at most q/c total KD-completion queries in a q -query transcript.

HASH-DEPENDENT PASSWORDS. We do not allow \mathcal{M} access to the random oracle H . This removes from consideration hash-dependent passwords. Our results should extend to cover hash-dependent passwords if one has explicit domain separation between use of H during password selection and during key derivation. Otherwise, an indistinguishability-style approach as we use here will not work due to limitations pointed out in [33]. A full analysis of the hash-dependent password setting would therefore appear to require direct analysis of PBE schemes without taking advantage of the modularity provided by simulation-based approaches.

SECURITY OF KD1. For a message sampler \mathcal{M} , let $\gamma(\mathcal{M}, r) := \Pr[\exists i \neq j : (\text{pw}[i], \text{sa}[i]) = (\text{pw}[j], \text{sa}[j])]$ where $(\text{pw}, \text{sa}) \leftarrow \mathcal{M}(r)$. We prove the following theorem in [6].

Theorem 7. Fix $r > 0$. Let KD1 be as above. There exists a simulator S such that for all adversaries \mathcal{D} making q RO queries, of which q_c are chain completion queries, and all message samplers \mathcal{M} ,

$$\mathbf{Adv}_{\text{KD1}, \mathcal{M}, r}^{\text{kdf}}(\mathcal{D}, S) \leq 4\gamma(\mathcal{P}, r) + \frac{2r^2 + 7(2q + rc)^2}{2^n}.$$

The simulator S makes at most q_c **Test** queries, and answers each query in time $O(c)$. \square

SECURITY OF PBE. We are now in a position to analyze the security of password based encryption as used in PKCS#5. The following theorem, proved in [6], uses the multi-user left-or-right security notion from [3] whose formalization is recalled in [6]:

Theorem 8. Let $m \geq 1$, let $\mathcal{SE}[\text{KD}, \text{SE}] = (\mathcal{P}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the encryption scheme built from an (k, s, c) -KDF KD and an encryption scheme $\text{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with

k-bit keys. Let \mathcal{A} be an adversary making ρ queries to $\mathbf{Enc}(i, \cdot, \cdot)$ for each $i \in \{1, \dots, m\}$ and making at most $q_c < m$ corruption queries. Let S be a c -amplifying simulator. Then there exists message sampler \mathcal{M} and adversaries \mathcal{D}, \mathcal{C} , and \mathcal{B} such that

$$\mathbf{Adv}_{SE,m}^{\text{lorx}}(\mathcal{A}) \leq m \cdot \mathbf{Adv}_{SE,\rho}^{\text{mu-lor}}(\mathcal{C}) + 2 \cdot \mathbf{Adv}_{P,m,\rho}^{\text{guess}}(\mathcal{B}) + 2 \cdot \mathbf{Adv}_{KD,M,m\rho}^{\text{kdf}}(\mathcal{D}, S)$$

If \mathcal{A} makes q queries to H , then: \mathcal{D} makes at most q queries to its H oracle; \mathcal{B} makes at most $\lceil q/c \rceil$ queries to **Test** and at most q_c corruption queries; and \mathcal{C} makes a single query $\mathbf{Enc}(i, \cdot, \cdot)$ for each $1 \leq i \leq \rho$. Moreover, \mathcal{C} 's running time equals $t_{\mathcal{A}} + q \cdot t_S$ plus a small, absolute constant, and where $t_{\mathcal{A}}$ is the running time of \mathcal{A} , and t_S is the time needed by S to answer a query. Finally, $\gamma(\mathcal{M}, m\rho) \leq m^2 \rho^2 / 2^s$. \square

Note that the theorem holds even when **SE** is only one-time secure (meaning it can be deterministic), which implies that the analysis covers tools such as WinZip (c.f., [22]). In terms of the bound we achieve, Theorem 7 for KD1 shows that an adversary that makes $\mathbf{Adv}_{KD,P^*,m\rho}^{\text{kdf}}(\mathcal{D}, S)$ large requires $q \approx 2^{n/2}$ queries to H , provided salts are large. If H is SHA-256 then this is about 2^{128} work. Likewise, a good choice of **SE** will ensure that $\mathbf{Adv}_{SE,K,\rho}^{\text{mu-lor}}(\mathcal{C})$ will be very small. Thus the dominating term ends up the guessing advantage of \mathcal{B} against \mathcal{P} , which measures its ability to guess $m - q_c$ passwords.

Acknowledgments. Bellare was supported in part by NSF grants CCF-0915675, CNS-0904380 and CNS-1116800. Ristenpart was supported in part by NSF grant CNS-1065134. Tessaro was supported in part by NSF grants CCF-0915675, CCF-1018064, and DARPA contracts FA8750-11-C-0096, FA8750-11-2-0225.

References

1. Abadi, M., Warinschi, B.: Password-Based Encryption Analyzed. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 664–676. Springer, Heidelberg (2005)
2. Baudron, O., Pointcheval, D., Stern, J.: Extended Notions of Security for Multi-cast Public Key Cryptosystems. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 499–511. Springer, Heidelberg (2000)
3. Bellare, M., Boldyreva, A., Micali, S.: Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In: Pneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS, pp. 394–403. IEEE Computer Society Press (October 1997)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Pneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Bellare, M., Ristenpart, T., Tessaro, S.: Multi-instance security and its application to password-based cryptography. Cryptology ePrint Archive, Report 2012/196 (2012), <http://eprint.iacr.org/>

7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press (November 1993)
8. Boyen, X.: Halting password puzzles: hard-to-break encryption from human-memorable keys. In: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, p. 9. USENIX Association (2007)
9. Boyen, X.: New Paradigms for Password Security (Abstract from the Keynote Lecture). In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 1–5. Springer, Heidelberg (2008)
10. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
11. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
12. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004)
13. Dodis, Y., Impagliazzo, R., Jaiswal, R., Kabanets, V.: Security Amplification for *Interactive* Cryptographic Primitives. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 128–145. Springer, Heidelberg (2009)
14. Gennaro, R., Lindell, Y.: A Framework for Password-Based Authenticated Key Exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)
15. Goldreich, O.: Three XOR-Lemmas — An exposition (1995), <http://www.wisdom.weizmann.ac.il/>
16. Goldreich, O., Impagliazzo, R., Levin, L.A., Venkatesan, R., Zuckerman, D.: Security preserving amplification of hardness. In: 31st FOCS, pp. 318–326. IEEE Computer Society Press (October 1990)
17. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC, pp. 25–32. ACM Press (May 1989)
18. Goldreich, O., Nisan, N., Wigderson, A.: On Yao’s XOR-Lemma. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. LNCS, vol. 6650, pp. 273–301. Springer, Heidelberg (2011)
19. Haitner, I., Harnik, D., Reingold, O.: On the Power of the Randomized Iterate. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 22–40. Springer, Heidelberg (2006)
20. Impagliazzo, R., Kabanets, V.: Constructive Proofs of Concentration Bounds. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 617–631. Springer, Heidelberg (2010)
21. Kelsey, J., Schneier, B., Hall, C., Wagner, D.: Secure Applications of Low-Entropy Keys. In: Okamoto, E., Davida, G., Mambo, M. (eds.) ISW 1997. LNCS, vol. 1396, pp. 121–134. Springer, Heidelberg (1998)
22. Kohno, T.: Attacking and repairing the winZip encryption scheme. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 72–81. ACM Press (October 2004)
23. Krawczyk, H.: Cryptographic Extraction and Key Derivation: The HKDF Scheme. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 631–648. Springer, Heidelberg (2010)
24. Luby, M., Rackoff, C.: A Study of Password Security. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 392–397. Springer, Heidelberg (1988)

25. Maurer, U.M., Pietrzak, K., Renner, R.: Indistinguishability Amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
26. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
27. Maurer, U., Tessaro, S.: Computational Indistinguishability Amplification: Tight Product Theorems for System Composition. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 355–373. Springer, Heidelberg (2009)
28. Maurer, U., Tessaro, S.: A Hardcore Lemma for Computational Indistinguishability: Security Amplification for Arbitrarily Weak PRGs with Optimal Stretch. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 237–254. Springer, Heidelberg (2010)
29. Morris, R., Thompson, K.: Password security: a case history. Commun. ACM 22, 594–597 (1979)
30. Myers, S.: Efficient amplification of the security of weak pseudo-random function generators. Journal of Cryptology 16(1), 1–24 (2003)
31. Panconesi, A., Srinivasan, A.: Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. SIAM J. Comput. 26(2), 350–368 (1997)
32. PKCS #5: Password-based cryptography standard (rfc 2898). RSA Data Security, Inc., Version 2.0 (September 2000)
33. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with Composition: Limitations of the Indifferentiability Framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011)
34. Tessaro, S.: Security Amplification for the Cascade of Arbitrarily Weak PRPs: Tight Bounds via the Interactive Hardcore Lemma. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 37–54. Springer, Heidelberg (2011)
35. Unger, F.: A probabilistic inequality with applications to threshold direct-product theorems. In: 50th FOCS, pp. 221–229. IEEE Computer Society Press (October 2009)
36. Wagner, D., Goldberg, I.: Proofs of Security for the Unix Password Hashing Algorithm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 560–572. Springer, Heidelberg (2000)
37. Yao, A.C.: Theory and applications of trapdoor functions. In: 23rd FOCS, pp. 80–91. IEEE Computer Society Press (November 1982)
38. Yao, F.F., Yin, Y.L.: Design and Analysis of Password-Based Key Derivation Functions. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 245–261. Springer, Heidelberg (2005)

Hash Functions Based on Three Permutations: A Generic Security Analysis

Bart Mennink and Bart Preneel

Dept. Electrical Engineering, ESAT/COSIC, KU Leuven, and IBBT, Belgium
`{bart.mennink,bart.preneel}@esat.kuleuven.be`

Abstract. We consider the family of $2n$ -to- n -bit compression functions that are solely based on at most three permutation executions and on XOR-operators, and analyze its collision and preimage security. Despite their elegance and simplicity, these designs are not covered by the results of Rogaway and Steinberger (CRYPTO 2008). By defining a carefully chosen equivalence relation on this family of compression functions, we obtain the following results. In the setting where the three permutations π_1, π_2, π_3 are selected independently and uniformly at random, there exist at most four equivalence classes that achieve optimal $2^{n/2}$ collision resistance. Under a certain extremal graph theory based conjecture, these classes are then proven optimally collision secure. Three of these classes allow for finding preimages in $2^{n/2}$ queries, and only one achieves optimal $2^{2n/3}$ preimage resistance (with respect to the bounds of Rogaway and Steinberger, EUROCRYPT 2008). Consequently, a compression function is optimally collision and preimage secure if and only if it is equivalent to $F(x_1, x_2) = x_1 \oplus \pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1))$. For compression functions that make three calls to the same permutation we obtain a surprising negative result, namely the impossibility of optimal $2^{n/2}$ collision security: for any scheme, collisions can be found with $2^{2n/5}$ queries. This result casts some doubt over the existence of any (larger) secure permutation-based compression function built only on XOR-operators and (multiple invocations of) a single permutation.

Keywords: Hash function, Permutation-based, Collision resistance, Preimage resistance.

1 Introduction

The traditional recipe for the design of a cryptographic hash function is to base it on one or more block ciphers. Since the late 70s, this methodology developed itself to become the dominating approach in the area of hash function design and plenty of hash functions have been constructed accordingly (either explicitly or implicitly) [3,4,7,8]. These designs are, however, characterized by the fact that the key input to the cipher depends on the input values; this implies that the key schedule has to be strong and that it needs to be executed for every encryption (or for every second encryption), which entails a substantial computational cost. An alternative approach is to fix one or more keys, and restrict the hash function

design to use the block cipher for these keys only. The usage of fixed-key block ciphers, or alternatively *permutations*, additionally causes gain that one does not need to implement an entire block cipher but only a limited number of instantiations of it.

Black, Cochran and Shrimpton [1] were the first to formally study this approach, demonstrating that a $2n$ -to- n -bit compression function F using one n -bit permutation π cannot be secure. This result has been generalized by Rogaway and Steinberger [11], and refined by Stam [13] and Steinberger [14]. Consider any mn -to- rn -bit compression function using k n -bit permutations: if $2^{n(2m-2r-k+1)/(k+1)} \geq 17$, collisions can be found in at most $(2^n)^{1-(m-r+1)/(k+1)}$ queries to the underlying primitives, a bound proven by Steinberger in [14] but commonly known as “Stam’s bound.” Collisions and preimages can even be found in at most $(2^n)^{1-(m-r/2)/k}$ and $(2^n)^{1-(m-r)/k}$ queries respectively, provided the compression function satisfies the “uniformity assumption” [11]. Due to Stam’s bound, a $2n$ -to- n -bit compression function, which is the simplest case after all, achieves optimal $2^{n/2}$ collision resistance *only if* it employs at least three permutations. Yet, it cannot achieve optimal preimage resistance if it fulfills the uniformity assumption. These observations apply to the “multi-permutation setting”, where each of the permutations is generated independently, as well as the “single-permutation setting” where the permutations are the same.

The construction of $2n$ -to- n -bit compression functions (based on three permutations) that provably attain optimal collision security, has turned out to be a very challenging exercise. In [10], Rogaway and Steinberger formally proved a broad class of $2n$ -to- n -bit compression functions using three distinct permutations and finite field scalar multiplications optimally collision and preimage secure (w.r.t. the bounds of [11]), provided the compression function satisfies a so-called “independence criterion” (a similar result for the single-permutation setting has been obtained by Lee and Kwon [5]). Unfortunately, this technical criterion rules out the most intuitive and elegant type of designs, namely compression functions that are (apart from the three permutations) solely based on XOR-operators. As the proof of [10] extensively relies on its independence criterion, the proof cannot be generalized to compression functions of this type. In [12], Shrimpton and Stam derived a XOR-based compression function, using three one-way functions rather than permutations: $F(x_1, x_2) = f_1(x_1) \oplus f_3(f_1(x_1) \oplus f_2(x_2))$. This function is proven collision resistant up to $2^{n/2}$ queries (asymptotically), but preimages can be found with high probability after $2^{n/2}$ queries [12]. It has been demonstrated by an automated analysis of Rogaway and Steinberger [10] that the same results hold if f_1, f_2, f_3 are Davies-Meyer-like compression functions using permutations π_1, π_2, π_3 , i.e. $f_i(x) = x \oplus \pi_i(x)$, but a formal security analysis has never been given. Since these works, a synthetic formal collision and preimage security analysis of XOR-based compression functions has remained an interesting and important theoretical open problem, because of their elegance and simplicity (the functions only employ XOR-operators) as well as their slight efficiency improvement (XOR-operators are slightly cheaper than finite field multiplications).

OUR CONTRIBUTIONS. We focus on the entire family of $2n$ -to- n -bit compression functions constructed only of three isolated permutations and of XOR-operators, and analyze the security of these functions against information-theoretic adversaries. For each of the functions, we either provide a proof of optimal collision resistance or a collision attack faster than the birthday bound. We also analyze the preimage resistance of the schemes that have optimal collision security.

The approach followed in this work is based on defining an equivalence class on the set of compression functions, and is of independent interest: informally, two compression functions are equivalent if there exists a tight bi-directional preimage and collision security reduction (cf. Def. 3). Consequently, security results of one compression function hold for the entire class, and it suffices to analyze the security of one function per class. In this work we restrict to equivalence reductions that are easy to verify, such as interchanging the inputs to the compression function.

For the *multi-permutation* setting, where the three permutations π_1, π_2, π_3 are assumed to be selected independently and uniformly at random, the results are as follows. A compression function F is optimally collision secure (asymptotically) *if and only if* it is equivalent to one of the four compression functions F_1, \dots, F_4 :

$$\begin{aligned} F_1(x_1, x_2) &= x_2 \oplus \pi_2(x_2) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1)), \\ F_2(x_1, x_2) &= x_1 \oplus \pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1)), \\ F_3(x_1, x_2) &= x_1 \oplus \pi_1(x_1) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2)), \\ F_4(x_1, x_2) &= x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2)). \end{aligned} \quad (1)$$

These compression functions are depicted in Fig. 1. Not surprisingly, the permutation-based variant of the Shrimpton-Stam compression function [12] is included, it equals F_3 . For compression functions non-equivalent to any of F_1, F_2, F_3, F_4 , collisions can be found faster than the birthday bound, namely in at most $2^{2n/5}$ queries. Compression functions equivalent to F_2 are proven optimally preimage secure up to $2^{2n/3}$ queries, and compression functions equivalent to F_1, F_3 or F_4 are additionally shown to achieve tight $2^{n/2}$ preimage security. Therefore, a compression function achieves optimal collision and preimage resistance (w.r.t. the bounds of [11]) if and only if it is equivalent to F_2 . Particularly,

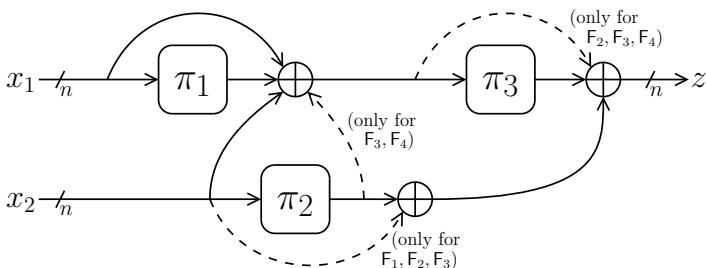


Fig. 1. A graphical representation of the compression functions F_1, \dots, F_4 of (1)

this class of functions beats the Shrimpton-Stam compression function [12] with respect to preimage resistance. These results are summarized in Table 1.

A minor part of the results in the multi-permutation setting, more concretely the collision resistance of F_1, F_2 and F_4 and the preimage resistance of F_2 , are based on an extremal graph theory based conjecture. Informally, this conjecture bounds the number of solutions $(x_1, x_2, x_3) \in X_1 \times X_2 \times X_3$ such that $x_2 \oplus x_3 = x_1 \oplus \pi_1(x_1)$, where X_1, X_2, X_3 are three sets of q elements. This conjecture is similar to (but more complex than) a problem posed by Zarankiewicz in 1951 (cf. [2, Ch. 6.2]), and is of independent interest. In the full version of this paper [6], we analyze our conjecture in more detail, provide it with a heuristic argument, and compare it with the conjecture of Zarankiewicz.

Table 1. The security results of this work for the multi-permutation setting. The functions F_1, \dots, F_4 are given in (1) and Fig. 1. The equivalence relation is defined in Def. 3. For F_2 , the obtained security results are optimal with respect to the bounds of Rogaway and Steinberger [11]. The proofs of the results with appended “[c]” fall back on Conjecture 1.

F equivalent to:	collision		preimage	
	security	attack	security	attack
F_1, F_4	$2^{n/2}$ [c]	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$
F_2	$2^{n/2}$ [c]	$2^{n/2}$	$2^{2n/3}$ [c]	$2^{2n/3}$
F_3	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$
none of these	?	$2^{2n/5}$?	?

In the *single-permutation* setting, where the compression function makes three calls to the same random permutation π , *there does not exist any* compression function that achieves optimal collision resistance. In particular, for any possible function, collisions can be found in at most $2^{2n/5}$ queries, beating the desired birthday bound. This negative result is surprising, given the fair amount of secure functions we have found in the multi-permutation setting. The attacks mainly rely on the fact that the adversary can misuse the single-permutation property by introducing dependencies between the two input values x_1 and x_2 . For instance, the function F_2 of (1) satisfies $F_2(x_1, x_2) = F_2(x_1, x_2 \oplus x_1 \oplus \pi(x_1))$ in the single-permutation setting. This result raises the interesting question whether (larger) compression functions exist based only on XOR-operators and (more than three invocations of) one single permutation.

OUTLINE. In Sect. 2, we present some background information, and formally describe the set of permutation-based compression functions we have analyzed. In Sect. 3, the equivalence relation on the set of compression functions is formally defined. The main results are given in Sect. 4 for the multi-permutation setting and in Sect. 5 for the single-permutation setting. We conclude the paper in Sect. 6.

2 Preliminaries

For an integer $n \in \mathbb{N}$, we denote by $\{0, 1\}^n$ the set of bit strings of length n . For two bit strings x, y , we denote by $x \| y$ their concatenation and by $x \oplus y$ their bitwise XOR. If \mathcal{X} is a set, by $x \xleftarrow{\$} \mathcal{X}$ we denote the uniformly random sampling of an element from \mathcal{X} . For two integers $m, n \in \mathbb{N}$, we denote by $\langle m \rangle_n$ the encoding of m as an n -bit string. By \log we denote the logarithm function with respect to base 2. By P_n we denote the set of all permutations operating on n bits. Vectors are denoted as \mathbf{x} , and by $\|\mathbf{x}\| = \sum_i |x_i|$ we denote the 1-norm of \mathbf{x} . For a matrix A , by $a_{i,j}$ we denote its coefficient at the i^{th} row and j^{th} column. By $\mathbf{a}_{i,*}$ we denote the i^{th} row of A , and by $\mathbf{a}_{*,j}$ its j^{th} column.

2.1 Permutation Based Compression Functions

We consider the following type of $2n$ -to- n -bit compression functions. Let $\pi_1, \pi_2, \pi_3 \in P_n$ be three permutations. For a binary 4×5 matrix A of the form

$$A = \left(\begin{array}{cc|ccc} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \end{array} \right), \quad (2)$$

the compression function $F_A : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ is defined as follows:

$$\begin{aligned} F_A(x_1, x_2) = z, \text{ where } y_1 &\leftarrow \pi_1(a_{11}x_1 \oplus a_{12}x_2), \\ y_2 &\leftarrow \pi_2(a_{21}x_1 \oplus a_{22}x_2 \oplus a_{23}y_1), \\ y_3 &\leftarrow \pi_3(a_{31}x_1 \oplus a_{32}x_2 \oplus a_{33}y_1 \oplus a_{34}y_2), \\ z &\leftarrow a_{41}x_1 \oplus a_{42}x_2 \oplus a_{43}y_1 \oplus a_{44}y_2 \oplus a_{45}y_3. \end{aligned} \quad (3)$$

The function F_A is depicted in Fig. 2. If the three permutations are all different, we refer to it as the *multi-permutation* setting. If π_1, π_2, π_3 are equal to

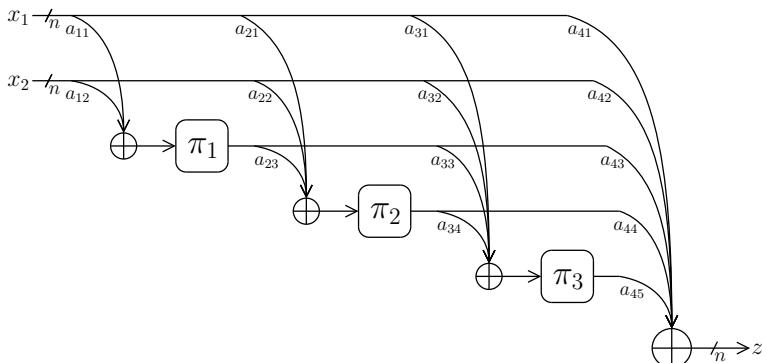


Fig. 2. The permutation-based compression function F_A of (3)

one permutation π , we are in the *single-permutation* setting. In total, we thus analyze $2 \cdot 2^{14}$ compression functions. Many of these, however, are trivially weak (cf. Sect. 2.3).

For the single-permutation setting, it is of interest to also consider the case where n -bit constants are added to the inputs to the permutations (e.g. $y_1 \leftarrow \pi_1(a_{11}x_1 \oplus a_{12}x_2 \oplus b_1)$ for $b_1 \in \{0, 1\}^n$). This results in many more schemes, but requires a more complex analysis. Therefore, we present our main results on F_A of (3), and in App. A we generalize our findings on the single-permutation setting to cover any F_A where additional affine transformations on the permutation inputs are taken into account.

2.2 Security Notions

An adversary is a probabilistic algorithm with oracle access to the underlying permutations π_1, π_2, π_3 . He can make forward and inverse queries to its oracles, and the queries are stored in a query history \mathcal{Q} . By $(x_k, y_k) \in \mathcal{Q}$, for $k \in \{1, 2, 3\}$, we denote that $y_k = \pi_k(x_k)$; the adversary either made a forward query x_k to obtain y_k or an inverse query y_k to obtain x_k . In the remainder, we assume that \mathcal{Q} always contains the queries required for the attack, and we assume that the adversary does not make trivial queries, i.e. queries to which the adversary already knows the answer in advance. In this work we consider information-theoretic adversaries only. This type of adversary has unbounded computational power, and its complexity is measured by the number of queries made to its oracles.

Definition 1. Let $F_A : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ be a compression function defined by a matrix A of the form (2). Let \mathcal{A} be a collision finding adversary for this compression function. The advantage of \mathcal{A} is defined as

$$\text{Adv}_{F_A}^{\text{col}}(\mathcal{A}) = \Pr \left(\pi_1, \pi_2, \pi_3 \xleftarrow{\$} P_n, x, x' \leftarrow \mathcal{A}^{\pi_i, \pi_i^{-1}} : x \neq x', F_A^{\pi_i}(x) = F_A^{\pi_i}(x') \right).$$

By $\text{Adv}_{F_A}^{\text{col}}(q)$ we denote the maximum advantage, taken over all adversaries making q queries to each of their oracles.

Several definitions for preimage resistance are known, but we opt for everywhere preimage resistance [9], which intuitively guarantees preimage security for every range point.

Definition 2. Let $F_A : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ be a compression function defined by a matrix A of the form (2). Let \mathcal{A} be an everywhere preimage finding adversary for this compression function. The advantage of \mathcal{A} is defined as

$$\text{Adv}_{F_A}^{\text{epre}}(\mathcal{A}) = \max_{z \in \{0, 1\}^n} \Pr \left(\pi_1, \pi_2, \pi_3 \xleftarrow{\$} P_n, x \leftarrow \mathcal{A}^{\pi_i, \pi_i^{-1}}(z) : z = F_A^{\pi_i}(x) \right).$$

By $\text{Adv}_{F_A}^{\text{epre}}(q)$ we denote the maximum advantage, taken over all adversaries making q queries to each of their oracles.

The security definitions for the single-permutation setting, where the compression function is built on one permutation π , are analogous.

2.3 Invalid Matrices

We will classify the set of optimally collision secure compression functions F_A of the form described in Sect. 2.1, but for some matrices A the induced compression function will clearly not fulfill the desired security requirements. For instance, if a compression function does not use one or more permutations, attacks faster than the birthday bound can easily be constructed. We introduce the notion of “valid” matrices, in order to rule out compression functions that trivially fail to achieve optimal collision resistance. A matrix A is called “valid” if it satisfies the following properties:

- (1) For the j^{th} column ($j = 1, 2$), we have $a_{1j} + a_{2j} + a_{3j} \geq 1$. This requirement ensures that input x_j is used in the computation of at least one permutation. If this would not be the case, collisions can easily be constructed;
- (2) For the j^{th} column ($j = 3, 4, 5$), we have $\|\mathbf{a}_{*,j}\| \geq 1$, and for the i^{th} row ($i = 1, 2, 3$), we have $\|\mathbf{a}_{i,*}\| \geq 1$. Notice that if the i^{th} row (resp. j^{th} column) would consist of zeroes only, it means that permutation π_i (resp. π_{j-2}) is not used in the computation, and collisions can be found in at most $2^{n/3}$ queries by Stam’s bound [13,14].

In the remainder, we will consider valid matrices A only. By an extensive computation one can show that $2796 < 2^{12}$ out of 2^{14} matrices are valid (for both the single- and multi-permutation setting).

3 Equivalence Classes of Permutation Based Compression Functions

We define an equivalence relation on the set of compression functions F_A . This equivalence relation intuitively describes classes of “equally secure” compression functions, and can be used to reduce the number of compression functions to be analyzed. Indeed, security properties of one compression function naturally convey to all compression functions in the same equivalence class. The equivalence relation is defined in Def. 3, and in Props. 1-4 we describe the four equivalence reductions that will be used in this work.

Definition 3. Two compression functions F_A and $F_{A'}$ are equivalent if for both collision and preimage security there exists a tight reduction from F_A to $F_{A'}$, and vice versa.

Proposition 1 (x-reduction). Consider two matrices $A = (\mathbf{a}_{*,1}; \mathbf{a}_{*,2}; \mathbf{a}_{*,3}; \mathbf{a}_{*,4}; \mathbf{a}_{*,5})$ and $A' = (\mathbf{a}_{*,2}; \mathbf{a}_{*,1}; \mathbf{a}_{*,3}; \mathbf{a}_{*,4}; \mathbf{a}_{*,5})$. Then, the compression functions F_A and $F_{A'}$ are equivalent. Intuitively, this reduction corresponds to swapping x_1 and x_2 .

Proposition 2 (XOR-reduction). Consider a matrix $A = (\mathbf{a}_{*,1}; \mathbf{a}_{*,2}; \mathbf{a}_{*,3}; \mathbf{a}_{*,4}; \mathbf{a}_{*,5})$, and let $k = \min\{ i \mid a_{i,2} \neq 0 \}$ (notice that $k \in \{1, 2, 3\}$ as A is valid). Let $c_0, \dots, c_2 \in \{0, 1\}$. Consider the matrix $A' = A \oplus (c_0 \mathbf{a}_{*,2}; \mathbf{0}; [k \geq$

$2]c_1\mathbf{a}_{*,2}; [k \geq 3]c_2\mathbf{a}_{*,2}; \mathbf{0})$, where $[X] = 1$ if X holds and 0 otherwise. Then, the compression functions F_A and $F_{A'}$ are equivalent. Intuitively, π_k is the first permutation that incorporates x_2 , and this reduction represents replacing x_2 by $x_2 \oplus c_0x_1 \oplus \sum_{i=1}^{k-1} c_iy_i$, where y_i is the outcome of the i^{th} permutation. Using Prop. 1, the same reduction holds for x_1 .

Proposition 3 (π -swap-reduction). Let $i \in \{1, 2\}$, and consider a matrix A with $a_{i+1,i+2} = 0$. Consider the matrix A' obtained from A by swapping rows $\mathbf{a}_{i,*}$ and $\mathbf{a}_{i+1,*}$ and consequently swapping columns $\mathbf{a}_{*,i+2}$ and $\mathbf{a}_{*,i+3}$. Then, the compression functions F_A and $F_{A'}$ are equivalent. Intuitively, this reduction corresponds to swapping π_i and π_{i+1} , which is only possible if the outcome of π_i is not used as input of π_{i+1} (i.e. if $a_{i+1,i+2} = 0$).

Proposition 4 (π -inverse-reduction). Consider a matrix A with $(a_{11}, a_{12}) = (1, 0)$. Consider the matrix A' obtained from A by swapping (a_{21}, a_{31}, a_{41}) and (a_{23}, a_{33}, a_{43}) . Then, the compression functions F_A and $F_{A'}$ are equivalent. Intuitively, this reduction corresponds to replacing π_1 by π_1^{-1} . Using Prop. 1 and Prop. 3 on $i = 1$, the same reduction holds for π_2 .

Proof (Proof of Props. 1-4). Let F_A and $F_{A'}$ be two compression functions defined as in either of the propositions. For simplicity, in case of Prop. 2 we only consider $k = 2$ (so $a_{12} = 0$, $a_{22} = 1$ and $c_2 = 0$), for Prop. 3 we only consider $i = 1$ (so $a_{23} = 0$). By construction, the compression functions F_A and $F_{A'}$ satisfy the following properties:

$$F_A^{\pi_1, \pi_2, \pi_3}(x_1, x_2) = \begin{cases} F_{A'}^{\pi_1, \pi_2, \pi_3}(x_2, x_1) & \text{for Prop. 1,} \\ F_{A'}^{\pi_1, \pi_2, \pi_3}(x_1, x_2 \oplus c_0x_1 \oplus c_1\pi_1(a_{11}x_1)) & \text{for Prop. 2,} \\ F_{A'}^{\pi_2, \pi_1, \pi_3}(x_1, x_2) & \text{for Prop. 3,} \\ F_{A'}^{\pi_1^{-1}, \pi_2, \pi_3}(\pi_1(x_1), x_2) & \text{for Prop. 4.} \end{cases} \quad (4)$$

We need to provide a bi-directional collision and preimage security reduction. For conciseness, we will provide only the collision security reduction; the case of preimage resistance is similar and is therefore omitted. Let \mathcal{A} be a collision finding adversary for the compression function F_A , that on input of $\pi_1, \pi_2, \pi_3 \xleftarrow{\$} P_n$, outputs two tuples $(x_1, x_2), (x'_1, x'_2)$ such that $F_A^{\pi_i}(x_1, x_2) = F_A^{\pi_i}(x'_1, x'_2)$. We construct a collision finding adversary \mathcal{A}' for $F_{A'}$ that uses \mathcal{A} as a subroutine and on input of $\pi'_1, \pi'_2, \pi'_3 \xleftarrow{\$} P_n$ outputs a collision for $F_{A'}^{\pi'_i}$. Adversary \mathcal{A}' operates as follows:

1. In Props. 1 and 2, the adversary \mathcal{A}' sends $(\pi_1, \pi_2, \pi_3) \leftarrow (\pi'_1, \pi'_2, \pi'_3)$ to \mathcal{A} . In Prop. 3, the adversary \mathcal{A}' sends $(\pi_1, \pi_2, \pi_3) \leftarrow (\pi'_2, \pi'_1, \pi'_3)$ to \mathcal{A} . In Prop. 4, the adversary \mathcal{A}' sends $(\pi_1, \pi_2, \pi_3) \leftarrow ((\pi'_1)^{-1}, \pi'_2, \pi'_3)$ to \mathcal{A} ;
2. \mathcal{A} outputs two tuples $(x_1, x_2), (x'_1, x'_2)$ such that $F_A^{\pi_i}(x_1, x_2) = F_A^{\pi_i}(x'_1, x'_2)$;
3. In Prop. 1, \mathcal{A}' outputs collision (x_2, x_1) and (x'_2, x'_1) . In Prop. 2, \mathcal{A}' outputs $(x_1, x_2 \oplus c_0x_1 \oplus c_1\pi_1(a_{11}x_1))$ and $(x'_1, x'_2 \oplus c_0x'_1 \oplus c_1\pi_1(a_{11}x'_1))$. In Prop. 3, \mathcal{A}' outputs (x_1, x_2) and (x'_1, x'_2) . In Prop. 4, \mathcal{A}' outputs $((\pi'_1)^{-1}(x_1), x_2)$ and $((\pi'_1)^{-1}(x'_1), x'_2)$.

Notice that in step one, (π_1, π_2, π_3) are clearly randomly and independently distributed as (π'_1, π'_2, π'_3) are, and therefore \mathcal{A} can output $(x_1, x_2), (x'_1, x'_2)$ such that $F_A^{\pi_1, \pi_2, \pi_3}(x_1, x_2) = F_A^{\pi'_1, \pi'_2, \pi'_3}(x'_1, x'_2)$ with probability $\text{Adv}_{F_A}^{\text{col}}(\mathcal{A})$. For $F_{A'}$ of Prop. 3, these tuples indeed render a collision as given in step 3:

$$\begin{aligned} F_{A'}^{\pi'_1, \pi'_2, \pi'_3}(x_1, x_2) &= F_A^{\pi'_2, \pi'_1, \pi'_3}(x_1, x_2) && \text{by (4),} \\ &= F_A^{\pi'_2, \pi'_1, \pi'_3}(x'_1, x'_2) && \text{by collision for } F_A, \\ &= F_{A'}^{\pi'_1, \pi'_2, \pi'_3}(x'_1, x'_2) && \text{by (4).} \end{aligned}$$

The same argument applies to the other propositions. In any case, \mathcal{A}' needs at most four queries more than \mathcal{A} , and thus we obtain $\text{Adv}_{F_A}^{\text{col}}(q) \leq \text{Adv}_{F_{A'}}^{\text{col}}(q+4)$. The reductions in the other direction (from $F_{A'}$ to F_A) are identical due to symmetry. \square

Except for Prop. 4, the reductions also hold in the single-permutation setting. We remark that these reductions are not only restricted to binary matrices, but apply to general matrices A . In particular, the independence criterion of [10] can be derived using the given reductions. Also, we note that the reductions can easily be represented by linear matrix operations.

4 Main Result for Multi-permutation Setting

We classify the set of permutation-based compression functions of the form (3) that achieve optimal collision resistance. Theorem 1 shows that the set of (asymptotically) secure functions is fully covered by four equivalence classes; for any other compression function collisions can be found faster than the birthday bound. One of these four classes – defined by F_{A_2} below – provides optimal (asymptotic) $2^{2n/3}$ preimage security, for the other three classes preimages can be found significantly faster.

Theorem 1. *Consider the multi-permutation setting. Let F_A be any compression function defined by a binary matrix A of the form (2). Let F_{A_k} for $k = 1, 2, 3, 4$ be the compression functions defined by matrices*

$$\begin{aligned} A_1 &= \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 1 \end{array} \right), & A_2 &= \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 1 & 1 \end{array} \right), \\ A_3 &= \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 & 1 \end{array} \right), & A_4 &= \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 & 1 \end{array} \right). \end{aligned} \tag{5}$$

Let $\varepsilon > 0$.

- (i) If F_A is equivalent to F_{A_k} for $k \in \{1, 2, 3, 4\}$, it satisfies $\lim_{n \rightarrow \infty} \text{Adv}_{F_A}^{\text{col}}(2^{n/2(1-\varepsilon)}) = 0$. Otherwise, it satisfies $\text{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^5/2^{2n})$;

- (ii) If F_A is equivalent to F_{A_2} , it satisfies $\lim_{n \rightarrow \infty} \mathbf{Adv}_{F_A}^{\text{epre}}(2^{2n/3(1-\varepsilon)}) = 0$;
- (iii) If F_A is equivalent to F_{A_k} for $k \in \{1, 3, 4\}$, it satisfies $\mathbf{Adv}_{F_A}^{\text{epre}}(q) = \Theta(q^2/2^n)$.

In other words, a compression function offers optimal collision resistance *if and only if* it is equivalent to either of $F_{A_1}, F_{A_2}, F_{A_3}, F_{A_4}$, and additionally achieves optimal preimage resistance (with respect to the bounds of [11]) *if and only if* it is equivalent to F_{A_2} .

In order to prove Thm. 1, more specifically part (i) for $k = 1, 2, 4$ and part (ii), we pose the following conjecture. This conjecture relates to the area of extremal graph theory and is of independent interest. In particular, it can be shown to be similar to (but more complex than) a longstanding problem of Zarankiewicz from 1951 [2, Ch. 6.2].

Conjecture 1. Let $q \leq 2^n$, and let Z be a set of q elements taken uniformly at random from $\{0, 1\}^n$. Let β denote the maximum number of tuples $(x_1, x_2, z) \in X_1 \times X_2 \times Z$ such that $x_1 \oplus x_2 = z$, where X_1, X_2 are any two subsets of $\{0, 1\}^n$ of size q . Formally:

$$\beta := \max_{\substack{X_1, X_2 \subseteq \{0, 1\}^n \\ |X_1| = |X_2| = q}} |\{(x_1, x_2, z) \in X_1 \times X_2 \times Z \mid x_1 \oplus x_2 = z\}|. \quad (6)$$

There exists a constant d_1 such that $\mathbf{Pr}(\beta > d_1 q \log q) \rightarrow 0$ for $n \rightarrow \infty$ and $q < 2^{n/2}$. Similarly, there exists a constant d_2 such that $\mathbf{Pr}(\beta > d_2 q^{3/2}) \rightarrow 0$ for $n \rightarrow \infty$ and $q < 2^{2n/3}$.

The first bound is used in the proof Thm. 1(i) for $k = 1, 2, 4$, and the second bound in the proof Thm. 1(ii). A detailed heuristic for Conj. 1 is given in [6], together with a comparison with Zarankiewicz's conjecture, but we leave a full proof of Conj. 1 as an open problem.

4.1 Proof of Theorem 1

The proof of Thm. 1 is structured as follows. Firstly, in Lem. 1 we show that any compression function F_A can be reduced either to an invalid compression function or to a compression function $F_{A'}$ defined by a matrix A' with first two rows 10000, 01000. By construction (see Sect. 3), the security properties of one compression function are valid for the whole equivalence class. Secondly, in Lem. 2 several collision attacks are described that invalidate the security of each of the remaining compression functions, except for the classes defined by F_{A_k} ($k \in \{1, 2, 3, 4\}$) for A_k as in (5). Thirdly, the collision and preimage resistance of the remaining four compression functions are analyzed in Lem. 3, which completes the proof of Thm. 1.

Lemma 1. *Any compression function F_A , for valid A , is equivalent to a compression function $F_{A'}$, where either A' is invalid or the first two rows of A' equal 10000, 01000.*

Proof. The proof is constructive. Several reductions are used, but for ease of notation apostrophes are omitted. Let F_A be a compression function defined by some valid matrix A . As A is valid, we have $a_{11} + a_{12} \geq 1$. If $a_{11} + a_{12} = 2$, we can apply Prop. 2 on $c_0 = 1$ to obtain $a_{11} + a_{12} = 1$. Now, by Prop. 1 we can assume that $(a_{11}, a_{12}) = (1, 0)$.

Considering the second row of A , we distinguish between $a_{22} = 1$ and $a_{22} = 0$. In the former case, a XOR-reduction (Prop. 2) on $(c_0, c_1) = (a_{21}, a_{23})$ reduces the scheme to the required form. In the latter case, where $a_{22} = 0$, we proceed as follows. If $a_{32} = 0$, A is equivalent to an invalid matrix. Otherwise, by applying Prop. 2 with $(c_0, c_1, c_2) = (a_{31}, a_{33}, a_{34})$ we obtain that F_A is equivalent to a compression function $F_{A'}$, for some matrix A' with rows $(10000, a'_{21}0a'_{23}00, 01000, a'_{41}a'_{42}a'_{43}a'_{44}a'_{45})$. The result is now obtained by swapping π_2 and π_3 (Prop. 3 for $i = 2$). \square

As a direct consequence of Lem. 1, it suffices to consider compression functions F_A , where

$$A = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & 0 \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & 1 \end{array} \right) \quad (7)$$

for some binary values a_{31}, \dots, a_{44} . Notice that $a_{45} = 1$ because of the validity of the matrix. We describe a couple of collision attacks that apply to compression functions of this form. We note that similar results also hold for preimage resistance.

Lemma 2. *Let F_A be a compression function defined by a valid matrix A of the form (7).*

- (i) *If A satisfies $(a_{31} + a_{33})(a_{32} + a_{34}) = 0$, then $\text{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^4/2^n)$;*
- (ii) *If A satisfies $\bigvee_{j=1}^4 a_{3j} = a_{4j} = 0$, then $\text{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^3/2^n)$;*
- (iii) *If A satisfies $\bigwedge_{j=1}^2 a_{3j}a_{4,j+2} \neq a_{3,j+2}a_{4j}$, then $\text{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^3/2^n)$;*
- (iv) *If A satisfies $a_{41} + a_{42} + a_{43} + a_{44} = 1$, then $\text{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^5/2^{2n})$.*

For clarity, the proofs of results (i), (ii), (iii) and (iv) will be given separately.

Proof (Proof of Lem. 2(i)). Without loss of generality, we assume $a_{32} + a_{34} = 0$, i.e. $a_{32} = a_{34} = 0$. Hence, we consider matrices A with $\left(\begin{smallmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{smallmatrix} \right) = \left(\begin{smallmatrix} a_{31} & 0 & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{smallmatrix} \right)$, where $a_{31} + a_{33} \geq 1$, by validity of A . This matrix defines the compression function:

$$F_A(x_1, x_2) = a_{41}x_1 \oplus a_{42}x_2 \oplus a_{43}\pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3(a_{31}x_1 \oplus a_{33}\pi_1(x_1)).$$

Define the functions $f_1(x) = a_{41}x \oplus a_{43}\pi_1(x) \oplus \pi_3(a_{31}x \oplus a_{33}\pi_1(x))$ and $f_2(x) = a_{42}x \oplus \pi_2(x)$. Notice that $F_A(x_1, x_2) = f_1(x_1) \oplus f_2(x_2)$. A collision-finding adversary \mathcal{A} for F_A proceeds as follows. He sets up two lists of q random elements $X_1 := \{x_1^{(1)}, \dots, x_1^{(q)}\}$ and $X_2 := \{x_2^{(1)}, \dots, x_2^{(q)}\}$, and computes the corresponding values $f_1(x_1^{(k)})$ and $f_2(x_2^{(k)})$ (for $k = 1, \dots, q$). Thus, in total \mathcal{A} makes

q queries to each of his random oracles. Given one of the $\binom{q}{2}^2$ combinations $x_1, x'_1 \in X_1$, $x_2, x'_2 \in X_2$, this combination yields a collision for F_A with probability $\Theta(2^{-n})$. Concluding, $\mathbf{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^4/2^n)$. \square

Proof (Proof of Lem. 2(ii)). For the cases $j \in \{3, 4\}$ as explained in Sect. 2.3 (these cases are in fact redundant due to the validity of A), collisions can be found in at most $2^{n/3}$ queries due to Stam's bound [13,14]. We consider a matrix A with $a_{32} = a_{42} = 0$ (the case $j = 2$), a similar analysis holds for $j = 1$. Note that F_A satisfies $F_A(x_1, x_2) = F_{A'}(x_1, \pi_2(x_2))$, where A' has third and fourth rows $(a_{31}a_{34}a_{33}00, a_{41}a_{44}a_{43}01)$. The compression function $F_{A'}$ satisfies the condition of this lemma for $j = 4$, and invertibility of π_2 guarantees a collision for F_A in the same amount of queries plus 2. We note that the result also follows from Prop. 4, but as we will use Lem. 2(ii) in the single-permutation setting as well, we here consider a more robust reduction. \square

Proof (Proof of Lem. 2(iii)). The idea of the attack is to focus on collisions $(x_1, x_2) \neq (x'_1, x'_2)$ for which the input to the third permutation π_3 is the same. We first consider the case of matrices A with $\begin{pmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ a_{41} & a_{42} & 1 & 1 \end{pmatrix}$, the general case is discussed afterwards. The matrix defines compression function

$$F_A(x_1, x_2) = a_{41}x_1 \oplus a_{42}x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3(x_1 \oplus x_2).$$

We construct an adversary \mathcal{A} that aims at finding a collision $(x_1, x_2) \neq (x'_1, x'_2)$ such that

$$x_1 \oplus x_2 = x'_1 \oplus x'_2, \quad (8a)$$

$$a_{41}x_1 \oplus a_{42}x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2) = a_{41}x'_1 \oplus a_{42}x'_2 \oplus \pi_1(x'_1) \oplus \pi_2(x'_2). \quad (8b)$$

The adversary sets up two lists of $q = 2^\alpha$ elements $X_1 := \{x_1^{(1)}, \dots, x_1^{(q)}\}$ and $X_2 := \{x_2^{(1)}, \dots, x_2^{(q)}\}$, where $x_1^{(k)} = x_2^{(k)} = 0^{n-\alpha} \parallel \langle k-1 \rangle_\alpha$ for $k = 1, \dots, q$. He computes the corresponding values $\pi_1(x_1^{(k)})$ and $\pi_2(x_2^{(k)})$ (for $k = 1, \dots, q$). Fix any x_1, x_2, x'_1 such that $x_1 \neq x'_1$. Then, there is exactly one x'_2 such that (8a) is satisfied. For any of these $q \binom{q}{2}$ options, (8b) is satisfied with probability $\Theta(2^{-n})$. For any of such succeeding tuples, the adversary additionally queries $\pi_3(x_1 \oplus x_2) = \pi_3(x'_1 \oplus x'_2)$ in order to get a collision. Concluding, $\mathbf{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^3/2^n)$.

The described attack relies on the key property that the set of equations

$$\begin{pmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} (x_1 \oplus x'_1, x_2 \oplus x'_2, \pi_1(x_1) \oplus \pi_1(x'_1), \pi_2(x_2) \oplus \pi_2(x'_2))^\top = 0$$

contains an equation in which x_1, x_2, x'_1, x'_2 occur exactly once. By the requirement of A, $\begin{pmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$ contains at least two zeroes. If two zeroes are located in the same row, this key property is satisfied and the attack succeeds. On the other hand, if both rows contain exactly one zero, one can XOR the first equation to the second one to return to the first case. \square

Proof (Proof of Lem. 2(iv)). Without loss of generality, we assume $a_{41} = 1$. By Lem. 2(ii), we can consider $a_{32} = a_{33} = a_{34} = 1$. The matrix defines compression function

$$\mathsf{F}_A(x_1, x_2) = x_1 \oplus \pi_3(a_{31}x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2)).$$

We construct a collision adversary \mathcal{A} for F_A . The adversary sets up a list of $q = 2^\alpha$ random elements $X_2 := \{x_2^{(1)}, \dots, x_2^{(q)}\}$, and computes the corresponding values $y_2^{(k)} = \pi_2(x_2^{(k)})$ (for $k = 1, \dots, q$). Additionally, the adversary sets up two lists $X_1 := \{x_1^{(1)}, \dots, x_1^{(q)}\}$ and $Y_3 := \{y_3^{(1)}, \dots, y_3^{(q)}\}$, where $x_1^{(k)} = y_3^{(k)} = 0^{n-\alpha} \parallel \langle k-1 \rangle_\alpha$ for $k = 1, \dots, q$. He computes the corresponding values $y_1^{(k)} = \pi_1(x_1^{(k)})$ and $x_3^{(k)} = \pi_3^{-1}(y_3^{(k)})$ (for $k = 1, \dots, q$). Fix any x_1, y_3, x'_1 such that $x_1 \neq x'_1$. Then, there is exactly one y'_3 such that $x_1 \oplus y_3 = x'_1 \oplus y'_3$. The adversary obtains a collision for F_A if X_2 contains two elements x_2, x'_2 such that $x_2 \oplus y_2 = a_{31}x_1 \oplus y_1 \oplus x_3$ and $x'_2 \oplus y'_2 = a_{31}x'_1 \oplus y'_1 \oplus x'_3$. Two such x_2, x'_2 exist with probability $\Omega(\binom{q}{2}/2^{2n})$. As the adversary needs to succeed for only one of the $q \binom{q}{2}$ choices of x_1, y_3, x'_1 , he finds a collision for F_A with probability $\Omega(q^5/2^{2n})$. \square

Next, the compression functions evolved from Lem. 1 are analyzed with respect to the attacks of Lem. 2. Before proceeding, we remark that for the multi-permutation setting, the following reductions apply to the compression function classes evolved from Lem. 1. We refer to these reductions as the “M- and N-reduction”.

M-reduction: Applying Prop. 1, and Prop. 3 on $i = 1$ corresponds to mutually swapping $\binom{a_{31}}{a_{41}} \leftrightarrow \binom{a_{32}}{a_{42}}$ and $\binom{a_{33}}{a_{43}} \leftrightarrow \binom{a_{34}}{a_{44}}$;

N-reduction: Prop. 4 reduces to swapping $\binom{a_{3j}}{a_{4j}} \leftrightarrow \binom{a_{3,j+2}}{a_{4,j+2}}$ for $j \in \{1, 2\}$.

We now continue evaluating the matrices A of the form (7), and consider the different values of $\|\mathbf{a}_{3,*}\|$.

$\|\mathbf{a}_{3,*}\| = 0$. The matrix is invalid and excluded by definition;

$\|\mathbf{a}_{3,*}\| = 1$. The matrix is vulnerable to the attack of Lem. 2(i);

$\|\mathbf{a}_{3,*}\| = 2$. The matrix contradicts either one of the requirements of Lem. 2. Technically, if $(a_{31} + a_{33})(a_{32} + a_{34}) = 0$ it violates Lem. 2(i), and otherwise the values a_{41}, \dots, a_{44} will violate either the requirement of Lem. 2(ii) or of Lem. 2(iii);

$\|\mathbf{a}_{3,*}\| = 3$. Due to M- and N-reductions, it suffices to consider $a_{31}a_{32}a_{33}a_{34} = 1110$, and consequently $a_{44} = 1$ by Lem. 2(ii). Lemma 2(iii) now states that we require $a_{41} = a_{43}$, which gives the following four options for $a_{41}a_{42}a_{43}$: 000, 010, 101 and 111. The first one is vulnerable to the attack of Lem. 2(iv), and the fourth matrix is equivalent to the second (by consequently applying Prop. 2 on $(c_0, c_1) = (1, 1)$, and Prop. 3 for $i = 2$). We are left with A_1 and A_2 of (5);

$\|\mathbf{a}_{3,*}\| = 4$. Due to M- and N-reductions, it suffices to consider $a_{41}a_{42}a_{43}a_{44} \in \{0000, 1000, 1010, 1100, 1110, 1111\}$. The cases 1000 and 1100 are vulnerable to the attacks of Lems. 2(iv) and 2(iii), respectively. For the cases 0000 and 1111, finding collisions is as hard as finding collisions for $F(x_1, x_2) = x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2)$ (for which collisions are found in at most $2^{n/3}$ queries, due to Stam's bound [13,14]). We are left with A_3 and A_4 of (5).

It remains to analyze collision and preimage security of the four compression functions defined by the matrices of (5). This is covered by following lemma, which is proven in [6]. Particularly, Lem. 3 completes the proof of Thm. 1.

Lemma 3. *Let $\varepsilon > 0$. Then:*

- (i) $\lim_{n \rightarrow \infty} \mathbf{Adv}_{F_{A_k}}^{\text{col}}(2^{n/2(1-\varepsilon)}) = 0$ for $k = 1, 2, 3, 4$;
- (ii) $\lim_{n \rightarrow \infty} \mathbf{Adv}_{F_{A_2}}^{\text{epre}}(2^{2n/3(1-\varepsilon)}) = 0$, and $\mathbf{Adv}_{F_{A_k}}^{\text{epre}}(q) = \Theta(q^2/2^n)$ for $k = 1, 3, 4$.

5 Main Result for Single-Permutation Setting

In a similar fashion as in Sect. 4, we analyze the security of compression functions based on three calls to the same permutations, the single-permutation setting. It turns out that *there does not exist any* compression function of the form (3) that achieves optimal collision resistance. We note that this result does not rely on Conj. 1. In App. A we show how the results of this section can be generalized to cover any single-permutation compression function where additional affine transformations on the permutation inputs are taken into account.

Theorem 2. *Consider the single-permutation setting, where $\pi_1 = \pi_2 = \pi_3 =: \pi$. Any compression function F_A defined by a binary matrix A of the form (2) satisfies $\mathbf{Adv}_{F_A}^{\text{col}}(q) = \Omega(q^5/2^{2n})$.*

Proof. The proof of Thm. 2 is similar to the proof of Thm. 1, and we highlight the differences. Lemmas 1 and 2 still apply, and additionally the M-reduction also holds in the single-permutation setting. Notice that the N-reduction *does not hold* as it incorporates Prop. 4. Similar to before, we will evaluate the matrices A of the form (7). The case $\|\mathbf{a}_{3,*}\| \leq 2$ is the same as before.

$\|\mathbf{a}_{3,*}\| = 3$. Due to M-reductions, it suffices to consider $a_{31}a_{32}a_{33}a_{34} \in \{1110, 0111\}$.

- $a_{31}a_{32}a_{33}a_{34} = 1110$. The same analysis as in Sect. 4.1 applies, leaving the matrices A_1 and A_2 of (5). In the single-permutation setting, the two corresponding compression functions satisfy $F_{A_1}(x_1, \pi(x_1)) = \pi^2(x_1)$ and $F_{A_2}(x_1, x_2) = F_{A_2}(x_1, x_1 \oplus x_2 \oplus \pi(x_1))$ for any x_1, x_2 . Collisions can thus be trivially found;
- $a_{31}a_{32}a_{33}a_{34} = 0111$. By Lem. 2(ii), we have $a_{41} = 1$. Lemma 2(iii) now states that we require $a_{42} = a_{44}$, which gives the following four options for $a_{42}a_{43}a_{44}$: 000, 010, 101 and 111. The first one is vulnerable

to the attack of Lem. 2(iv), the second, third and fourth matrix satisfy $F_A(x_1, x_1) = x_1$, $F_A(x_1, x_1) = 0$ and $F_A(x_1, x_1) = \pi(x_1)$, respectively, for any x_1 . Collisions can thus be trivially found;

$\|\mathbf{a}_{3,*}\| = 4$. Except for $a_{41}a_{42}a_{43}a_{44} \in \{1010, 1001, 0110, 0101\}$, all induced compression functions satisfy $F_A(x_1, x_1) \oplus \pi(0) \in \{0, x_1, \pi(x_1)\}$ for any x_1 , for which collisions can be trivially found. The cases 1001, 0110 are vulnerable to Lem. 2(iii). The remaining two cases, which are equivalent by M-reduction, allow for trivial collisions as well: the compression function induced by $(a_{41}a_{42}a_{43}a_{44}) = (1010)$ satisfies $F_A(x_1, \pi^{-1}(x_1 \oplus \pi(x_1))) = 0$ for any x_1 (cf. [10]).

Hence, the analyzed compression functions either allow for trivial collision or are vulnerable to Lem. 2, therewith allowing for collisions in at most $2^{2n/5}$ queries. \square

Concluding, for any compression function F_A of the form (3), where the three permutations are equal to one single permutation π , collisions can be found in at most $2^{2n/5}$ queries, hence considerably faster than in $2^{n/2}$ queries.

6 Conclusions

We provided a full security classification of $2n$ -to- n -bit compression functions that are solely built of XOR-operators and of three permutations. Therewith, we have analyzed compression functions that are not included in the analysis of Rogaway and Steinberger [10], but yet are interesting because of their elegance (they only employ XOR-operators) and efficiency (XOR-operators are slightly cheaper than finite field multiplications by constants). For any of the 2^{15} compression functions of the described form, we either provide a formal collision and preimage security proof or a collision attack more efficient than the birthday bound.

For the multi-permutation setting, where the three permutations are different, there are exactly four equivalence classes of functions that allow for optimal collision resistance, one class of which the compression functions achieve optimal preimage resistance w.r.t. the bounds of [11]. A summary of these results is given in Table 1. Regarding the absolute number of collision/preimage secure compression functions, by ways of an extensive computation one finds 96 functions equivalent to F_{A_1} (including the F_{A_1} itself), 48 functions in each of the classes defined by F_{A_2} and F_{A_4} , and 24 functions equivalent to F_{A_3} . In total, we have thus proven 216 compression functions optimally collision secure, 48 of which we have proven optimally preimage secure. A small part of the results for the multi-permutation setting relies on an extremal graph theory based conjecture, Conj. 1, which we supported by an extensive and detailed heuristic. We leave the full analysis of Conj. 1 as an open problem.

For the single-permutation setting, where the three permutations are the same, we show that it is not possible to construct a $2n$ -to- n -bit compression

function that achieves optimal collision resistance. In light of the amount of optimally secure compression functions we have found in the multi-permutation setting, this observation is not as expected. This negative result casts doubts over the existence of any (larger) permutation-based XOR-based compression function built on (multiple invocations of) one single permutation. We leave this question as an open problem.

The results in this work are derived in the permutation setting. Different results may be obtained if we consider three underlying primitives to be one-way functions: in particular, the π -inverse-reduction (Prop. 4) and Lem. 2 rely on the invertibility of these primitives. Further research questions include the applicability of the approach followed in this work to different classes of compression functions, for instance with larger domain and range, with more permutations or random functions instead, or defined over different fields.

Acknowledgments. This work has been funded in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, and in part by the Research Council K.U.Leuven: GOA TENSE. The first author is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

1. Black, J., Cochran, M., Shrimpton, T.: On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 526–541. Springer, Heidelberg (2005)
2. Bollobás, B.: *Extremal Graph Theory*. Academic Press (1978)
3. Hirose, S.: Some Plausible Constructions of Double-Block-Length Hash Functions. In: Robshaw, M. (ed.) *FSE 2006*. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
4. Lai, X., Massey, J.L.: Hash Functions Based on Block Ciphers. In: Rueppel, R.A. (ed.) *EUROCRYPT 1992*. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
5. Lee, J., Kwon, D.: Security of single-permutation-based compression functions. *Cryptology ePrint Archive*, Report 2009/145 (2009)
6. Mennink, B., Preneel, B.: Hash functions based on three permutations: A generic security analysis. *Cryptology ePrint Archive*, Report 2011/532 (2011), full version of this paper
7. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) *CRYPTO 1993*. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
8. Rabin, M.: Digitalized signatures. In: *Foundations of Secure Computation 1978*, pp. 155–166. Academic Press, New York (1978)
9. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) *FSE 2004*. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)

10. Rogaway, P., Steinberger, J.: Constructing Cryptographic Hash Functions from Fixed-Key Blockciphers. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 433–450. Springer, Heidelberg (2008)
11. Rogaway, P., Steinberger, J.: Security/Efficiency Tradeoffs for Permutation-Based Hashing. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008)
12. Shrimpton, T., Stam, M.: Building a Collision-Resistant Compression Function from Non-compressing Primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórssón, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008)
13. Stam, M.: Beyond Uniformity: Better Security/Efficiency Tradeoffs for Compression Functions. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 397–412. Springer, Heidelberg (2008)
14. Steinberger, J.: Stam’s Collision Resistance Conjecture. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 597–615. Springer, Heidelberg (2010)

A Generalization of Theorem 2

We generalize our findings on the single-permutation setting to cover *any* function, where affine transformations on the inputs to the permutations are taken into account. This generalization is straightforward, but technical and more elaborate. For a matrix $B = (b_1, b_2, b_3, b_4)^\top$ with elements in $\{0, 1\}^n$, we define the compression function F_{AB} as follows:

$$\begin{aligned} F_{AB}(x_1, x_2) = z, \text{ where } & y_1 \leftarrow \pi_1(a_{11}x_1 \oplus a_{12}x_2 \oplus b_1), \\ & y_2 \leftarrow \pi_2(a_{21}x_1 \oplus a_{22}x_2 \oplus a_{23}y_1 \oplus b_2), \\ & y_3 \leftarrow \pi_3(a_{31}x_1 \oplus a_{32}x_2 \oplus a_{33}y_1 \oplus a_{34}y_2 \oplus b_3), \\ & z \leftarrow a_{41}x_1 \oplus a_{42}x_2 \oplus a_{43}y_1 \oplus a_{44}y_2 \oplus a_{45}y_3 \oplus b_4. \end{aligned} \tag{9}$$

where A is as in Sect. 2.1. We note that for the multi-permutation setting, this generalization is of no added value, as the permutations are independently distributed anyway. Adding constants is, however, a customary approach to obtain “different” permutations from a single one (e.g. $\pi_i(x) = \pi(b_i \oplus x)$ for $i = 1, 2, 3$), but as we will show, the findings of Thm. 2 also apply to this extended setting.

We reformulate Props. 1-3 to the case of F_{AB} (recall that Prop. 4 did not apply to the single-permutation setting in the first place). Propositions 1 and 2 apply to any F_{AB} and $F_{A'B'}$ with $B = B'$ and Prop. 3 holds for any B and B' with $(b'_i, b'_{i+1}) = (b_{i+1}, b_i)$. Given this, the proof of Thm. 2 almost carries over. Lemmas 1 and 2 apply with straightforward generalization. It remains to evaluate the matrices A of the form (7) for any $B \in (\{0, 1\}^n)^{4 \times 1}$. The case $\|\mathbf{a}_{3,*}\| \leq 2$ is the same as in the proof of Thm. 2.

$\|\mathbf{a}_{3,*}\| = 3$. Due to M-reductions, it suffices to consider $a_{31}a_{32}a_{33}a_{34} \in \{1110, 0111\}$.

– $a_{31}a_{32}a_{33}a_{34} = 1110$. The same analysis as in Sect. 4.1 applies, leaving the matrices A_1 and A_2 of (5). In the extended single-permutation setting, the two corresponding compression functions satisfy $F_{A_1B}(x_1 \oplus$

$b_1, \pi(x_1) \oplus b_1 \oplus b_3 = \pi(\pi(x_1) \oplus b_1 \oplus b_2 \oplus b_3) \oplus b_1 \oplus b_3 \oplus b_4$ and $F_{A_2B}(x_1, x_2) = F_{A_2B}(x_1, x_1 \oplus x_2 \oplus \pi(x_1 \oplus b_1) \oplus b_2 \oplus b_3)$ for any x_1, x_2 . Collisions can thus be trivially found;

- $a_{31}a_{32}a_{33}a_{34} = 0111$. By Lem. 2(ii), we have $a_{41} = 1$. Lemma 2(iii) now states that we require $a_{42} = a_{44}$, which gives the following four options for $a_{42}a_{43}a_{44}$: 000, 010, 101 and 111. The first one is vulnerable to the attack of Lem. 2(iv), the second, third and fourth matrix satisfy $F_{AB}(x_1, \pi^{-1}(\pi(x_1 \oplus b_1) \oplus b_2 \oplus b_3) \oplus b_2) = x_1 \oplus b_2 \oplus b_3 \oplus b_4$, $F_{AB}(x_1 \oplus b_1, x_1 \oplus b_2) = F_{AB}(x_1 \oplus b_1 \oplus b_2 \oplus b_3, x_1 \oplus b_3)$ and $F_{AB}(x_1 \oplus b_1, x_1 \oplus b_2) = \pi(x_1 \oplus b_2 \oplus b_3) \oplus b_1 \oplus b_2 \oplus b_4$, respectively, for any x_1 . Collisions can thus be trivially found;

$\|\mathbf{a}_{3,*}\| = 4$. Except for $a_{41}a_{42}a_{43}a_{44} \in \{1010, 1001, 0110, 0101\}$, all induced compression functions satisfy $F_{AB}(x_1 \oplus b_1, x_1 \oplus b_2) \oplus \pi(b_1 \oplus b_2 \oplus b_3) \oplus a_{41}b_1 \oplus a_{42}b_2 \oplus b_4 \in \{0, x_1, \pi(x_1)\}$ for any x_1 , for which collisions can be trivially found. The cases 1001, 0110 are vulnerable to Lem. 2(iii). The remaining two cases, which are equivalent by M-reduction, allow for trivial collisions as well: the compression function induced by $(a_{41}a_{42}a_{43}a_{44}) = (1010)$ satisfies $F_{AB}(x_1, \pi^{-1}(x_1 \oplus \pi(x_1 \oplus b_1) \oplus b_2 \oplus b_3) \oplus b_2) = b_2 \oplus b_3 \oplus b_4$ for any x_1 .

Hence, any of the analyzed compression functions either allows for trivial collision or is vulnerable to Lem. 2, therewith allowing for collisions in at most $2^{2n/5}$ queries.

Concluding, for any compression function F_{AB} of the generalized form (9), collisions can be found in at most $2^{2n/5}$ queries, hence considerably faster than in $2^{n/2}$ queries.

To Hash or Not to Hash Again? (In)Differentiability Results for H^2 and HMAC

Yevgeniy Dodis¹, Thomas Ristenpart², John Steinberger³,
and Stefano Tessaro⁴

¹ New York University
`dodis@cs.nyu.edu`

² University of Wisconsin–Madison
`rirst@cs.wisc.edu`

³ Tsinghua University
`jpsteinb@gmail.com`

⁴ Massachusetts Institute of Technology
`tessaro@csail.mit.edu`

Abstract. We show that the second iterate $H^2(M) = H(H(M))$ of a random oracle H cannot achieve strong security in the sense of indifferentiability from a random oracle. We do so by proving that indifferentiability for H^2 holds only with poor concrete security by providing a lower bound (via an attack) and a matching upper bound (via a proof requiring new techniques) on the complexity of any successful simulator. We then investigate HMAC when it is used as a general-purpose hash function with arbitrary keys (and not as a MAC or PRF with uniform, secret keys). We uncover that HMAC’s handling of keys gives rise to two types of weak key pairs. The first allows trivial attacks against its indifferentiability; the second gives rise to structural issues similar to that which ruled out strong indifferentiability bounds in the case of H^2 . However, such weak key pairs do not arise, as far as we know, in any deployed applications of HMAC. For example, using keys of any fixed length shorter than $d - 1$, where d is the block length in bits of the underlying hash function, completely avoids weak key pairs. We therefore conclude with a positive result: a proof that HMAC is indifferentiable from a RO (with standard, good bounds) when applications use keys of a fixed length less than $d - 1$.

Keywords: Indifferentiability, Hash functions, HMAC.

1 Introduction

Cryptographic hash functions such as those in the MD and SHA families are constructed by extending the domain of a fixed-input-length compression function via the Merkle-Damgård (MD) transform. This applies some padding to a message and then iterates the compression function over the resulting string to compute a digest value. Unfortunately, hash functions built this way are vulnerable to extension attacks that abuse the iterative structure underlying MD [22, 34]:

given the hash of a message $H(M)$ an attacker can compute $H(M \parallel X)$ for some arbitrary X , even without knowing M .

In response, suggestions for shoring up the security of MD-based hash functions were made. The simplest is due to Ferguson and Schneier [20], who advocate a hash-of-hash construction: $H^2(M) = H(H(M))$, the second iterate of H . An earlier example is HMAC [5], which similarly applies a hash function H twice, and can be interpreted as giving a hash function with an additional key input. Both constructions enjoy many desirable features: they use H as a black box, do not add large overheads, and appear to prevent the types of extension attacks that plague MD-based hash functions.

Still, the question remains whether they resist other attacks. More generally, we would like that H^2 and HMAC behave like random oracles (ROs). In this paper, we provide the first analysis of these functions as being indifferentiable from ROs in the sense of [13, 29], which (if true) would provably rule out most structure-abusing attacks. Our main results surface a seemingly paradoxical fact, that the hash-of-hash H^2 cannot be indifferentiable from a RO with good bounds, even if H is itself modeled as a keyed RO. We then explore the fall out, which also affects HMAC.

INDIFFERENTIABILITY. Coron et al. [13] suggest that hash functions be designed so that they “behave like” a RO. To define this, they use the indifferentiability framework of Maurer et al. [29]. Roughly, this captures that no adversary can distinguish between a pair of oracles consisting of the construction (e.g., H^2) and its underlying ideal primitive (an ideal hash H) and the pair of oracles consisting of a RO and a simulator (which is given access to the RO). A formal definition is given in Section 2. Indifferentiability is an attractive goal because of the MRH composition theorem [29]: if a scheme is secure when using a RO it is also secure when the RO is replaced by a hash construction that is indifferentiable from a RO. The MRH theorem is widely applicable (but not ubiquitously, c.f., [31]), and so showing indifferentiability provides broad security guarantees.

While there exists a large body of work showing various hash constructions to be indifferentiable from a RO (c.f., [1, 7, 11–13, 15, 16, 23]), none have yet analyzed either H^2 or HMAC. Closest is the confusingly named HMAC construction from [13], which hashes a message by computing $H^2(0^d \parallel M)$ where H is MD using a compression function with block size d bits. This is not the same as HMAC proper nor H^2 , but seems close enough to both that one would expect that the proofs of security given in [13] apply to all three.

1.1 The Second Iterate Paradox

Towards refuting the above intuition, consider that $H^2(H(M)) = H(H^2(M))$. This implies that an output of the construction $H^2(M)$ can be used as an intermediate value to compute the hash of the message $H(M)$. This property does not exist in typical indifferentiable hash constructions, which purposefully ensure that construction outputs are unlikely to coincide with intermediate values. However, and unlike where extension attacks apply (they, too, take advantage

of outputs being intermediate values), there are no obvious ways to distinguish H^2 from a RO.

Our first technical contribution, then, is detailing how this structural property might give rise to vulnerabilities. Consider computing a hash chain of length ℓ using H^2 as the hash function. That is, compute $Y = H^{2\ell}(M)$. Doing so requires 2ℓ H -applications. But the structural property of H^2 identified above means that, given M and Y one can compute $H^{2\ell}(H(M))$ using only one H -application: $H(Y) = H(H^{2\ell}(M)) = H^{2\ell}(H(M))$. Moreover, the values computed along the first hash chain, namely the values $Y_i \leftarrow H^{2i}(M)$ and $Y'_i \leftarrow H^{2i}(H(M))$ for $0 \leq i \leq \ell$ are disjoint with overwhelming probability (when ℓ is not unreasonably large). Note that for chains of RO applications, attempting to cheaply compute such a second chain would not lead to disjoint chains. This demonstrates a way in which a RO and H^2 differ.

We exhibit a cryptographic setting, called *mutual proofs of work*, in which the highlighted structure of H^2 can be exploited. In mutual proofs of work, two parties prove to each other that they have computed some asserted amount of computational effort. This task is inspired by, and similar to, client puzzles [18, 19, 24, 25, 33] and puzzle auctions [35]. We give a protocol for mutual proofs of work whose computational task is computing hash chains. This protocol is secure when using a random oracle, but when using instead H^2 an attacker can cheat by abusing the structural properties discussed above.

INDIFFERENTIABILITY LOWER BOUND. The mutual proofs of work example already points to the surprising fact that H^2 does not “behave like” a RO. In fact, it does more, ruling out proofs of indifferentiability for H^2 with good bounds. (The existence of a tight proof of indifferentiability combined with the composition theorem of [29] would imply security for mutual proofs of work, yielding a contradiction.) However, we find that the example does not surface well why simulators must fail, and the subtlety of the issues here prompt further investigation. We therefore provide a direct negative result in the form of an indifferentiability distinguisher. We prove that should the distinguisher make q_1, q_2 queries to its two oracles, then for any simulator the indifferentiability advantage of the distinguisher is lower-bounded by $1 - (q_1 q_2)/qs - q_S^2/2^n$. (This is slightly simpler than the real bound, see Section 3.2.) What this lower bound states is that the simulator must make very close to $\min\{q_1 q_2, 2^{n/2}\}$ queries to prevent this distinguisher’s success. The result extends to structured underlying hash functions H as well, for example should H be MD-based.

To the best of our knowledge, our results are the first to show lower bounds on the number of queries an indifferentiability simulator must use. That a simulator must make a large number of queries hinders the utility of indifferentiability. When one uses the MRH composition theorem, the security of a scheme when using a monolithic RO must hold up to the number of queries the simulator makes. For example, in settings where one uses a hash function needing to be collision-resistant and attempts to conclude security via some (hypothetical) indifferentiability bound, our results indicate that the resulting security bound for the application can be at most $2^{n/4}$ instead of the expected $2^{n/2}$.

UPPER BOUNDS FOR SECOND ITERATES. We have ruled out good upper bounds on indifferentiability, but the question remains whether weak bounds exist. We provide proofs of indifferentiability for H^2 that hold up to about $2^{n/4}$ distinguisher queries (our lower bounds rule out doing better) when H is a RO. We provide some brief intuition about the proof. Consider an indifferentiability adversary making at most q_1, q_2 queries. The adversarial strategy of import is to compute long chains using the left oracle, and then try to “catch” the simulator in an inconsistency by querying it on a value at the end of the chain and, afterwards, filling in the intermediate values via further left and right queries. But the simulator can avoid being caught if it prepares long chains itself to help it answer queries consistently. Intuitively, as long as the simulator’s chains are a bit longer than q_1 hops, then the adversary cannot build a longer chain itself (being restricted to at most q_1 queries) and will never win. The full proofs of these results are quite involved, and so we defer more discussion until the body. We are unaware of any indifferentiability proofs that requires this kind of nuanced strategy by the simulator.

1.2 HMAC with Arbitrary Keys

HMAC was introduced by Bellare, Canetti, and Krawczyk [5] to be used as a pseudorandom function or message authentication code. It uses an underlying hash function H ; let H have block size d bits and output length n bits. Computing a hash $\text{HMAC}(K, M)$ works as follows [26]. If $|K| > d$ then redefine $K \leftarrow H(K)$. Let K' be K padded with sufficiently many zeros to get a d bit string. Then $\text{HMAC}(K, M) = H(K' \oplus \text{opad} \parallel H(K' \oplus \text{ipad} \parallel M))$ where opad and ipad are distinct d -bit constants. The original (provable security) analyses of HMAC focus on the setting that the key K is honestly generated and secret [3, 5]. But what has happened is that HMAC’s speed, ubiquity, and assumed security properties have lead it to be used in a wide variety of settings.

Of particular relevance are settings in which existing (or potential) proofs of security model HMAC as a keyed RO, a function that maps each key, message pair to an independent and uniform point. There are many examples of such settings. The HKDF scheme builds from HMAC a general-purpose key derivation function [27, 28] that uses as key a public, uniformly chosen salt. When used with a source of sufficiently high entropy, Krawczyk proves security using standard model techniques, but when not proves security assuming HMAC is a keyed RO [28]. PKCS#5 standardizes password-based key derivation functions that use HMAC with key being a (low-entropy) password [30]. Recent work provides the first proofs of security when modeling HMAC as a RO [9]. Ristenpart and Yilek [32], in the context of hedged cryptography [4], use HMAC in a setting whose cryptographic security models allow adversarially specified keys. Again, proofs model HMAC as a keyed RO.

As mentioned previously, we would expect a priori that one can show that HMAC is indifferentiable from a keyed RO even when the attacker can query arbitrary keys. Then one could apply the composition theorem of [29] to derive proofs of security for the settings just discussed.

WEAK KEY PAIRS IN HMAC. We are the first to observe that HMAC has weak key pairs. First, there exist $K \neq K'$ for which $\text{HMAC}(K, M) = \text{HMAC}(K', M)$. These pairs of keys arise because of HMAC’s ambiguous encoding of differing-length keys. Trivial examples of such “colliding” keys include any K, K' for which either $|K| < d$ and $K' = K \parallel 0^s$ (for any $1 \leq s \leq d - |K|$), or $|K| > d$ and $K' = H(K)$. Colliding keys enable an easy attack that distinguishes $\text{HMAC}(\cdot, \cdot)$ from a random function $\mathcal{R}(\cdot, \cdot)$, which also violates the indifferentiability of HMAC. On the other hand, as long as H is collision-resistant, two keys of the same length can never collide. Still, even if we restrict attention to (non-colliding) keys of a fixed length, there still exist weak key pairs, but of a different form that we term ambiguous. An example of an ambiguous key pair is K, K' of length d bits such that $K \oplus \text{ipad} = K' \oplus \text{opad}$. Because the second least significant bit of ipad and opad differ (see Section 4) and assuming $d > n - 2$, ambiguous key pairs of a fixed length k only exist for $k \in \{d - 1, d\}$. The existence of ambiguous key pairs in HMAC leads to negative results like those given for H^2 . In particular, we straightforwardly extend the H^2 distinguisher to give one that lower bounds the number of queries any indifferentiability simulator must make for HMAC.

UPPER BOUNDS FOR HMAC. Fortunately, it would seem that weak key pairs do not arise in typical applications. Using HMAC with keys of some fixed bit length smaller than $d - 1$ avoids weak key pairs. This holds for several applications, for example the recommendation with HKDF is to use n -bit uniformly chosen salts as HMAC keys. This motivates finding positive results for HMAC when one avoids the corner cases that allow attackers to exploit weak key pairs.

Indeed, as our main positive result, we prove that, should H be a RO or an MD hash with ideal compression functions, *HMAC is indifferentiable from a keyed RO for all distinguishers that do not query weak key pairs*. Our result holds for the case that the keys queried are of length d or less. This upper bound enjoys the best, birthday-bound level of concrete security possible (up to small constants), and provides the *first positive result about the indifferentiability of the HMAC construction*.

1.3 Discussion

The structural properties within H^2 and HMAC are, in theory, straightforward to avoid. Indeed, as mentioned above, Coron et al. [13] prove indifferentiable from a RO the construction $H^2(0^d \parallel M)$ where H is MD using a compression function with block size d bits and chaining value length $n \leq d$ bits. Analogously, our positive results about HMAC imply as a special case that $\text{HMAC}(K, M)$, for any fixed constant K , is indifferentiable from a RO.

We emphasize that we are unaware of any deployed cryptographic application for which the use of H^2 or HMAC leads to a vulnerability. Still, our results show that future applications should, in particular, be careful when using HMAC with keys which are under partial control of the attacker. More importantly, our results demonstrate the importance of provable security in the design of hash functions (and elsewhere in cryptography), as opposed to the more common

“attack-fix” cycle. For example, the hash-of-hash suggestion of Ferguson and Schneier [20] was motivated by preventing the extension attack. Unfortunately, in so doing they accidentally introduced a more subtle (although less dangerous) attack, which was not present on the original design.¹ Indeed, we discovered the subtlety of the problems within H^2 and HMAC, including our explicit attacks, only after attempting to prove indifferentiability of these constructions (with typical, good bounds). In contrast, the existing indifferentiability proofs of (seemingly) small modifications of these hash functions, such as $H^2(0^d \parallel M)$ [13], provably rule out these attacks.

1.4 Prior Work

There exists a large body of work showing hash functions are indifferentiable from a RO (c.f., [1, 7, 11–13, 15, 16, 23]), including analyses of variants of H^2 and HMAC. As mentioned, a construction called HMAC was analyzed in [13] but this construction is not HMAC as standardized. Krawczyk [28] suggests that the analysis of $H^2(0 \parallel M)$ extends to the case of HMAC, but does not offer proof.² HMAC has received much analysis in other contexts. Proofs of its security as a pseudorandom function under reasonable assumptions appear in [3, 5]. These rely on keys being uniform and secret, making the analyses inapplicable for other settings. Analysis of HMAC’s security as a randomness extractor appear in [14, 21]. These results provide strong information theoretic guarantees that HMAC can be used as a key derivation function, but only in settings where the source has a relatively large amount of min-entropy. This requirement makes the analyses insufficient to argue security in many settings of practical importance. See [28] for further discussion.

FULL VERSION. Due to space constraints, many of our technical results and proofs are deferred to the full version of this paper [17].

2 Preliminaries

NOTATION AND GAMES. We denote the empty string by λ . If $|X| < |Y|$ then $X \oplus Y$ signifies that the X is padded with $|Y| - |X|$ zeros first. For set \mathcal{X} and value x , we write $\mathcal{X} \leftarrow x$ to denote $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$. For non-empty sets $Keys$, Dom , and Rng with $|Rng|$ finite, a random oracle $f : Keys \times Dom \rightarrow Rng$ is a function taken randomly from the space of all possible functions $Keys \times Dom \rightarrow Rng$. We will sometimes refer to random oracles as keyed when $Keys$ is non-empty, whereas we omit the first parameter when $Keys = \emptyset$.

We use code-based games [10] to formalize security notions and within our proofs. In the execution of a game G with adversary \mathcal{A} , we denote by $G^{\mathcal{A}}$ the

¹ We note the prescience of the proposers of H^2 , who themselves suggested further analysis was needed [20].

² Fortunately, the HKDF application of [28] seems to avoid weak key pairs, and thus our positive results for HMAC appear to validate this claim [28] for this particular application.

event that the game outputs true and by $\mathcal{A}^G \Rightarrow y$ the event that the adversary outputs y . Fixing some RAM model of computation, our convention is that the running time $\text{Time}(\mathcal{A})$ of an algorithm \mathcal{A} includes its code size. Queries are unit cost, and we will restrict attention to the absolute worst case running time which must hold regardless of queries are answered.

HASH FUNCTIONS. A *hash function* $H[P]: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ is a family of functions from Dom to Rng , indexed by a set Keys , that possibly uses (black-box) access to an underlying primitive P (e.g., a compression function). We call the hash function *keyed* if Keys is non-empty, and key-less otherwise. (In the latter case, we omit the first parameter.) We assume that the number of applications of P in computing $H[P](K, M)$ is the same for all K, M with the same value of $|K| + |M|$. This allows us to define the *cost* of computing a hash function $H[P]$ on a key and message whose combined length is ℓ , denoted $\text{Cost}(H, \ell)$, as the number of calls to P required to compute $H[P](K, M)$ for K, M with $|K| + |M| = \ell$. For a keyed random oracle $\mathcal{R}: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$, we fix the convention that $\text{Cost}(\mathcal{R}, \ell) = 1$ for any ℓ for which there exists a key $K \in \text{Keys}$ and message $M \in \text{Dom}$ such that $|K| + |M| = \ell$.

A *compression function* is a hash function for which $\text{Dom} = \{0, 1\}^n \times \{0, 1\}^d$ and $\text{Rng} = \{0, 1\}^n$ for some numbers $n, d > 0$. Our focus will be on keyless compression functions, meaning those of the form $f: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^n$. Our results lift in a straightforward way to the dedicated-key setting [8]. The ℓ -th iterate of $H[P]$ is denoted $H^\ell[P]$, and defined for $\ell > 0$ by $H^\ell[P](X) = H[P](H[P](\cdots H[P](X)) \cdots)$ where the number of applications of H is ℓ . We let $H^0[P](X) = X$. We will often write H instead of $H[P]$ when the underlying primitive P is clear or unimportant.

MERKLE-DAMGÅRD. Let $\text{Pad}: \{0, 1\}^{\leq L} \rightarrow (\{0, 1\}^n)^+$ be an injective padding function. The one used in many of the hash functions within the SHA family outputs $M \parallel 10^r \parallel \langle |M| \rangle_{64}$ where $\langle |x| \rangle_{64}$ is the encoding of the length of M as a 64-bit string and r is the smallest number making the length a multiple of d . This makes $L = 2^{64} - 1$. The function $\text{MD}[f]: ((\{0, 1\}^n)^+) \rightarrow \{0, 1\}^n$ is defined as

$$\text{MD}[f](M) = f(f(\cdots f(f(IV, M_1), M_2), \cdots), M_k)$$

where $|M| = kd$ and $M_1 \parallel \cdots \parallel M_k$. The function $\text{SMD}[f]: \{0, 1\}^{\leq L} \rightarrow \{0, 1\}^n$ is defined as $\text{SMD}[f](M) = \text{MD}[f](\text{Pad}(M))$.

INDIFFERENTIABILITY FROM A RO. Let $\mathcal{R}: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ be a random oracle. Consider a hash construction $H[P]: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ from an ideal primitive P . Let game $\text{Real}_{H[P]}$ be the game whose main procedure runs an adversary $\mathcal{A}^{\text{Func}, \text{Prim}}$ and returns the bit that \mathcal{A} outputs. The procedure **Func** on input $K \in \text{Keys}$ and $M \in \text{Dom}$ returns $H[P](K, M)$. The procedure **Prim** on input X returns $P(X)$. For a simulator \mathcal{S} , let game $\text{Ideal}_{\mathcal{R}, \mathcal{S}}$ be the game whose main procedure runs an adversary $\mathcal{A}^{\text{Func}, \text{Prim}}$ and returns the bit that \mathcal{A} outputs. The procedure **Func** on input $K \in \text{Keys}$ and $M \in \text{Dom}$ returns $\mathcal{R}(K, M)$. The procedure **Prim** on input X returns $\mathcal{S}^{\mathcal{R}}(X)$. The indifferentiability advantage of \mathcal{D} is defined as

$$\mathbf{Adv}_{H[P], \mathcal{R}, \mathcal{S}}^{\text{indiff}}(\mathcal{D}) = \Pr \left[\text{Real}_{H[P]}^{\mathcal{D}} \Rightarrow y \right] - \Pr \left[\text{Ideal}_{\mathcal{R}, \mathcal{S}}^{\mathcal{D}} \Rightarrow y \right].$$

We focus on simulators that must work for any adversary, though our negative results extend as well to the weaker setting in which the simulator can depend on the adversary. The total query cost σ of an adversary \mathcal{D} is the cumulative cost of all its `Func` queries plus q_2 . (This makes σ the total number of P uses in game $\text{Real}_{H[P]}$. In line with our worst-case conventions, this means the same maximums hold in $\text{Ideal}_{\mathcal{R}, \mathcal{S}}$ although here it does not translate to P applications.)

We note that when `Keys` is non-empty, indifferentiability here follows [8] and allows the distinguisher to choose keys during an attack. This reflects the desire for a keyed hash function to be indistinguishable from a keyed random oracle for arbitrary uses of the key input.

3 Second Iterates and Their Security

Our investigation begins with the second iterate of a hash function, meaning $H^2(M) = H(H(M))$ where $H : \text{Dom} \rightarrow \text{Rng}$ for sets $\text{Dom} \supseteq \text{Rng}$. For simplicity, let $\text{Rng} = \{0, 1\}^n$ and assume that H is itself modeled as a RO. Is H^2 good in the sense of being like a RO? Given that we are modeling H as a RO, we would expect that the answer would be “yes”. The truth is more involved. As we’ll see in Section 4, similar subtleties exist in the case of the related HMAC construction.

We start with the following observations. When computing $H^2(M)$ for some M , we refer to the value $H(M)$ as an intermediate value. Then, we note that the value $Y = H^2(M)$ is in fact the intermediate value used when computing $H^2(X)$ for $X = H(M)$. Given $Y = H^2(M)$, then, one can compute $H^2(H(M))$ directly by computing $H(Y)$. That the hash value Y is also the intermediate value used in computing the hash of another message is cause for concern: other hash function constructions that are indifferentiable from a RO (c.f., [2, 7, 8, 13, 23]) explicitly attempt to ensure that outputs are *not* intermediate values (with overwhelming probability over the randomness of the underlying idealized primitive). Moreover, prior constructions for which hash values are intermediate values have been shown to *not* be indifferentiable from a RO. For example Merkle-Damgård-based iterative hashes fall to extension attacks [13] for this reason. Unlike with Merkle-Damgård, however, it is not immediately clear how an attacker might abuse the structure of H^2 .

We turn our attention to hash chains, where potential issues arise. For a hash function H , we define a hash chain $Y = (Y_0, \dots, Y_\ell)$ to be a sequence of $\ell + 1$ values where Y_0 is a message and $Y_i = H(Y_{i-1})$ for $1 \leq i \leq \ell$. Likewise when using H^2 a hash chain $Y = (Y_0, \dots, Y_\ell)$ is a sequence of $\ell + 1$ values where Y_0 is a message and $Y_i = H^2(Y_{i-1})$ for $1 \leq i \leq \ell$. We refer to Y_0 as the *start* of the hash chain and Y_ℓ as the *end*. Two chains Y, Y' are *non-overlapping* if no value in one chain occurs in the other, meaning $Y_i \neq Y'_j$ for all $0 \leq i \leq j \leq \ell$.

For any hash function and given the start and end of a hash chain $Y = (Y_0, \dots, Y_\ell)$, one can readily compute the start and end of a new chain with just

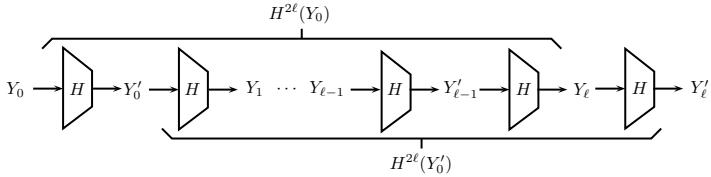


Fig. 1. Diagram of two hash chains $Y = (Y_0, \dots, Y_\ell)$ and $Y' = (Y'_0, \dots, Y'_\ell)$ for hash function H^2

two hash calculations. That is, set $Y'_0 \leftarrow H(Y_0)$ and $Y'_\ell \leftarrow H(Y_\ell)$. However, the chain $Y' = (Y'_0, \dots, Y'_\ell)$ and the chain Y overlap. For good hash functions (i.e., ones that behave like a RO) computing the start and end of a non-overlapping chain given the start and end of a chain Y_0, Y_ℓ requires at least ℓ hash computations (assuming $\ell \ll 2^{n/2}$).

Now consider H^2 . Given the start and end of a chain $Y = (Y_0, \dots, Y_\ell)$, one can readily compute a non-overlapping chain $Y' = (Y'_0, \dots, Y'_\ell)$ using just two hash computations instead of the expected 2ℓ computations. Namely, let $Y'_0 \leftarrow H(Y_0)$ and $Y'_\ell \leftarrow H(Y_\ell)$. Then these are the start and end of the chain $Y' = (Y'_0, \dots, Y'_\ell)$ because

$$H^{2\ell}(Y'_0) = H^{2\ell}(H(Y_0)) = H(H^{2\ell}(Y_0))$$

which we call the chain-shift property of H^2 . Moreover, assuming H is itself a RO outputting n -bit strings, the two chains Y, Y' do not overlap with probability at least $1 - (2\ell + 2)^2/2^n$. Figure 1 provides a pictorial diagram of the two chains Y and Y' .

3.1 A Vulnerable Application: Mutual Proofs of Work

In the last section we saw that the second iterate fails to behave like a RO in the context of hash chains. But the security game detailed in the last section may seem far removed from real protocols. For example, it's not clear where an attacker would be tasked with computing hash chains in a setting where it, too, was given an example hash chain. We suggest that just such a setting could arise in protocols in which parties want to assert to each other, in a verifiable way, that they performed some amount of computation. Such a setting could arise when parties must (provably) compare assertions of computational power, as when using cryptographic puzzles [18, 19, 24, 25, 33, 35]. Or this might work when trying to verifiably calibrate differing computational speeds of the two parties' computers. We refer to this task as a *mutual proof of work*.

MUTUAL PROOFS-OF-WORK. For the sake of brevity, we present an example hash-chain-based protocol and dispense with a more general treatment of mutual proofs of work. Consider the two-party protocol shown in the left diagram of Figure 2. Each party initially chooses a random nonce and sends it to the other. Then, each party computes a hash chain of some length —chosen by the

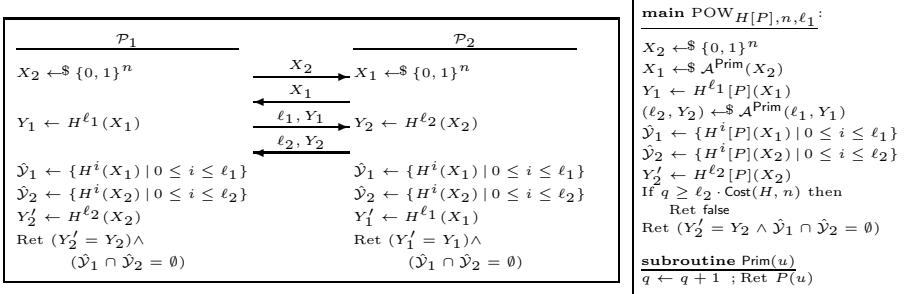


Fig. 2. Example protocol (**left**) and adversarial \mathcal{P}_2 security game (**right**) for mutual proofs of work

computing party— starting with the nonce chosen by the other party, and sends the chain’s output along with the chain’s length to the other party. At this point, both parties have given a witness that they performed a certain amount of work. So now, each party checks the other’s asserted computation, determining if the received value is the value resulting from chaining together the indicated number of hash applications and checking that the hash chains used by each party are non-overlapping. Note that unlike puzzles, which require fast verification, here the verification step is as costly as puzzle solution.

The goal of the protocol is to ensure that the other party did compute exactly their declared number of iterations. Slight changes to the protocol would lead to easy ways of cheating. For example, if during verification the parties did not check that the chains are non-overlapping, then \mathcal{P}_2 can easily cheat by choosing X_1 so that it can reuse a portion of the chain computed by \mathcal{P}_1 .

Security would be achieved should no cheating party succeed at convincing an honest party using less than ℓ_1 (resp. ℓ_2) work to compute Y_1 (resp. Y_2). The game $\text{POW}_{H[P], n, \ell_1}$ formalizes this security goal for a cheating \mathcal{P}_2 ; see the right portion of Figure 2. We let $\mathbf{Adv}_{H[P], n, \ell_1}^{\text{pow}}(\mathcal{A}) = \Pr[\text{POW}_{H[P], n, \ell_1}^{\mathcal{A}}]$. Note that the adversary \mathcal{A} only wins should it make $q < \ell_2 \cdot \text{Cost}(H, n)$ queries, where ℓ_2 is the value it declared and $\text{Cost}(H)$ is the cost of computing H . Again we will consider both the hash function $H[P](M) = P(M)$ that just applies a RO P and also $H^2[P](M) = P(P(M))$, the second iterate of a RO. In the former case the can make only $\ell_2 - 1$ queries and in the latter case $2\ell_2 - 1$.

When $H[P](M) = P(M)$, no adversary making $q < \ell_2$ queries to Prim can win the $\text{POW}_{H[P], n, \ell_1}$ game with high advantage. Intuitively, the reason is that, despite being given X_1 and Y_1 where $Y_1 = P^{\ell_1}(X_1)$, a successful attacker must still compute a full ℓ_2 -length chain and this requires ℓ_2 calls to P . A more formal treatment appears in the full version.

ATTACK AGAINST ANY SECOND ITERATE. Now let us analyze this protocol’s security when we use as hash function $H^2[P] = P(P(M))$ for a RO P : $\text{Dom} \rightarrow \text{Rng}$ with $\text{Rng} \subseteq \text{Dom}$. We can abuse the chain-shift property of H^2 in order to win the $\text{POW}_{H^2, P, n, \ell_1}$ game for any $n > 0$ and $\ell_1 > 2$. Our adversary \mathcal{A} works as follows. It receives X_2 and then chooses it’s nonce as $X_1 \leftarrow \text{Prim}(X_2)$. When

it later receives $Y_1 = P^{2\ell_1}(X_1)$, the adversary proceeds by setting $\ell_2 = \ell_1 + 1$ and setting $Y_2 \leftarrow \text{Prim}(Y_1)$. Then by the chain-shift property we have that

$$Y_2 = P(Y_1) = P(P^{2\ell_1}(X_1)) = P(P^{2\ell_1}(P(X_2))) = P^{2\ell_2}(X_2).$$

The two chains will be non-overlapping with high probability (over the coins used by P). Finally, \mathcal{A} makes only 2 queries to Prim , so the requirement that $q < 2\ell_2$ is met whenever $\ell_1 > 1$.

DISCUSSION. As far as we are aware, mutual proofs of work have not before been considered — the concept may indeed be of independent interest. A full treatment is beyond the scope of this work. We also note that, of course, it is easy to modify the protocols using H^2 to be secure. Providing secure constructions was not our goal, rather we wanted to show protocols which are insecure using H^2 but secure when H^2 is replaced by a monolithic RO. This illustrates how, hypothetically, the structure of H^2 could give rise to subtle vulnerabilities in an application.

3.2 Indifferentiability Lower and Upper Bounds

In this section we prove that *any* indifferentiability proof for the double iterate H^2 is subject to inherent quantitative limitations. Recall that indifferentiability asks for a simulator \mathcal{S} such that no adversary can distinguish between the pair of oracles $H^2[P], P$ and \mathcal{R}, \mathcal{S} where P is some underlying ideal primitive and \mathcal{R} is a RO with the same domain and range as H^2 . The simulator can make queries to \mathcal{R} to help it in its simulation of P . Concretely, building on the ideas behind the above attacks in the context of hash chains, we show that in order to withstand a differentiating attack with q queries, any simulator for $H^2[P]$, for *any* hash construction $H[P]$ with output length n , must issue *at least* $\Omega(\min\{q^2, 2^{n/2}\})$ queries to the RO \mathcal{R} . As we explain below, such a lower bound severely limits the concrete security level which can be inferred by using the composition theorem for indifferentiability, effectively neutralizing the benefits of using indifferentiability in the first place.

THE DISTINGUISHER. In the following, we let $H = H[P]$ be an *arbitrary* hash function with n -bit outputs relying on a primitive P , such as a fixed input-length random oracle or an ideal cipher. We are therefore addressing an arbitrary second iterate, and not focusing on some particular ideal primitive P (such as a RO as in previous sections) or construction H . Indeed, H could equally well be Merkle-Damgård and P an ideal compression function, or H could be any number of indifferentiable hash constructions using appropriate ideal primitive P .

Recall that Func and Prim are the oracles associated with construction and primitive queries to $H^2 = H^2[P]$ and P , respectively. Let w, ℓ be parameters (for now, think for convenience of $w = \ell$). The attacker $\mathcal{D}_{w,\ell}$ starts by issuing ℓ queries to Func to compute a *chain* of n -bit values $(x_0, x_1, \dots, x_\ell)$ where $x_i = H^2(x_{i-1})$ and x_0 is a random n -bit string. Then, it also picks a random index $j \in [1..w]$, and creates a list of n -bit strings $\mathbf{u}[1], \dots, \mathbf{u}[w]$ with $\mathbf{u}[j] = x_\ell$, and all remaining $\mathbf{u}[i]$ for $i \neq j$ are chosen uniformly and independently. Then, for all $i \in [1..w]$,

the distinguisher $\mathcal{D}_{w,\ell}$ proceeds in asking all Prim queries in order to compute $\mathbf{v}[i] = H(\mathbf{u}[i])$. Subsequently, the attacker compute $y_0 = H(x_0)$ via Prim queries, and also computes the chain $(y_0, y_1, \dots, y_\ell)$ such that $y_i = H^2(y_{i-1})$ by making ℓ Func queries. Finally, it decides to output 1 if and only if $y_\ell = \mathbf{v}[j]$ and x_ℓ as well as $\mathbf{v}[i]$ for $i \neq j$ are not in $\{y_0, y_1, \dots, y_\ell\}$. The attacker $\mathcal{D}_{w,\ell}$ therefore issues a total of 2ℓ Func queries and $(2w + 1) \cdot \text{Cost}(H, n)$ Prim queries.

In the real-world experiment, the distinguisher $\mathcal{D}_{w,\ell}$ outputs 1 with very high probability, as the condition $y_\ell = \mathbf{v}[j]$ always holds by the chain-shifting property of H^2 . In fact, the only reason for \mathcal{D} outputting 0 is that one of x_ℓ and $\mathbf{v}[i]$ for $i \neq j$ incidentally happens to be in $\{y_0, y_1, \dots, y_\ell\}$. The (typically small) probability that this occurs obviously depends on the particular construction $H[P]$ at hand; it is thus convenient to define the shorthand

$$p(H, w, \ell) = \Pr[\{x_\ell, H(U_1), \dots, H(U_{w-1})\} \cap \{y_0, y_1, \dots, y_\ell\} \neq \emptyset],$$

where $x_0, y_0, x_1, \dots, y_{\ell-1}, x_\ell, y_\ell$ are the intermediate value of a chain of $2\ell + 1$ consecutive evaluations of $H[P]$ starting at a random n -bit string x_0 , and U_1, \dots, U_{w-1} are further independent random n -bit values. In the full version of this paper we prove that for $H[P] = P = \mathcal{R}$ for a random oracle $\mathcal{R} : \{0, 1\}^* \rightarrow \{0, 1\}^n$ we have $p(H, w, \ell) = \Theta((w\ell + \ell^2)/2^n)$. Similar reasoning can be applied to essentially all relevant constructions.

In contrast, in the ideal-world experiment, we expect the simulator to be completely ignorant about the choice of j as long as it does not learn x_0 , and in particular it does not know j while answering the Prim queries associated with the evaluations of $H(\mathbf{u}[i])$. Consequently, the condition required for $\mathcal{D}_{w,\ell}$ to output 1 appears to force the simulator, for all $i \in [1..w]$, to prepare a distinct chain of ℓ consecutive \mathcal{R} evaluations ending in $\mathbf{v}[i]$, hence requiring $w \cdot \ell$ random oracle queries.

The following theorem quantifies the advantage achieved by the above distinguisher $\mathcal{D}_{w,\ell}$ in differentiating against any simulator for the construction $H[P]$. Its proof is given in the full version.

Theorem 1. [Attack against H^2] *Let $H[P]$ be an arbitrary hash construction with n -bit outputs, calling a primitive P , and let $\mathcal{R} : \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a random oracle. For all integer parameters $w, \ell \geq 1$, there exists an adversary $\mathcal{D}_{w,\ell}$ making 2ℓ Func -queries and $(w + 1) \cdot \text{Cost}(H, n)$ Prim -queries such that for all simulators \mathcal{S} ,*

$$\mathbf{Adv}_{H^2[P], \mathcal{R}, \mathcal{S}}^{\text{indiff}}(\mathcal{D}_{w,\ell}) \geq 1 - p(H, w, \ell) - \frac{5\ell^2}{2^{n+1}} - \frac{q_S \ell}{2^n} - \frac{q_S^2}{2^n} - \frac{q_S}{w \cdot \ell} - \frac{1}{w},$$

where q_S is the overall number of \mathcal{R} queries by \mathcal{S} when replying to $\mathcal{D}_{w,\ell}$'s Prim queries. ■

DISCUSSION. We now elaborate on Theorem 1. If we consider the distinguisher $\mathcal{D}_{w,\ell}$ from Theorem 1, we observe that by the advantage lower bound in the theorem statement, if $\ell, w \ll 2^{n/4}$ and consequently $p(H, w, \ell) \approx 0$, the number of queries made by the simulator, denoted $q_S = q_S(2\ell, w + 1)$ must satisfy $q_S = \Omega(w \cdot \ell) = \Omega(q_1 \cdot q_2)$ to ensure a sufficiently small indistinguishability

advantage. This in particular means that in the case where both q_1 and q_2 are large, the simulator must make a *quadratic* effort to prevent the attacker from distinguishing. Below, in Theorem 2, we show that this simulation effort is essentially optimal.

In many scenarios, this quadratic lower bound happens to be a problem, as we now illustrate. As a concrete example, let $\mathcal{SS} = (\text{key}, \text{sign}, \text{ver})$ be an arbitrary signature scheme signing n bits messages, and let $\widetilde{\mathcal{SS}}[\mathcal{R}] = (\widetilde{\text{key}}^{\mathcal{R}}, \widetilde{\text{sign}}^{\mathcal{R}}, \widetilde{\text{ver}}^{\mathcal{R}})$ for $\mathcal{R} : \{0, 1\}^* \rightarrow \{0, 1\}^n$ be the scheme obtained via the hash-then-sign paradigm such that $\widetilde{\text{sign}}^{\mathcal{R}}(\text{sk}, m) = \text{sign}(\text{sk}, \mathcal{R}(m))$. It is well known that for an adversary \mathcal{B} making q_{sign} signing and $q_{\mathcal{R}}$ random oracle queries, there exists an adversary \mathcal{C} making q_{sign} signing queries such that

$$\mathbf{Adv}_{\widetilde{\mathcal{SS}}[\mathcal{R}]}^{\text{uf-cma}}(\mathcal{B}^{\mathcal{R}}) \leq \frac{(q_{\text{sign}} + q_{\mathcal{R}})^2}{2^n} + \mathbf{Adv}_{\mathcal{SS}}^{\text{uf-cma}}(\mathcal{C}), \quad (1)$$

where $\mathbf{Adv}_{\widetilde{\mathcal{SS}}[\mathcal{R}]}^{\text{uf-cma}}(\mathcal{B}^{\mathcal{R}})$ and $\mathbf{Adv}_{\mathcal{SS}}^{\text{uf-cma}}(\mathcal{C})$ denote the respective advantages in the standard uf-cma game for security of signature schemes (with and without a random oracle, respectively). This in particular means that $\widetilde{\mathcal{SS}}$ is secure for q_{sign} and $q_{\mathcal{R}}$ as large as $\Theta(2^{n/2})$, provided \mathcal{SS} is secure for q_{sign} signing queries. However, let us now replace \mathcal{R} by $H^2[P]$ for an arbitrary construction $H = H[P]$. Then, for all adversaries \mathcal{A} making q_P queries to P and q_{sign} signing queries, we can combine the concrete version of the MRH composition theorem proven in [31] and (1) to infer that there exists an adversary \mathcal{C} and a distinguisher \mathcal{D} such that

$$\mathbf{Adv}_{\widetilde{\mathcal{SS}}[H^2[P]]}^{\text{uf-cma}}(\mathcal{A}^P) \leq \Theta\left(\frac{(q_{\text{sign}} \cdot q_P)^2}{2^n}\right) + \mathbf{Adv}_{\mathcal{SS}}^{\text{uf-cma}}(\mathcal{C}) + \mathbf{Adv}_{H^2[P], \mathcal{R}, \mathcal{S}}^{\text{indiff}}(\mathcal{D}),$$

where \mathcal{C} makes q_{sign} signing queries. Note that even if the term $\mathbf{Adv}_{H^2[P], \mathcal{R}, \mathcal{S}}^{\text{indiff}}(\mathcal{D})$ is really small, this new bound can only ensure security for the resulting signature scheme as long as $q_{\text{sign}} \cdot q_P = \Theta(2^{n/2})$, i.e., if $q_{\text{sign}} = q_P$, we only get security up to $\Theta(2^{n/4})$ queries, a remarkable loss with respect to the security bound in the random oracle model.

We note that of course this does *not* mean that $H^2[P]$ for a concrete H and P is unsuitable for a certain application, such as hash-then-sign. In fact, $H^2[P]$ may well be optimally collision resistant. However, our result shows that a sufficiently strong security level cannot be inferred from *any* indistinguishability statement via the composition theorem, taking us back to a direct ad-hoc analysis and completely loosing the one main advantage of having indistinguishability in the first place.

UPPER BOUND. Our negative results do not rule out positive results completely: there could be indistinguishability upper bounds, though for simulators that make around $\mathcal{O}(q^2)$ queries. Ideally, we would like upper bounds that match closely the lower bounds given in prior sections. We do so for the special case of $H^2[g](M) = g(g(M))$ for $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$ being a RO.

Theorem 2. *Let $q_1, q_2 \geq 0$ and $N = 2^n$. Let $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and $\mathcal{R} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be uniform random functions. Then there exists a simulator \mathcal{S} such that*

$$\mathbf{Adv}_{G[g], \mathcal{R}, \mathcal{S}}^{\text{indiff}}(\mathcal{D}) \leq \frac{2((4q_1 + 3)q_2 + 2q_1)^2}{N} + \frac{2((4q_1 + 3)q_2 + 2q_1)(q_1 + q_2)}{(N - 2q_2 - 2q_1)}$$

for any adversary \mathcal{D} making at most q_1 queries to its left oracle and at most q_2 queries to its right oracle. Moreover, for each query answer that it computes, \mathcal{S} makes at most $3q_1 + 1$ queries to RO and runs in time $\mathcal{O}(q_1)$. \square

The proof of the theorem appears in the full version of the paper. We note that the simulator used must know the maximum number of queries the attacker will make, but does not otherwise depend on the adversary's strategy. The security bound of the theorem is approximately $(q_1 q_2)^2/N$, implying that security holds up to $q_1 q_2 \approx 2^{n/2}$.

4 HMAC as a General-Purpose Keyed Hash Function

HMAC [5] uses a hash function to build a keyed hash function, i.e. one that takes both a key and message as input. Fix some hash function³ $H: \{0, 1\}^* \rightarrow \{0, 1\}^n$. HMAC assumes this function H is built by iterating an underlying compression function with a message block size of $d \geq n$ bits. We define the following functions:

$$\begin{aligned} F_K(M) &= H((\rho(K) \oplus \text{ipad}) \parallel M) \\ G_K(M) &= H((\rho(K) \oplus \text{opad}) \parallel M) \end{aligned} \quad \text{where } \rho(K) = \begin{cases} H(K) & \text{if } |K| > d \\ K & \text{otherwise.} \end{cases}$$

The two constants used are $\text{ipad} = 0x36^{d/8}$ and $\text{opad} = 0x5c^{d/8}$. These constants are given in hexadecimal, translating to binary gives $0x36 = 0011\ 0110_2$ and $0x5c = 0101\ 1100_2$. Recall that we have defined the \oplus operator so that, if $|K| < d$, it first silently pads out the shorter string by sufficiently many zeros before computing the bitwise xor. It will also be convenient to define $\text{xpad} = \text{ipad} \oplus \text{opad}$. The function $\text{HMAC}: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^n$ is defined by

$$\text{HMAC}(K, M) = G_K(F_K(M)) = (G_K \circ F_K)(M).$$

We sometimes write $\text{HMAC}_d[P]$, HMAC_d , or $\text{HMAC}[P]$ instead of HMAC when we want to make the reliance on the block size and/or an underlying ideal primitive explicit.

In the following sections, we will therefore analyze the security of HMAC in the sense of being indistinguishable from a keyed RO. As we will see, the story is more involved than one might expect.

4.1 Weak Key Pairs in HMAC

Towards understanding the indistinguishability of HMAC , we start by observing that the way HMAC handles keys gives rise to two worrisome classes of weak key pairs.

³ RFC 2104 defines HMAC over strings of bytes, but we chose to use bits to provide more general positive results — all our negative results lift to a setting in which only byte strings are used. Note also that for simplicity we assumed H with domain $\{0, 1\}^*$. In practice hash functions often do have some maximal length (e.g., 2^{64}), and in this case HMAC must be restricted to smaller lengths.

COLLIDING KEYS. We say that keys $K \neq K'$ *collide* if $\rho(K) \parallel 0^{d-|\rho(K)|} = \rho(K') \parallel 0^{d-|\rho(K')|}$. For any message M and colliding keys K, K' it holds that $\text{HMAC}(K, M) = \text{HMAC}(K', M)$. Colliding keys exist because of HMAC's ambiguous encoding of different-length keys. Examples of colliding keys include any K, K' for which $|K| < d$ and $K' = K \parallel 0^s$ where $1 \leq s \leq d - |K|$. Or any K, K' such that $|K| > d$ and $K' = H(K)$. As long as H is collision-resistant, two keys of the same length can never collide.

Colliding keys enable a simple attack against indifferentiability. Consider $\text{HMAC}[P]$ for any underlying function P . Then let \mathcal{A} pick two keys $K \neq K'$ that collide and an arbitrary message M . It queries its `Func` oracle on (K, M) and (K', M) to retrieve two values Y, Y' . If $Y = Y'$ then it returns 1 (guessing that it is in game $\text{Real}_{\text{HMAC}[P], \mathcal{R}}$) and returns 0 otherwise (guessing that it is in game $\text{Ideal}_{\mathcal{R}, \mathcal{S}}$). The advantage of \mathcal{A} is equal to $1 - 2^n$ regardless of the simulator \mathcal{S} , which is never invoked.

Note that this result extends directly to rule out related-key attack security [6] of HMAC as a PRF should a related-key function be available that enables deriving colliding keys.

AMBIGUOUS KEYS. A pair of keys $K \neq K'$ is *ambiguous* if $\rho(K) \oplus \text{ipad} = \rho(K') \oplus \text{opad}$. For any X , both $F_K(X) = G_{K'}(X)$ and $G_K(X) = F_{K'}(X)$ when K, K' are ambiguous. An example such pair is K, K' of length d bits for which $K \oplus K' = \text{xpad}$.

For any key K , there exists one key K' that is easily computable and for which K, K' are ambiguous: set $K' = \rho(K) \oplus \text{xpad}$. Finding a third key K'' that is also ambiguous with K is intractable should H be collision resistant. The easily-computable K' will not necessarily have the same length as K . In fact, there exist ambiguous key pairs of the same length k only when $k \in \{d-1, d\}$. For a fixed length shorter than $d-1$, no ambiguous key pairs exist due to the fact that the second least significant bit of xpad is 1. For a fixed length longer than d bits, if $n < d-1$ then no ambiguous key pairs exist and if $n \geq d-1$ then producing ambiguous key pairs would require finding K, K' such that $H(K) \oplus H(K')$ equals the first n bits of xpad . This is intractable for any reasonable hash function H .

Ambiguous key pairs give rise to a chain-shift like property. Let M be some message and K, K' be an ambiguous key pair. Then, we have that $\rho(K') = \rho(K) \oplus \text{xpad}$ and so $F_K(M) = G_{K'}(M)$. Thus,

$$\text{HMAC}(K', F_K(M)) = G_{K'}(\text{HMAC}(K, M)).$$

As with H^2 , this property gives rise to problems in the context of hash chains. A hash chain $Y = (K, Y_0, \dots, Y_\ell)$ is a key K , a message Y_0 , and a sequence of ℓ values $Y_i = H(K, Y_{i-1})$ for $1 \leq i \leq \ell$. So a keyed hash chain $Y = (K, Y_0, \dots, Y_\ell)$ for HMAC has $Y_i = \text{HMAC}(K, Y_{i-1})$ for $1 \leq i \leq \ell$. Given K, Y_0, Y_ℓ for a chain $Y = (K, Y_0, \dots, Y_\ell)$, it is easy for an adversary to compute the start and end of a new chain $Y' = (K', Y'_0, \dots, Y'_\ell)$ that does not overlap with Y . See Figure 3. In the full version, we detail how this structure can be abused in the context of an HMAC-based mutual proofs of work protocol. We also give an analogue of

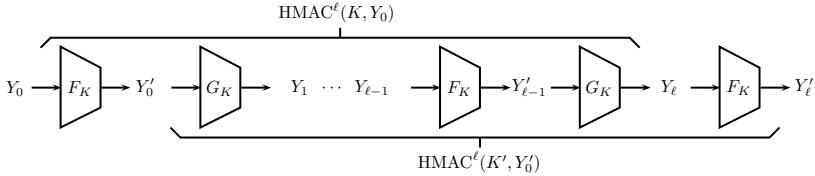


Fig. 3. Diagram of two hash chains $(K, Y) = (Y_0, \dots, Y_\ell)$ and $(K', Y') = (Y'_0, \dots, Y'_\ell)$ for HMAC where $\rho(K') = \rho(K) \oplus \text{xpad}$

Theorem 1, i.e., a lower bound on the indifferentiability of HMAC from a RO when ambiguous key pairs can be queried.

4.2 Indifferentiability of HMAC with Restricted Keys

We have seen that HMAC's construction gives rise to two kinds of weak key pairs that can be abused to show that HMAC is not indifferentiable from a keyed RO (with good bounds). But weak key pairs are serendipitously avoided in most applications. For example, the recommended usage of HKDF [28] specifies keys of a fixed length less than $d - 1$. Neither kind of weak key pairs exist within this subset of the key space.

While one can show indifferentiability for a variety of settings in which weak key pairs are avoided, we focus for simplicity on the case mentioned above. That is, we restrict to keys K for which $|K| = k$ and k is a fixed integer different less than $d - 1$. The full version provides a more general set of results, covering also, for example, use of HMAC with a fixed key of any length less than or equal to d .

As our first positive result, we have the following theorem, which establishes the security of HMAC when modeling the underlying hash function as a RO.

Theorem 3. Fix $d, k, n > 0$ with $k < d - 1$. Let $P: \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a RO, and consider $\text{HMAC}_d[P]$ restricted to k -bit keys. Let $\mathcal{R}: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a keyed RO. Then there exists a simulator \mathcal{S} such that for any distinguisher \mathcal{A} whose total query cost is σ it holds that

$$\mathbf{Adv}_{\text{HMAC}_d[P], \mathcal{R}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) \leq \mathcal{O}\left(\frac{\sigma^2}{2^n}\right)$$

\mathcal{S} makes at most q_2 queries and runs in time $\mathcal{O}(q_2 \log q_2)$ where q_2 is the number of Prim queries made by \mathcal{A} . \square

The use of $\mathcal{O}(\cdot)$ just hides small constants. The proof is given in the full version. Combining Theorem 3 with the indifferentiability composition theorem allows us to conclude security for $\text{HMAC}_d[H]$ for underlying hash function H that is, itself, indifferentiable from a RO. For example, should H be one of the proven-indifferentiable SHA-3 candidates. This does not, however, give us a security guarantee should H not be indifferentiable from a RO, as is the case with MD based hash functions. We therefore also prove, in the full version, the following

theorem that establishes indifferentiability of HMAC using an underlying hash function built via the strengthened Merkle-Damgård (SMD) domain extension transform.

Theorem 4. Fix $d, k, n > 0$ with $k < d - 1$ and $d \geq n$. Let $f: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^n$ be a RO and consider $\text{HMAC}_d[\text{SMD}[f]]$ restricted to k -bit keys. Let $\mathcal{R}: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a keyed RO. Then there exists a simulator \mathcal{S} such that for any distinguisher \mathcal{A} whose total query cost is $\sigma \leq 2^{n-2}$ it holds that

$$\mathbf{Adv}_{\text{HMAC}_d[\text{SMD}[f]], \mathcal{R}, \mathcal{S}}^{\text{indiff}}(\mathcal{A}) \leq \mathcal{O}\left(\frac{\sigma^2}{2^n}\right)$$

\mathcal{S} makes at most q_2 queries and runs in time $\mathcal{O}(q_2 \log q_2)$ where q_2 is the number of Prim queries by \mathcal{A} . \square

We note that the restriction to $\sigma \leq 2^{n-2}$ in the theorem statement is just a technicality to make the bound simpler and likewise the use of $\mathcal{O}(\cdot)$ in the advantage statement hides just a small constant.

Unlike our positive results about H^2 , the bounds provided by Theorems 3 and 4 match, up to small constants, results for other now-standard indifferentiable constructions (c.f., [13]). First, the advantage bounds both hold up to the birthday bound, namely $\sigma \approx 2^{n/2}$. Second, the simulators are efficient and, specifically, make at most one query per invocation. All this enables use of the indifferentiability composition theorem in a way that yields strong, standard concrete security bounds.

Acknowledgments. The authors thank Hugo Krawczyk for providing significant feedback and suggestions, in particular encouraging the authors to include positive results for the indifferentiability of HMAC; Niels Ferguson for in-depth discussions regarding the security of H^2 ; and the anonymous reviewers for their helpful suggestions. Dodis was supported in part by NSF grants CNS-1065134, CNS-1065288, CNS-1017471, CNS-0831299. Ristenpart was supported in part by NSF grant CNS-1065134. Steinberger is supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174, and by NSF grant 0994380. Tessaro was supported in part by NSF grants CCF-0915675, CCF-1018064, and DARPA contracts FA8750-11-C-0096, FA8750-11-2-0225.

References

1. Andreeva, E., Mennink, B., Preneel, B.: On the Indifferentiability of the Grøstl Hash Function. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 88–105. Springer, Heidelberg (2010)
2. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-Property-Preserving Iterated Hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)

3. Bellare, M.: New Proofs for **NMAC** and **HMAC**: Security Without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
4. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged Public-Key Encryption: How to Protect against Bad Randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
5. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
6. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
7. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
8. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
9. Bellare, M., Ristenpart, T., Tessaro, S.: Multi-Instance Security and Its Application to Password-Based Cryptography. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 312–329. Springer, Heidelberg (2012)
10. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
12. Chang, D., Nandi, M.: Improved Indifferentiability Security Analysis of chopMD Hash Function. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 429–443. Springer, Heidelberg (2008)
13. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
14. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004)
15. Dodis, Y., Reyzin, L., Rivest, R.L., Shen, E.: Indifferentiability of Permutation-Based Compression Functions and Tree-Based Modes of Operation, with Applications to MD6. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 104–121. Springer, Heidelberg (2009)
16. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging Merkle-Damgård for Practical Applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Heidelberg (2009)
17. Dodis, Y., Ristenpart, T., Steinberger, J., Tessaro, S.: To Hash or Not to Hash, Again? On the Indifferentiability of the Second Iterate and HMAC (2012); Full version of this paper. Available from authors' websites
18. Dwork, C., Naor, M.: Pricing via Processing or Combatting Junk Mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993)

19. Dwork, C., Naor, M., Wee, H.: Pebbling and Proofs of Work. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 37–54. Springer, Heidelberg (2005)
20. Ferguson, N., Schneier, B.: Practical cryptography. Wiley (2003)
21. Fouque, P.-A., Pointcheval, D., Zimmer, S.: HMAC is a randomness extractor and applications to TLS. In: Abe, M., Gligor, V. (eds.) ASIACCS 2008: 3rd Conference on Computer and Communications Security, pp. 21–32. ACM Press (March 2008)
22. Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., Stewart, L.: An Extension to HTTP: Digest Access Authentication. RFC 2069 (Proposed Standard) (January 1997); Obsoleted by RFC 2617
23. Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
24. Juels, A., Brainard, J.G.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: ISOC Network and Distributed System Security Symposium – NDSS 1999. The Internet Society (February 1999)
25. Karame, G.O., Čapkun, S.: Low-Cost Client Puzzles Based on Modular Exponentiation. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 679–697. Springer, Heidelberg (2010)
26. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (February 1997)
27. Krawczyk, H., Eronen, P.: HMAC-based extract-and-expand key derivation function (HKDF). RFC 5869 (Proposed Standard) (January 2010)
28. Krawczyk, H.: Cryptographic Extraction and Key Derivation: The HKDF Scheme. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 631–648. Springer, Heidelberg (2010)
29. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
30. PKCS #5: Password-based cryptography standard (RFC 2898). RSA Data Security, Inc., Version 2.0 (September 2000)
31. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with Composition: Limitations of the Indifferentiability Framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011)
32. Ristenpart, T., Yilek, S.: When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In: Network and Distributed Systems Security– NDSS 2010. ISOC (2010)
33. Stebila, D., Kuppusamy, L., Rangasamy, J., Boyd, C., Gonzalez Nieto, J.: Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 284–301. Springer, Heidelberg (2011)
34. Tsudik, G.: Message authentication with one-way hash functions. In: Proceedings IEEE INFOCOM 1992, vol. 3, pp. 2055–2059. IEEE (1992)
35. Wang, X.F., Reiter, M.K.: Defending against denial-of-service attacks with puzzle auction. In: IEEE Symposium on Security and Privacy, pp. 78–92 (2003)

New Preimage Attacks against Reduced SHA-1

Simon Knellwolf^{1,*} and Dmitry Khovratovich²

¹ ETH Zurich and FHNW, Switzerland

² Microsoft Research Redmond, USA

Abstract. This paper shows preimage attacks against reduced SHA-1 up to 57 steps. The best previous attack has been presented at CRYPTO 2009 and was for 48 steps finding a two-block preimage with incorrect padding at the cost of $2^{159.3}$ evaluations of the compression function. For the same variant our attacks find a one-block preimage at $2^{150.6}$ and a correctly padded two-block preimage at $2^{151.1}$ evaluations of the compression function. The improved results come out of a differential view on the meet-in-the-middle technique originally developed by Aoki and Sasaki. The new framework closely relates meet-in-the-middle attacks to differential cryptanalysis which turns out to be particularly useful for hash functions with linear message expansion and weak diffusion properties.

Keywords: SHA-1, preimage attack, differential meet-in-the-middle.

1 Introduction

Hash functions are an important cryptographic primitive that play a crucial role in numerous security protocols. They are supposed to satisfy at least three security requirements which are collision resistance, preimage resistance, and second preimage resistance. Here, “resistance” means the absence of any specific technique that allows to find collisions, preimages, or second preimages faster than a generic algorithm. If the hash output is n bits long, a generic algorithm requires $2^{n/2}$ evaluations of the hash function for finding a collision, for preimages and second preimages 2^n evaluations are required in average.

In the past, collision resistance of hash functions had been studied much more intensively than preimage resistance. This can be attributed to differential cryptanalysis as a very powerful tool to accelerate the collision search [6]. No such tool was available for the preimage search and the few published attacks were based on ad hoc methods. Notable examples are the first attacks on GOST [13], MD4 [12], and reduced variants of SHA-0/1 [5]. The situation changed with the introduction of the meet-in-the-middle technique into hash function cryptanalysis. Originally, the technique was used for block ciphers, starting with Diffie and Hellman [8] showing that double encryption under two different keys does not double the security level, and later by Chaum and Evertse [7] for key recovery attacks on reduced DES.

* Most of this work was done during an internship at Microsoft Research Redmond.

Only recently, Aoki and Sasaki translated the approach to the hash function context. In a series of papers they developed and refined a framework that resulted in the first preimage attack on MD5 and the best results on reduced variants of SHA-1, the SHA-2 family, and similar hash functions [1–3, 16–19]. Guo, Ling, Rechberger, and Wang [9] obtained improved results on MD4, SHA-2, and Tiger. A notable technical contribution has been made by Khovratovich, Rechberger, and Savelieva [10] with the formalization of the initial structure technique in terms of complete bipartite graphs (bicliques). The derived algorithms enhance meet-in-the-middle attacks using tools from differential cryptanalysis. Applications include key recovery attacks on AES [4] and preimage attacks on reduced variants of the SHA-3 finalist Skein and members of the SHA-2 family [10].

Most of the existing meet-in-the-middle framework has been developed for preimage attacks against hash functions with permutation based message expansions such as MD5. Even though the techniques have been generalized, notably in [3] to the linear message expansion of SHA-1, the original terminology did not translate suitably to these algorithms.

Technical Contribution of This Work. We carry on the simple and elegant differential view suggested by bicliques to other techniques such as partial matching, indirect partial matching, partial fixing, and probabilistic matching. Indeed, these techniques become quite natural from a differential perspective. Finding the attack parameters reveals to be equivalent to finding two sets of suitable related-key differentials. Finding high probability differentials is a well studied problem from collision attacks and insights can be reused. Two algorithms are proposed to find suitable attack parameters. They facilitate a systematic security evaluation while previous meet-in-the-middle attacks seem to heavily rely on elaborated by hand analysis and intuition. The framework applies particularly well to hash functions with linear message expansion, which is demonstrated by various new attacks against reduced variants of SHA-1.

SHA-1 and Improved Results. The SHA-1 hash function was specified in 1995 by the U.S. National Security Agency as a successor of SHA-0. No weakness has been found in the full SHA-1 until 2005, when Wang, Yin, and Yu presented astonishing collision attacks [21]. These attacks let the U.S. National Institute of Standards and Technology recommend the use of SHA-2 instead of SHA-1. However, SHA-1 is still widely used in practice and it is still believed preimage and second preimage resistant. De Cannière and Rechberger [5] presented the first dedicated preimage attack on reduced SHA-1. They could break 44 steps with a complexity of 2^{157} compression function evaluations using an approach that could not be extended so far. Aoki and Sasaki [3] presented an attack on 48 steps with a complexity of $2^{159.3}$ using the meet-in-the-middle technique, but their attack only finds messages without padding. Finding a correct padding makes the analysis more complicated and typically leads to higher attack complexities or to unpractically long messages. No progress has been made since 2009. Our results improve the existing results in several directions: variants with more steps can

Table 1. Preimage attacks against reduced SHA-1. If not stated otherwise, proper preimages with a correct padding are computed.

Steps	Complexity	# Blocks	Reference	Remark
44	$2^{157.0}$	1	[5]	
48	$2^{156.9}$	1	[3]	pseudo-preimage, no padding
48	$2^{159.3}$	2	[3]	no padding
44	$2^{146.2}$	1	Section 3.2	no padding
48	$2^{150.6}$	1	"	"
56	$2^{157.9}$	1	"	"
57	$2^{158.7}$	1	"	"
48	$2^{149.2}$	1	Section 3.3	pseudo-preimage
59	$2^{156.8}$	1	"	"
60	$2^{157.5}$	1	"	"
48	$2^{151.1}$	2	Section 3.4	
56	$2^{158.1}$	2	"	
57	$2^{158.8}$	2	"	

be attacked, significantly lower complexities are obtained for previously attacked variants, and correctly padded (short!) messages can be computed. The results are summarized in Table 1 and illustrated in Fig. 4 at the end of the paper.

Organization. Section 2 describes the new differential perspective on the meet-in-the-middle attack. Section 3 applies the framework to SHA-1 leading to various improved results. Section 4 briefly discusses a slight optimization of the generic brute-force search which can serve as a minimal benchmark for actual attacks. Section 5 summarizes and concludes the paper.

2 Meet-in-the-Middle: A Differential Framework

The compression function of SHA-1 and other dedicated hash functions can be seen as a block cipher used in Davies-Meyer mode, albeit with unusual key and block length. Let $F : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a block cipher with block length n and key length κ . Finding a preimage for an n -bit target value H is the problem of finding M such that $H = F(M, IV) + IV$, where IV is an initial vector. In the following, $P = IV$ and $C = H - IV$. Then, finding a preimage is equivalent to finding a right key for the plaintext/ciphertext pair (P, C) . A generic algorithm tests 2^n random messages such that one preimage is expected to be found ($\kappa > n$ is assumed).

2.1 Differential View on the Meet-in-the-Middle Technique

We now describe our new interpretation of the meet-in-the-middle technique. First, F is divided into two parts, $F = F_2 \circ F_1$. Then, from a differential point of view, the attacker tries to find two linear spaces $D_1, D_2 \subset \{0, 1\}^\kappa$ as follows:

- $D_1 \cap D_2 = \{0\}$.
- For each $\delta_1 \in D_1$ there is a $\Delta_1 \in \{0, 1\}^n$ such that

$$\Pr[\Delta_1 = F_1(M, P) \oplus F_1(M \oplus \delta_1, P)] = 1 \quad (1)$$

for uniformly chosen M , that is, $(\delta_1, 0) \rightarrow \Delta_1$ is a related-key differential for F_1 with probability 1.

- For each $\delta_2 \in D_2$ there is a $\Delta_2 \in \{0, 1\}^n$ such that

$$\Pr[\Delta_2 = F_2^{-1}(M, C) \oplus F_2^{-1}(M \oplus \delta_2, C)] = 1 \quad (2)$$

for uniformly chosen M , that is, $(\delta_2, 0) \rightarrow \Delta_2$ is a related-key differential for F_2^{-1} with probability 1.

The first condition makes sure that the search space can be partitioned into affine sets of the form $M \oplus D_1 \oplus D_2$. If D_1 and D_2 both have dimension d these sets contain 2^{2d} different messages. Each such set can be searched for a preimage by computing 2^d times F_1 and 2^d times F_2^{-1} using Algorithm 1. The algorithm computes two lists L_1 and L_2 . A match between the two lists identifies a preimage. The case $d = 1$ is illustrated by Fig. 1.

The second and the third condition make sure that the algorithm always answers correctly. Using (1) and (2) it follows that $L_1[\delta_2] = L_2[\delta_1]$ (in the last loop of Algorithm 1) is equivalent to $F_1(M \oplus \delta_1 \oplus \delta_2, P) = F_2^{-1}(M \oplus \delta_1 \oplus \delta_2, C)$, which is true if and only if $M \oplus \delta_1 \oplus \delta_2$ is a preimage.

Algorithm 1. Testing $M \oplus D_1 \oplus D_2$ for a preimage

Input: $D_1, D_2 \subset \{0, 1\}^\kappa$, $M \in \{0, 1\}^\kappa$

Output: A preimage if one is contained in $M \oplus D_1 \oplus D_2$.

```

for all  $\delta_2 \in D_2$  do
    Compute  $L_1[\delta_2] = F_1(M \oplus \delta_2, P) \oplus \Delta_2$ .
end for
for all  $\delta_1 \in D_1$  do
    Compute  $L_2[\delta_1] = F_2^{-1}(M \oplus \delta_1, C) \oplus \Delta_1$ .
end for
for all  $(\delta_1, \delta_2) \in D_1 \times D_2$  do
    if  $L_1[\delta_2] = L_2[\delta_1]$  then
        return  $M \oplus \delta_1 \oplus \delta_2$ 
    end if
end for
return No preimage in  $M \oplus D_1 \oplus D_2$ 

```

Complexity Analysis. Algorithm 1 has to be repeated 2^{n-2d} times in order to test 2^n messages. If we denote by Γ_1 and Γ_2 the cost of one evaluation of F_1 and F_2 , respectively, this results in a total complexity of $2^{n-2d}(2^d\Gamma_1 + 2^d\Gamma_2) = 2^{n-d}\Gamma$, where Γ is the cost of F . Depending on the implementation, memory is required to store the list L_1 and/or L_2 . Both lists have length 2^d and entries of about $n + d$ bits.

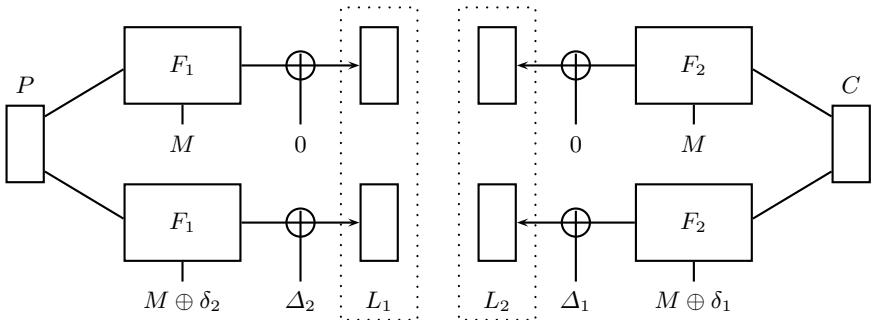


Fig. 1. Illustration of the meet-in-the-middle attack: A match between the lists L_1 and L_2 identifies a preimage. Here, the D_1 and D_2 only have dimension 1 which allows to test four messages at the cost of two. In general, 2^{2d} messages are tested at the cost of 2^d , where d is the dimension of D_1 and D_2 .

Remark. Using non-zero output differences Δ_1 and Δ_2 corresponds to the idea of indirect matching, introduced in [1], which is a rather advanced matching technique in the Aoki-Sasaki framework. From a differential point of view it appears quite natural.

2.2 Using Probabilistic and Truncated Differentials

Advanced matching techniques such as partial matching [2], indirect partial matching [1], partial fixing [1], and variants of probabilistic matching [10, 20] have a straightforward interpretation in terms of differentials. The conditions on the related-key differentials $(\delta_i, 0) \rightarrow \Delta_i$ are relaxed. Instead of differentials with probability 1 on the full state, probabilistic differentials with truncated output differences can be considered. Truncated differentials go back to Knudsen [11].

For a truncation mask $T \in \{0, 1\}^n$ we denote by $=_T$ equality on those bits of T which are 1, that is, $A =_T B$ if and only if $A \wedge T = B \wedge T$, where \wedge denotes bitwise AND. Instead of (1) and (2), the following probabilities should be high (but can be smaller than 1):

$$p_1 = \Pr[\Delta_1 =_T F_1(M, P) \oplus F_1(M \oplus \delta_1, P)], \quad (3)$$

$$p_2 = \Pr[\Delta_2 =_T F_2^{-1}(M, C) \oplus F_2^{-1}(M \oplus \delta_2, C)]. \quad (4)$$

The only thing that has to be changed in Algorithm 1 is in the last loop, where $=$ has to be replaced with $=_T$. However, the answer of the algorithm is not always correct when using probabilistic and/or truncated differentials which has consequences on the complexity of the attack.

Complexity Analysis. Testing a message $M \oplus \delta_1 \oplus \delta_2$ by the modified variant of Algorithm 1 is best formulated as a hypothesis test with null hypothesis

$$H_0 : M \oplus \delta_1 \oplus \delta_2 \text{ is a preimage.}$$

H_0 is rejected if $F_1(M \oplus \delta_2, P) \oplus \Delta_2 \neq_T F_2^{-1}(M \oplus \delta_1, C) \oplus \Delta_1$. H_0 is falsely rejected with probability $\alpha = 1 - p_1 p_2$ (type I error probability). On the other hand, if H_0 is false, we fail to reject it with probability $\beta = 1/r$ (type II error probability), where r is the Hamming weight of T . As a consequence, returned messages are only *candidate* preimages which have to be retested and the total number of tested messages has to be increased. If $\bar{\alpha}$ is the average type I error probability over all $(\delta_1, \delta_2) \in D_1 \times D_2$, $2^n/(1 - \bar{\alpha})$ messages have to be tested in order to compensate a fraction of $\bar{\alpha}$ preimages that is falsely rejected. This results in a total complexity of

$$(2^{n-d} \Gamma + 2^{n-r} \Gamma_{re}) / (1 - \bar{\alpha}),$$

where Γ_{re} is the cost of retesting a candidate preimage.

2.3 Splice and Cut

The splice and cut technique has been first used in [2]. The idea is to start the computations at an intermediate state, connecting the last and the first step via the feed-forward of the Davies-Meyer mode. The technique is fully compatible with our differential framework. Formally, the computation of F is cut into $F = F_2 \circ F_1$ and, for a given target value H , a new function F' is defined by $F'(M, V) = F_1(M, H - F_2(M, V))$. Then, the meet-in-the-middle technique is used to find M such that $F'(M, V) = V$ for some V . This is equivalent to $F(M, H - F_2(M, V)) + H - F_2(M, V) = H$, that is, M is a pseudo-preimage. Typically, the pseudo-preimage attack is then generically transformed into a preimage attack as described in [14, Fact 9.99].

2.4 Bicliques

In [19] the initial structure technique has been introduced as an extension to splice and cut. In [10], the technique is formalized in terms of bicliques. The idea is to start the computations not from a single state, but from a precomputed structure of states that covers several rounds. Formally, the computation of F is divided into three parts, $F = F_3 \circ F_2 \circ F_1$, and bicliques are constructed for one of them, say for F_3 . A biclique for F_3 is a tuple $\{M, D_1, D_2, Q_1, Q_2\}$ where M is a message, the D_i are linear difference spaces of dimension d , and the Q_i are lists of 2^d states $Q_i[\delta]$, for $\delta \in D_i$, such that for all $(\delta_1, \delta_2) \in D_1 \times D_2$: $Q_2[\delta_2] = F_3(M \oplus \delta_1 \oplus \delta_2, Q_1[\delta_1])$. With such a biclique, the set $M \oplus D_1 \oplus D_2$ can be searched for candidate pseudo-preimages by testing $F_1(M \oplus \delta_2, H - Q_2[\delta_2]) \oplus \Delta_2 =_T F_2^{-1}(M \oplus \delta_1, Q_1[\delta_1]) \oplus \Delta_1$. This requires only 2^d computations of F_1 and 2^d computations of F_2 . If the amortized cost of constructing many bicliques is negligible, the number of attacked rounds is increased by the rounds of F_3 without actually increasing the total complexity of the attack.

2.5 Special Case: Linear Message Expansion

The differential view is particularly useful if the underlying block cipher F uses a linear key expansion and relatively small round keys. In the following, suppose

that F performs N rounds and that the key expansion can be described by N linear maps $\varphi_i : \{0, 1\}^\kappa \rightarrow \{0, 1\}^w$, for $0 \leq i < N$, where w is the size of the round keys and $\varphi_i(K)$ is the i -th round key derived from K .

Differences as Kernel Elements. If D_1 lies in the kernel of φ_i no differences are introduced at round i . For some $k > 0$, the attacker can choose D_1 as a subspace of $\bigcap_{i=0}^{k-1} \ker(\varphi_i)$. This makes that no differences are introduced in the first k rounds. Similar, D_2 can be chosen as a subspace of $\bigcap_{i=N-k}^{N-1} \ker(\varphi_i)$. Then, $2k$ rounds can be attacked without advanced matching techniques. By a careful choice of D_1 and D_2 the attack extends to more rounds by using probabilistic and truncated differentials. The kernels can be computed by linear algebra or, as in the case of SHA-1, by using the message expansion in forward and backward direction.

Remark. In [3] kernels of message expansion have been used already. The kernel elements have not been interpreted as differences, but a sophisticated linear transformation matrix has been used to derive a “chunk separation” and “neutral words”. To make this possible, the condition has been imposed that kernel elements for the two chunks must not have overlapping non-zero words. From a differential point of view it is clear that only $D_1 \cap D_2 = \{0\}$ is required which gives us more freedom in the choice of the attack parameters.

3 Application to SHA-1

We now apply the attack framework to SHA-1, which leads to various new results. Different types of “preimages” will be obtained:

One-block preimages. We first compute one-block preimages without padding up to 57 steps. These attacks are comparable to those by Aoki and Sasaki in [3] which find two-block preimages without padding up to 48 steps. Then, we show how to obtain correctly padded one-block preimages up to 52 steps.

One-block pseudo-preimages. A pseudo-preimage is a “preimage” that uses an incorrect IV. The additional freedom degree allows to use bicliques and the splice and cut technique. This results in pseudo-preimage attacks up to 60 steps. A careful choice of the biclique position avoids additional restrictions from the padding rule such that all our attacks allow to put a correct padding to the pseudo-preimage.

Two-block preimages. Combining the above techniques enables us to compute two-block preimages with a correct padding almost as efficiently as one-block preimages without padding. As a result, we obtain preimage attacks up to 57 steps finding correctly padded two-block preimages.

Before describing the attacks, a short description of SHA-1 is given.

3.1 Description of SHA-1

The hash function SHA-1 was designed by the U.S. National Security Agency and is specified in [15]. The construction follows the Merkle-Damgård principle with a block cipher based compression function in Davies-Meyer mode. A message is padded to a length which is a multiple of 512 bits. This is done by appending a 1, a variable number of 0s, and the length in bits as a 64-bit integer. After the padding, the message is split into 512-bit blocks $M = (M^0, \dots, M^{L-1})$ which are iteratively processed according to

$$H^0 = \text{IV}, \quad H^{l+1} = F(M^l, H^l) + H^l, \quad 0 \leq l < L.$$

The chaining values H^l consist of five 32-bit words. The last chaining value is the output of the hash function.

The function F can be considered as a block cipher with key length $\kappa = 512$ and block length $n = 160$. The computation of $F(M^l, H^l)$ has two parts: First, the message block of 16 words, denoted by $M^l = (M_0, \dots, M_{15})$, is expanded to an extended message of 80 words, denoted by $W = (W_0, \dots, W_{79})$:

$$\begin{aligned} W_i &= M_i, \quad 0 \leq i \leq 15, \\ W_i &= (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1, \quad 16 \leq i < 80. \end{aligned} \tag{5}$$

Second, the chaining value H^l is loaded into the registers (A, B, C, D, E) and updated through 80 steps according to Fig. 2. At each step, an expanded message

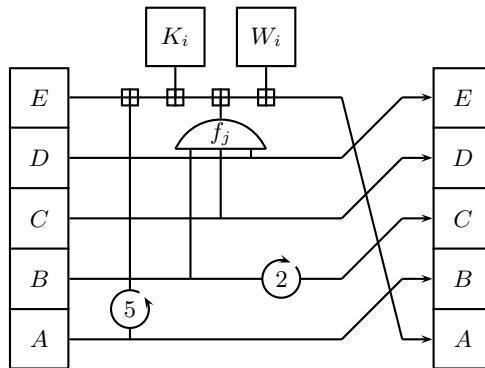


Fig. 2. The step transformation of SHA-1

word W_i , a bitwise boolean function f_i , and a constant K_i are used. The final content of the registers is the output of F .

3.2 One-Block Preimages

Given H , we want to find M such that $H = F(M, \text{IV}) + \text{IV}$. The attacks work as described in Section 2, using probabilistic and truncated differentials, but not using bicliques and the splice and cut technique.

Finding Suitable Attack Parameters. An attack on N steps requires the following parameters: a separation $F = F_2 \circ F_1$ into n_1 and $n_2 = N - n_1$ steps, two linear spaces $D_1, D_2 \subset \{0, 1\}^\kappa$ of dimension d , output differences Δ_1 (resp. Δ_2) for all differences $\delta_1 \in D_1$ (resp. $\delta_2 \in D_2$), and a truncation mask $T \in \{0, 1\}^n$ of Hamming weight r . The message expansion of SHA-1 is linear and we can use the techniques from Section 2.5. For $0 \leq k < 16$, the kernel of any k consecutive steps has dimension $(16 - k) \cdot 32$. Thus, an attack on $2 \cdot 15 = 30$ steps is possible without advanced matching techniques. When attacking more steps, $(n_1 - 15) : (n_2 - 15) \approx 1 : 3$ turned out to be a good ratio, because the diffusion is weaker in the backward direction than in the forward direction.

Our main tool for finding attack parameters are two algorithms which allow to experimentally evaluate candidate configurations. Given n_1, n_2, D_1 , and D_2 , the output differences are computed by linearization: $\Delta_1 = \overline{F}_1(\delta_1, 0)$ and $\Delta_2 = \overline{F}_2^{-1}(\delta_2, 0)$, where \overline{F}_1 and \overline{F}_2 are obtained from F_1 and F_2 by replacing all non-linear Boolean functions f_i by $(X, Y, Z) \mapsto X \oplus Y \oplus Z$, replacing $+$ by \oplus , and setting the constants to zero. Then, Algorithm 2 is used to determine a truncation mask T based on a ranking of bitwise difference probabilities, and finally, Algorithm 3 estimates the corresponding type I error probability $\overline{\alpha}$. The expected attack complexity is computed as in Section 2.2 with $\Gamma_{re} = (N - 30)/N$ (the first 15 and the last 15 steps don't have to be recomputed when retesting a candidate preimage).

There is a trade-off between d and $\overline{\alpha}$. While this trade-off is hard to analyze by hand, it can be readily explored by our experimental approach. Note that Algorithm 2 always returns a mask of weight $r = d$. We did not find significantly better attacks for different choices of r . Table 2 summarizes the results for different N . The results have been found by extensive experiments using $Q = 2^{16}$ in both algorithms. The full attack parameters for $N = 57$ are given below, for the other attacks they are given in an extended version of this paper.

Algorithm 2. Find truncation mask T for matching

Input: $D_1, D_2 \subset \{0, 1\}^\kappa$

Output: A truncation mask $T \in \{0, 1\}^n$ of Hamming weight d .

c = an array of n counters set to 0

for $q = 1$ to Q **do**

 Choose $M \in \{0, 1\}^\kappa$ at random

$C \leftarrow F(M, IV)$

 Choose $(\delta_1, \delta_2) \in D_1 \times D_2$ at random

$\nabla \leftarrow F_1(M \oplus \delta_1, IV) \oplus \Delta_1 \oplus F_2^{-1}(M \oplus \delta_2, C) \oplus \Delta_2$

for $i = 0$ to $n - 1$ **do**

if the i -th bit of ∇ is 1 **then**

$c[i] \leftarrow c[i] + 1$

end if

end for

end for

Set those d bits of T to 1 which have the lowest counters.

Algorithm 3. Estimate type I error probability

Input: $D_1, D_2 \subset \{0, 1\}^\kappa$, $T \in \{0, 1\}^n$
Output: Average type I error probability $\bar{\alpha}$.

```

 $c =$  a counter set to 0
for  $q = 1$  to  $Q$  do
    Choose  $M \in \{0, 1\}^\kappa$  at random
     $C \leftarrow F(M, IV)$ 
    Choose  $(\delta_1, \delta_2) \in D_1 \times D_2$  at random
     $\nabla \leftarrow F_1(M \oplus \delta_1, IV) \oplus \Delta_1 \oplus F_2^{-1}(M \oplus \delta_2, C) \oplus \Delta_2$ 
    if  $\nabla \neq_T 0^n$  then
         $c \leftarrow c + 1$  // a false rejection of  $H_0$ 
    end if
end for
return  $c/Q$ 
```

Dealing with the Padding. If M is required to have correct padding, the least significant bit of M_{13} must be 1, $M_{14} = 0$, and $M_{15} = 447$ (assuming a message length of 447 bits). The differences in D_1 and D_2 must be zero on the corresponding bits. This imposes 65 bit conditions which restrict our choice of D_1 and D_2 . In general, D_1 and D_2 can have only 13 steps without differences (instead of 15). Experiments show that we loose about 5 steps. One step can be attributed to the fact that differences in D_2 tend to have higher Hamming weight.

Attack Parameters for $N = 57$ (one-block, no padding). The best result was obtained for $n_1 = 21$, $n_2 = 36$, and $d = 3$, using the subspaces D_1 and D_2 and the truncation mask T given below.

Basis of D_1 :

```

0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000010
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000020
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000040
```

Basis of D_2 :

```

0x80000000 0x80000000 0x80000000 0x00000000 0x00000000 0x40000000
0x00000000 0xA0000000 0x80000000 0xA0000000 0x00000000 0x80000000
0x00000000 0xC0000000 0x00000000 0x80000000
0x00000001 0x00000001 0x00000001 0x00000000 0x00000000 0x80000000
0x00000000 0x40000001 0x00000001 0x40000001 0x00000000 0x00000001
0x00000000 0x80000001 0x00000000 0x00000001
```

```
0x00000002 0x00000002 0x00000002 0x00000000 0x00000000 0x00000001
0x00000000 0x80000002 0x00000002 0x80000002 0x00000000 0x00000002
0x00000000 0x00000003 0x00000000 0x00000002
```

Truncation mask T (leftmost word is register A):

```
0x00000007 0x00000000 0x00000000 0x00000000 0x00000000
```

The average type I error probability is estimated as $\bar{\alpha} = 0.541$ by Algorithm 3 which results in an expected attack complexity of $2^{158.68}$ evaluations of the compression function.

Table 2. One-block preimages: N is the number of attacked steps, n_1 and n_2 the number of steps computed by F_1 and F_2 , respectively, d the dimension of D_1 and D_2 , and $\bar{\alpha}$ the average type I error probability for the filtering of candidate preimages

N	n_1	n_2	d	$\bar{\alpha}$	Complexity	Remark
44	17	27	15	0.428	$2^{146.21}$	no padding
48	18	30	11	0.552	$2^{150.62}$	
49	18	31	10	0.593	$2^{151.78}$	
50	18	32	10	0.811	$2^{152.89}$	
51	18	33	9	0.842	$2^{154.16}$	
52	20	32	7	0.631	$2^{154.95}$	
53	20	33	6	0.577	$2^{155.76}$	
54	20	34	5	0.547	$2^{156.67}$	
55	21	34	5	0.699	$2^{157.27}$	
56	21	35	4	0.620	$2^{157.94}$	
57	21	36	3	0.541	$2^{158.68}$	
44	16	28	10	0.546	$2^{151.54}$	with padding
48	17	31	7	0.484	$2^{154.41}$	
50	18	32	5	0.598	$2^{156.80}$	
51	18	33	4	0.590	$2^{157.78}$	
52	18	34	4	0.738	$2^{158.44}$	

3.3 One-Block Pseudo-preimages

Given H , we want to find M and H' such that $H = F(M, H') + H'$. The freedom of choosing H' allows us to use bicliques and the splice and cut technique.

We separate F into three parts as shown in Fig. 3. The bicliques are constructed for F_3 computing the steps $27 - n_3$ to 26 (n_3 steps). F_1 computes the steps 27 to $26 + n_1$ (n_1 steps) and F_2 computes the steps $27 + n_1$ to $N - 1$ and 0 to $26 - n_3$ (n_2 steps) using the splice and cut technique. With this choice, the elements of the kernel of the steps 27 to 41 (15 steps) and the elements of the kernel of the steps $11 - n_3$ to $26 - n_3$ (15 steps) automatically satisfy the padding conditions if $0 \leq n_3 \leq 11$.

Except for the bicliques, finding attack parameters is very similar to the case of one-block preimages. The bicliques for all attacks have been found by simple

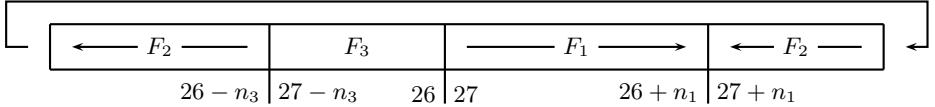


Fig. 3. Separation of F for pseudo-preimage attacks. Bicliques are constructed for F_3 .

trial and error search and because they are shorter than 16 steps, many bicliques can be generated from a single one by just modifying some message words outside the biclique. As a result, the amortized cost to construct bicliques is negligible and the complexity computes as $2^{n-d}(\Gamma_1 + \Gamma_2) + 2^{n-d}\Gamma_{re}$, where $\Gamma_1 + \Gamma_2 = (n_1 + n_2)/N$ and $\Gamma_{re} = (N - n_3 - 30)/N$. Table 3 summarizes the results and the full parameters are given for $N = 57$.

Attack Parameters for $N = 57$ (pseudo-preimage, with padding). The best result was obtained for $n_1 = 18$, $n_2 = 33$, $n_3 = 6$, and $d = 7$, using the subspaces D_1 and D_2 and the truncation mask T given below. A sample biclique is specified by $Q_1[0]$ and a message M . The remaining states of the biclique can be computed by using the defining property of bicliques given in Section 2.4. Note that M has a correct padding which is preserved by the differences in D_1 and D_2 .

Basis of D_1 :

```

0x00020000 0x00020000 0x00010000 0x00000000 0x00020000 0x00020000
0x00010000 0x00000000 0x00000000 0x00020000 0x00030000 0x00000000
0x00020000 0x00020000 0x00000000 0x00000000

0x00040000 0x00040000 0x00020000 0x00000000 0x00040000 0x00040000
0x00020000 0x00000000 0x00000000 0x00040000 0x00060000 0x00000000
0x00040000 0x00040000 0x00000000 0x00000000

0x00080000 0x00080000 0x00040000 0x00000000 0x00080000 0x00080000
0x00040000 0x00000000 0x00000000 0x00080000 0x000C0000 0x00000000
0x00080000 0x00080000 0x00000000 0x00000000

0x00100000 0x00100000 0x00080000 0x00000000 0x00100000 0x00100000
0x00080000 0x00000000 0x00000000 0x00100000 0x00180000 0x00000000
0x00100000 0x00100000 0x00000000 0x00000000

0x00200000 0x00200000 0x00100000 0x00000000 0x00200000 0x00200000
0x00100000 0x00000000 0x00000000 0x00200000 0x00300000 0x00000000
0x00200000 0x00200000 0x00000000 0x00000000

0x00400000 0x00400000 0x00200000 0x00000000 0x00400000 0x00400000
0x00200000 0x00000000 0x00000000 0x00400000 0x00600000 0x00000000
0x00400000 0x00400000 0x00000000 0x00000000

0x00800000 0x00800000 0x00400000 0x00000000 0x00800000 0x00800000
0x00400000 0x00000000 0x00000000 0x00800000 0x00C00000 0x00000000
0x00800000 0x00800000 0x00000000 0x00000000

```

Basis of D_2 :

```

0x00000000 0x80000000 0x00000000 0x80000000 0x00000000 0x80000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000

0x00000000 0x00000001 0x00000000 0x00000001 0x00000000 0x00000001
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000

0x00000000 0x00000002 0x00000000 0x00000002 0x00000000 0x00000002
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000

0x00000000 0x00000004 0x00000000 0x00000004 0x00000000 0x00000004
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000

0x00000000 0x00000008 0x00000000 0x00000008 0x00000000 0x00000008
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000

0x00000000 0x00000010 0x00000000 0x00000010 0x00000000 0x00000010
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000

0x00000000 0x00000020 0x00000000 0x00000020 0x00000000 0x00000020
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000

```

Truncation mask T (leftmost word is register A):

```
0x00000003 0x00000000 0x80000003 0x00000003 0x00000000
```

$Q_1[0]$ for sample biclique:

```
0x1c2652fe 0x53eb4c0a 0x57e9168f 0xf65b3a56 0x7c428e01
```

M for sample biclique:

```
0xce369809 0x3ea1797b 0x1ab39a0d 0x96d1d5e0 0x7a550f31 0xad4da4dd
0x0f72712f 0x17d8a5e8 0xdad6d21d 0x3b0faf80 0x7cc259ff 0xb27a9d25
0x22a02a94 0x88bbfd35 0x00000000 0x000003bf
```

The average type I error probability is estimated as $\overline{\alpha} = 0.676$ which results in an expected attack complexity of $2^{154.96}$ evaluations of the compression function.

3.4 Two-Block Preimages

Given H , we want to find $M = (M^0, M^1)$ with a correct padding and such that $H = F(M^1, H^1) + H^1$, where $H^1 = F(M^0, \text{IV}) + \text{IV}$. The problem can be separated into two steps:

1. Find M^1 such that $H = F(H^1, M^1) + H^1$ for some H^1 and such that M^1 has a correct padding (a “one-block pseudo-preimage with padding”).

Table 3. One-block pseudo-preimages: N is the number of attacked steps, n_1 and n_2 the number of steps computed by F_1 and F_2 , respectively, n_3 is the length of the bicliques, d the dimension of D_1 and D_2 , and $\overline{\alpha}$ the average type I error probability for the filtering of the candidate preimages

N	n_1	n_2	n_3	d	$\overline{\alpha}$	Complexity	Remark
48	17	29	2	12	0.447	$2^{149.22}$	pseudo-preimage, with padding
49	18	29	2	11	0.398	$2^{150.12}$	
50	18	30	2	10	0.404	$2^{151.15}$	
51	16	31	4	9	0.381	$2^{152.02}$	
52	17	31	4	8	0.332	$2^{152.93}$	
53	17	30	6	7	0.128	$2^{153.46}$	
54	17	31	6	8	0.569	$2^{153.50}$	
55	17	31	7	6	0.208	$2^{154.60}$	
56	18	32	6	8	0.765	$2^{154.41}$	
57	18	33	6	7	0.675	$2^{154.96}$	
58	20	31	7	5	0.478	$2^{156.25}$	
59	18	35	6	5	0.626	$2^{156.78}$	
60	19	35	6	4	0.524	$2^{157.45}$	

2. For the H^1 obtained in the first step, find M_0 such that $H^1 = F(M^0, \text{IV}) + \text{IV}$ (a “one-block preimage without padding”).

The two steps can be solved using the attacks from Section 3.2 and 3.3, respectively. The total complexity is the sum of both steps, which is dominated by the second step (computing the first block). As an example, for $N = 48$ we can compute a correctly padded two-block preimage with complexity $2^{150.62} + 2^{149.22} = 2^{151.08}$. For $N = 57$ the complexity is $2^{158.79}$.

For $N \geq 58$ we lack a method to compute the first block faster than brute-force and we would have to use the generic method from [14, Fact 9.99] to convert a pseudo-preimage attack into a preimage attack. In fact, this is the procedure of most meet-in-the-middle preimage attacks. A pseudo-preimage attack with complexity 2^m can be converted to a preimage attack with complexity $2^{1+(m+n)/2}$. In our case, the resulting speed-up is less than a factor two for all results with $N \geq 58$ given in Table 3.

4 Accelerated Brute-Force Search

In this last section we briefly describe a generic optimization of the brute-force search which comes out of the meet-in-the-middle approach. It applies to any number of rounds, but the speed-up is very small. The idea is to not recompute parts of F if they are identical for several messages. This has been previously applied to MD5 [2] and HAVAL-5 [17], and the same idea underlies “matching with precomputation” used for key recovery attacks on AES [4].

Suppose that F can be separated into three parts, $F = F_2 \circ F_3 \circ F_1$, such that D_1 and D_2 can be found as in Section 2.1 for F_1 and F_2 , but with zero

output differences. Then, Algorithm 4 can be used to test a set $M \oplus D_1 \oplus D_2$. The additional cost compared to Algorithm 1 comes from the 2^{2d} computations of F_3 in the last loop. Testing 2^n messages has complexity $2^{n-d}(\Gamma_1 + \Gamma_2) + 2^n \Gamma_3$. Thus, for reasonably large d , the complexity of brute-force search is essentially reduced to 2^n evaluations of F_3 instead of 2^n evaluations of F .

Algorithm 4. Accelerated brute-force search

Input: $D_1, D_2 \subset \{0, 1\}^\kappa, M \in \{0, 1\}^\kappa$

Output: A preimage if one is contained in $M \oplus D_1 \oplus D_2$.

for all $\delta_2 \in D_2$ do

 Compute $L_1[\delta_2] = F_1(M \oplus \delta_2, P)$.

end for

for all $\delta_1 \in D_1$ do

 Compute $L_2[\delta_1] = F_2^{-1}(M \oplus \delta_1, C)$.

end for

for all $(\delta_1, \delta_2) \in D_1 \times D_2$ do

 if $F_3(M \oplus \delta_1 \oplus \delta_2, L_1[\delta_2]) = L_2[\delta_1]$ then

 return $M \oplus \delta_1 \oplus \delta_2$

 end if

end for

return No preimage in $M \oplus D_1 \oplus D_2$

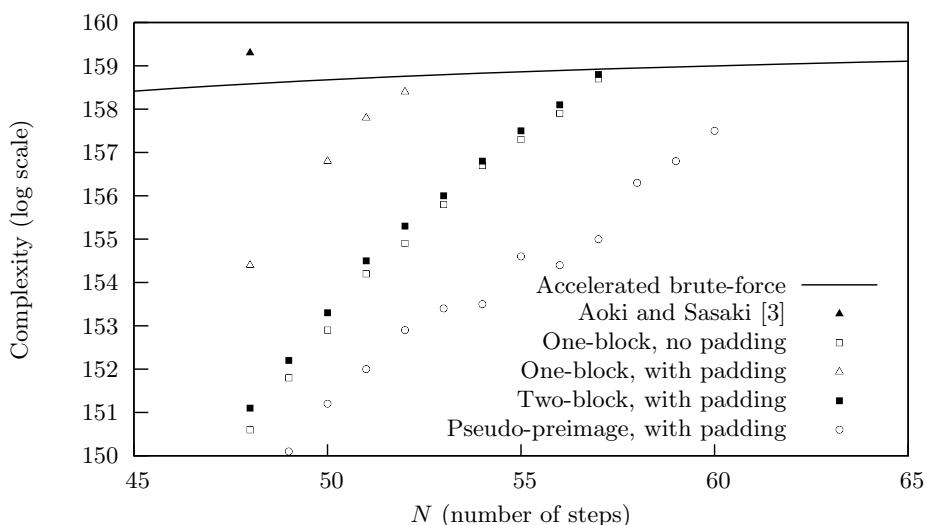


Fig. 4. Preimage attacks against reduced SHA-1: Illustration of the new results and comparison to accelerated brute-force search

Application to SHA-1. The speed-up factor is about $N/(N - 30)$ for a variant with N steps. Slight improvements are possible by using probabilistic and truncated differentials for F_2^{-1} . For the full SHA-1, a speed-up factor of about two can be obtained. Such an optimization might not be considered as an attack, but it provides a minimal benchmark for actual attacks. Figure 4 compares our results to this benchmark.

5 Summary and Conclusion

We proposed a differential view on the meet-in-the-middle framework originally introduced by Aoki and Sasaki. Advanced matching techniques such as partial matching, indirect partial matching, partial fixing, and probabilistic matching appear very natural from this perspective. For block cipher based hash functions in Davies-Meyer mode, the principal attack parameters are two sets of suitable related-key differentials. Tools are proposed that facilitate a systematic search for these sets.

Applied to SHA-1, our framework leads to significantly better preimage attacks up to 57 out of 80 steps. The results are illustrated in Fig. 4 and compared to the best previous attack as well as to accelerated brute-force search. The improvements essentially come from a more systematic use of probabilistic matching. It is remarkable that we do not rely on the generic conversion of pseudo-preimage attacks into preimage attacks. This allows us to obtain speed-up factors that would be hard to achieve with the generic conversion.

Application of the framework to the SHA-2 family seems more complicated, namely due to the non-linear message expansion. Nevertheless, it is expected that the differential perspective on meet-in-the-middle attacks leads to improved results on other primitives as well.

Acknowledgements. We thank Christian Rechberger for interesting discussions on preimage attacks and SHA-1. This work was partially supported by the Hasler Foundation www.haslerfoundation.ch under project number 08065.

References

1. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for Step-Reduced SHA-2. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 578–597. Springer, Heidelberg (2009)
2. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
3. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
4. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)

5. De Cannière, C., Rechberger, C.: Preimages for Reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008)
6. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
7. Chaum, D., Evertse, J.-H.: Cryptanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 192–211. Springer, Heidelberg (1986)
8. Diffie, W., Hellman, M.: Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10, 74–84 (1977)
9. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010)
10. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: Canteaut, A. (ed.) FSE 2012. LNCS. Springer (to appear, 2012)
11. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
12. Leurent, G.: MD4 is Not One-Way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
13. Mendel, F., Pramstaller, N., Rechberger, C., Kontak, M., Szmidt, J.: Cryptanalysis of the GOST Hash Function. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 162–178. Springer, Heidelberg (2008)
14. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
15. National Institute of Standards and Technology: FIPS 180-3: Secure Hash Standard (2008), <http://www.itl.nist.gov/fipspubs/>
16. Sasaki, Y., Aoki, K.: A Preimage Attack for 52-Step HAS-160. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 302–317. Springer, Heidelberg (2009)
17. Sasaki, Y., Aoki, K.: Preimage Attacks on 3, 4, and 5-Pass HAVAL. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
18. Sasaki, Y., Aoki, K.: Preimage Attacks on Step-Reduced MD5. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 282–296. Springer, Heidelberg (2008)
19. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
20. Wang, L., Sasaki, Y.: Finding Preimages of Tiger Up to 23 Steps. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 116–133. Springer, Heidelberg (2010)
21. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

Stam’s Conjecture and Threshold Phenomena in Collision Resistance

John Steinberger^{1,*}, Xiaoming Sun^{2,**}, and Zhe Yang³

¹ Institute of Theoretical Computer Science, Tsinghua University, Beijing

² Institute of Computing Technology, China Academy of Sciences

³ Hulu Software, Beijing

{jpsteinb,xiaoming.sun,yangzhe1990}@gmail.com

Abstract. At CRYPTO 2008 Stam [8] conjectured that if an $(m+s)$ -bit to s -bit compression function F makes r calls to a primitive f of n -bit input, then a collision for F can be obtained (with high probability) using $r2^{(nr-m)/(r+1)}$ queries to f , which is sometimes less than the birthday bound. Steinberger [9] proved Stam’s conjecture up to a constant multiplicative factor for most cases in which $r = 1$ and for certain other cases that reduce to the case $r = 1$. In this paper we prove the general case of Stam’s conjecture (also up to a constant multiplicative factor). Our result is qualitatively different from Steinberger’s, moreover, as we show the following novel threshold phenomenon: that exponentially many (more exactly, $2^{s-2(m-n)/(r+1)}$) collisions are obtained with high probability after $O(1)r2^{(nr-m)/(r+1)}$ queries. This in particular shows that threshold phenomena observed in practical compression functions such as JH are, in fact, unavoidable for compression functions with those parameters.

1 Introduction

The ideal primitive model (IPM) is a popular paradigm in cryptographic security proofs. In this model one assumes that some primitive used by a construction, such as a blockcipher, is “ideal”—namely perfectly random subject to the constraints of the type of primitive under consideration—and one then bounds the chance of success of an adversary given oracle access to the ideal primitive, in some given security experiment, for some given number of queries. The adversary considered is almost always information-theoretic. As such, the adversary’s only obstacle to achieving its attack is the randomness of the query responses.

Because the IPM considers information-theoretic adversaries certain limitations naturally arise as to what kind of security can be achieved for a certain

* Supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174, and by NSF grant 0994380.

** Supported by the National Natural Science Foundation of China Grant 61170062, 61061130540, and the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301.

functionality using a certain primitive a certain number of times. For example, consider the task of constructing a $2n$ -bit to n -bit compression function F using a random n -bit to n -bit permutation f as a primitive. There are 2^{2n} inputs to F but only 2^n inputs to f . Thus each input to f corresponds on average to 2^n inputs to F , so with just two calls to f we can learn to evaluate F on at least $2 \cdot 2^n$ inputs. But this is more than the number of outputs of F , so a collision can be obtained with probability 1 in just two queries. Note that determining which two f -queries to make is no problem for an information-theoretic adversary, nor is “finding the collision” among the $2 \cdot 2^n$ mapped values. Thus it is not possible to design a compression function with these parameters that is collision resistant in the IPM.

This paper follows a line of work [2, 6, 8, 9] in the same vein as the above argument, seeking to establish the limits of provable security in the IPM model. Specifically, we focus the following question related to work of Stam [8] and, before that, of Rogaway and Steinberger [6]: given $m, n, r, s \geq 1$, what is the maximum collision security of a compression function $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^s$ that makes r calls to an ideal primitive f of domain $\{0, 1\}^n$? (The range of f is not specified because it turns out to be immaterial¹.) Here “collision security” means the largest number of f -queries the best information-theoretic adversary can ask before achieving probability $\frac{1}{2}$ of obtaining a collision.

Since it costs at most r queries to evaluate any point in the domain, a birthday attack implies that collision security cannot exceed $q = O(1)r2^{s/2}$ queries. However, depending on the parameters, other attacks may be more effective than birthday attacks. In particular Stam [8] conjectured that

$$q = r \lceil 2^{(nr-m)/(r+1)} \rceil + 1 \quad (1)$$

queries should always suffice for finding a collision with probability at least $\frac{1}{2}$. (We restate Stam's conjecture as slightly modified by Steinberger [9].) Roughly speaking, this bound is less than a birthday attack when $s/2 > (nr - m)/(r + 1)$. The latter occurs for example when $(m, n, r, s) = (n, n, 2, n)$, the case of a $2n$ -bit to n -bit compression function making two calls to a primitive of n -bit input, for which Stam's bound forecasts a maximum collision resistance of $2^{n/3}$, which is more restrictive than the birthday bound of $2^{n/2}$. As a second example, Stam's bound is even more restrictive when $(m, n, r, s) = (n, n, 1, n)$, for which it forecasts a maximum collision resistance of $O(1)$ queries; in fact this setting of parameters coincides with the first example discussed in the paper (regarding a compression function $F : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ making a single call to an n -bit random permutation).

Stam's conjecture is appealing because it apparently constitutes the *optimal* upper bound on collision resistance for all cases for which it beats the birthday bound, while the birthday bound can apparently be achieved in all other cases. In other words, as far as currently understood, it seems like the maximum collision

¹ Immaterial to proving the upper bound under consideration in this paper; better upper bounds on security should be provable if f has sufficiently small range, see comments by Stam [8].

resistance of a compression function $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^s$ making r calls to a random function f of n -bit input equals

$$\min(r2^{s/2}, r\lceil 2^{(nr-m)/(r+1)} \rceil)$$

up to possible lower order terms. This thesis is supported by a number of constructions [5, 7, 8].

Steinberger [9] obtained the only previous results on Stam's conjecture. He proved that when

$$(2m - n(r - 1))/(r + 1) \geq 4.09 \quad (2)$$

$O(1)r\lceil 2^{(nr-m)/(r+1)} \rceil$ queries suffice to find a collision for F with probability at least 0.5. The condition (2) is increasingly restrictive as r grows; for $r = 1$, it reduces to $m \geq 4.09$; for $r = 2$, it reduces to $\frac{2}{3}m - \frac{1}{3}n \geq 4.09$; for $r = 3$ it reduces to $\frac{1}{2}m - \frac{1}{2}n \geq 4.09$; and so on. If $m = n$ (a typical case in real-world constructions) then (2) is false for all $r \geq 3$. Thus Stam's conjecture was until now, and despite Steinberger's result, very much open in the general case.

Steinberger also made the observation that for certain parameters (m, n, r, s) Stam's conjecture can be reduced to parameters (m', n, r', s) such that $m' < m$ and $r' < r$. (To be precise, such a reduction can be effected whenever $mr \geq n$.) In fact, the core of Steinberger's result is a proof of Stam's conjecture for the case² $r = 1$ and $m \geq 4.09$. Other parameters (m, n, r, s) with $r > 1$ to which Steinberger's result applies are precisely those for which the reduced tuple (m', n, r', s) has $r' = 1$ and $m' \geq 4.09$ (inequality (2) is sufficient and necessary for both $r' = 1$ and $m' \geq 4.09$ to hold). Thus Steinberger's result is "really" about the case $r = 1$ of Stam's conjecture. (To be fair, the results of [9] nonetheless cover a large number of parameter settings of practical interest.)

In this paper we resolve the general case of Stam's conjecture. More precisely we show that if $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^s$ is a compression function using r calls to a primitive f of n -bit input, where $m \geq 1$, then with high probability a collision can be found for F in at most

$$O(1)r \left\lceil 2^{\frac{nr-m}{r+1}} \right\rceil$$

queries to f , where the $O(1)$ term represents a constant independent of all other parameters. This constant, being in the vicinity of 16000, is large but not astronomical. Note that *some* lower bound must be imposed on m , since if $m = 0$ the domain has the same size as the range, and F may have no collisions at all (e.g., F may ignore its primitive f , and be the identity on $\{0, 1\}^s$). In fact, Stam's conjecture doesn't hold under the sole assumption $m > 0$, as is easy to see. For example, this would allow the domain to have a single more point than the range, making a collision very hard to find³.

² When m is not an integer, our meaning is that F is a compression function of domain of size at least $\lceil 2^{s+m} \rceil$; the qualitative nature of the domain (be it bitstrings, or some other set) is not relevant. See Section 2 for a more precise statement.

³ See Wiener [10] for details on the effectiveness of birthday attacks in functions where the size of the domain approaches the size of the range.

At this point we emphasize that, like for Steinberger's theorem, the “primitive” f called by the compression function F can be *any* type of primitive of n -bit input, i.e., can be drawn from *any* distribution. Such a primitive can model, for example, a n -bit to n -bit permutation, but it can also model, say, a blockcipher⁴, or essentially any type of function-like primitive.

On the other hand our result is qualitatively different from Steinberger's in that we show an interesting threshold phenomenon: for the range of parameters in which Stam's bound is less than the cost of a birthday attack (namely, when $s/2 > (nr - m)/(r + 1)$), we show *many* collisions are obtained with high probability as soon as $O(1)r\lceil 2^{\frac{nr-m}{r+1}} \rceil$ queries are made; more precisely, one obtains at least

$$2^{s - \frac{2(nr-m)}{r+1}} \quad (3)$$

collisions with high probability, using at most $16000r\lceil 2^{\frac{nr-m}{r+1}} \rceil$ queries to f . In this regard, it is worth recalling that Stam's bound is conjecturally optimal (and for some settings of parameters provably optimal), implying that with only

$$o(1)r\lceil 2^{\frac{nr-m}{r+1}} \rceil$$

queries to f , *no* collisions for F are found with high probability (presuming an adequate F). Note the exponent $s - 2(nr - m)/(r + 1)$ in (3) is precisely twice the difference between the exponent in the cost of a birthday attack (a.k.a. $s/2$) and the exponent of Stam's conjecture (a.k.a. $(nr - m)/(r + 1)$). Thus, the further Stam's bound is beneath the cost of a birthday attack, the sharper the threshold phenomenon.

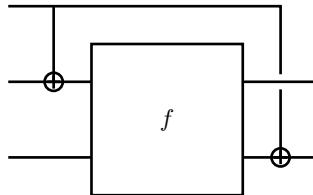


Fig. 1. The JH compression function $G : \{0, 1\}^{1.5n} \rightarrow \{0, 1\}^n$. All wires carry $n/2$ -bit values.

As an example, we can consider the compression function G of JH [11], one of the finalists in NIST's SHA-3 competition, pictured in Figure 1. This is a compression function from $\{0, 1\}^{1.5n}$ to $\{0, 1\}^n$ using a single call to a primitive f of n -bit input (more precisely, f is a permutation). Thus the JH compression

⁴ A blockcipher with m -bit word and k -bit key can be modeled as a primitive of input length $n = m + k$, or of input length $n = m + k + 1$ if the construction also uses “inverse” blockcipher calls (in which case the extra bit indicates whether the call is forward or backward).

function G has parameters $(m, n, r, s) = (0.5n, n, 1, n)$. The cost of a birthday attack for G is $2^{n/2}$. Stam's bound, however, is

$$2^{\frac{nr-m}{r+1}} = 2^{\frac{n-0.5n}{2}} = 2^{n/4}.$$

Thus Stam's conjecture indicates that the JH compression function G must have significantly weaker than birthday collision resistance. It is indeed easy to see that, on average, only $2^{n/4}$ queries are required to find a collision for G , since all one needs is to find a collision on the top half of output (the bottom half can then be adjusted via the input wire). Steinberger's theorem (and our own result as well) shows that *any* compression function with parameters $(m, n, r, s) = (0.5n, n, 1, n)$, regardless of its design, will likewise have collision resistance at most $2^{n/4}$ (up to a small constant factor). Observe, also, that once a single collision is obtained for G , $2^{n/2}$ collisions are obtained at once, since we can replicate the collision with any value on the bottom output wire. Our own theorem, beyond showing that collision resistance cannot exceed $2^{n/4}$, predicts this threshold behavior as well. More precisely, we show that for *any* compression function with parameters $(m, n, r, s) = (0.5n, n, 1, n)$, an adversary making at most⁵ $200 \cdot 2^{n/4}$ queries to f can obtain

$$2^{s - \frac{2(nr-m)}{r+1}} = 2^{n - \frac{2(n-0.5n)}{2}} = 2^{0.5n}$$

collisions⁶ with high probability. On the other hand, we emphasize that *no* collisions are obtained for G with $\frac{1}{10} \cdot 2^{n/4}$ queries (for any adversary, w.h.p.). Thus we not only pinpoint the collision resistance up to a constant factor, but we also pinpoint the exact “payoff” that occurs once collision resistance is breached.

As a second example, in [8] Stam exhibits a compression function with parameters $(m, n, r, s) = (n, n, 2, n)$ of collision resistance $2^{n/3}$ (Stam's bound). Stam's compression function has the particularity that $n/3$ bits are simply forwarded untouched from the input to the output, whereas the remaining $2n - n/3 = \frac{5}{3}n$ input bits are cryptographically processed into the remaining $n - n/3 = \frac{2}{3}n$ output bits. Obviously, for such a compression function, $2^{n/3}$ collisions are obtained once a single collision is obtained. Our own result shows this sudden jump (from no collisions to $2^{n/3}$ collisions) is essentially unavoidable, in the sense that with

$$16000 \cdot 2^{\frac{nr-m}{r+1}} = 16000 \cdot 2^{\frac{2n-n}{3}} = 16000 \cdot 2^{n/3}$$

queries an adversary can obtain

$$2^{s - \frac{2(nr-m)}{r+1}} = 2^{n - \frac{2(2n-n)}{3}} = 2^{n/3}$$

⁵ When $r = 1$ the multiplicative constant can be improved from 16000 to 200. See Section 3 for more details.

⁶ Traditionally, the “number of collisions” means the “number of distinct pairs of inputs that collide”. Note, however, that under this definition $2^{0.5n}$ “collisions” may be caused by only $2^{0.25n}$ inputs, all involved in one big multi-collision. We show, in fact, that the *number of different inputs involved in a collision* is at least $2^{0.5n}$, which constitutes an even stronger result.

collisions with high probability, and this for *any* compression function with parameters $(m, n, r, s) = (n, n, 2, n)$.

ORGANIZATION. Section 2 contains relevant definitions and conventions. Section 3 states and briefly discusses our main result. Section 4 gives an overview of the proof, and briefly compares our proof techniques to those of Steinberger [9]. The actual proof of our main theorem (this being Theorem 2 in Section 3) is left to the full version of this paper for reasons of space, but the key technical lemmas, which contain the more mathematically interesting techniques and on which the overview of Section 4 is also based, are proved in Appendix A.

2 Definitions and Preliminaries

COMPRESSION FUNCTIONS. Let $m \geq 0$ be a real number and let $s \geq 0$ be an integer. (Our results hold as stated even when $s \geq 0$ is a real number such that 2^s is an integer and, likewise, also when $n \geq 0$ is a real number such that 2^n is an integer. However, for notational and conceptual simplicity, we shall assume n, s are integers.) By “a function of domain $\{0, 1\}^{s+m}$ ” we mean a function with a domain of size $\lceil 2^{s+m} \rceil$ —the exact nature of the domain will not matter for our results, but for notational convenience we still write the domain as $\{0, 1\}^{s+m}$ (even though m is not necessarily an integer and, furthermore, even though 2^{s+m} is not necessarily an integer). Readers who feel uneasy about this convention may think of $\{0, 1\}^{s+m}$ as being a shorthand for some fixed subset of $\{0, 1\}^{\lceil s+m \rceil}$ of size $\lceil 2^{s+m} \rceil$.

Let now $m \geq 0$ be a real number and let $r \geq 1, n, s \geq 0$ be integers. We formalize the notion of a compression function $F : \{0, 1\}^{s+m} \rightarrow \{0, 1\}^s$ making r calls to a primitive f of domain $\{0, 1\}^n$.

In fact we allow F to call potentially distinct primitives f_1, \dots, f_r in *fixed order mode*, meaning f_i is called before f_j for $i < j$. Let f_1, \dots, f_r be (not necessarily distinct) functions of domain $\{0, 1\}^n$ and range $\{0, 1\}^b$, where b is arbitrary. The compression function $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^s$ is defined by r functions g_1, \dots, g_r where $g_i : \{0, 1\}^{m+s} \times \{0, 1\}^{b(i-1)} \rightarrow \{0, 1\}^n$ and a function $h : \{0, 1\}^{m+s} \times \{0, 1\}^{br} \rightarrow \{0, 1\}^s$. We then define $F(v) = h(v, y_1, \dots, y_r)$ where $y_j = f_j(g_j(v, y_1, \dots, y_{j-1}))$ for $j = 1 \dots r$. We call the values y_1, \dots, y_r *intermediate chaining variables* and we refer to the functions g_1, \dots, g_r as the *intermediate processing functions*. We note that g_1, \dots, g_r are, for a given construction, fixed finite functions with a public description.

We say an adversary A with oracle access to f_1, \dots, f_r “knows the first k chaining variables” for some input $v \in \{0, 1\}^{m+s}$ when A has made the queries $f_1(g_1(v)) = y_1, f_2(g_2(v, y_1)) = y_2, \dots, f_k(g_k(v, y_1, \dots, y_{k-1})) = y_k$, where $0 \leq k \leq r$. In this case, we also say A “knows the relevant queries to f_1, \dots, f_k ” for v .

When F is as defined above we call F an “ (m, n, r, s) compression function”. By default, the primitives called by such a compression function are always named f_1, \dots, f_r (in order).

COLLISION ACCOUNTING. The following definition is somewhat nonstandard, but central to the paper:

Definition 1. Let $F : D \rightarrow R$ be a function of domain D and range R . Let $S \subseteq D$. The set of colliding inputs in S (with respect to F) is the set

$$\{x \in S : \exists y \in S, y \neq x, \text{ s.t. } F(x) = F(y)\}.$$

Let F be an (m, n, r, s) compression function calling primitives f_1, \dots, f_r . Let A be an adversary with oracle access to f_1, \dots, f_r . The set of inputs *learned* by A is the set of inputs $S \subseteq \{0, 1\}^{s+m}$ for which A has made the relevant queries to f_1, \dots, f_r at the end of its attack (and therefore, for which A knows the value of F). The set of *colliding inputs obtained by A* is the set $C \subseteq S$ of colliding inputs in S , with respect to F . We say A obtains z colliding inputs if $|C| \geq z$.

It is worth noting that $|C| \geq |S| - |R|$, given that only $|R|$ elements of S can occupy their “own” slots in the range. Thus an adversary that learns $|S|$ inputs for a compression function of range size $|R|$ automatically obtains at least $|S| - |R|$ colliding inputs.

YIELD. The following basic observation is due to Rogaway and Steinberger [6]:

Lemma 1. Let $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^s$ be a compression function calling primitives $f_1, \dots, f_r : \{0, 1\}^n \rightarrow \{0, 1\}^b$ in fixed-order mode. Then there exists an adversary that with at most q queries to each f_i can learn the first i intermediate chaining variables for at least

$$2^{m+s} \left(\frac{q}{2^n}\right)^i$$

inputs, for $0 \leq i \leq r$.

In other words, there exists an adversary making at most q queries to each f_i , and for which

$$|S_i| \geq 2^{m+s} \left(\frac{q}{2^n}\right)^i$$

for $0 \leq i \leq r$, where $S_i \subseteq \{0, 1\}^{m+s}$ is the set of inputs for which the relevant queries to f_1, \dots, f_i have been made. The adversary in question is very straightforward: it is a greedy adversary that starts by choosing its queries to f_1 such as to maximize the size of S_1 , then, after making its queries to f_1 , chooses its queries to f_2 such as to maximize the size of $S_2 \subseteq S_1$, and so on. For a full proof see any of [6], [8] or [9].

Setting $i = r$ in Lemma 1 we obtain the following corollary:

Corollary 1. Let $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^s$ be a compression function calling primitives $f_1, \dots, f_r : \{0, 1\}^n \rightarrow \{0, 1\}^b$ in fixed-order mode. Then with q queries to each f_i , an adversary can learn to evaluate F on at least

$$2^{m+s} \left(\frac{q}{2^n}\right)^r$$

inputs.

3 Results

The following Theorem dispatches the “easy” cases of Stam’s conjecture; similar results are already given in [6, 8, 9].

Theorem 1. (cf. [6, 8, 9]) *Let F be an (m, n, r, s) compression function with $m \geq 1$. Then: (i) if $s/2 \leq (nr - m)/(r + 1)$, a collision can be found for F with at most*

$$q = 2\sqrt{2} \cdot 2^{s/2} + 1 \leq 2\sqrt{2} \cdot 2^{\frac{nr-m}{r+1}} + 1$$

queries to each f_i , with probability at least 0.5; and (ii) if $m \geq nr$, a collision can be found for F with at most 2 queries to each f_i , with probability 1.

Proof. Statement (i) follows by a birthday attack and the fact that $m \geq 1$ (so that the domain of F has size at least twice the range); see [9, 10] for more details. Statement (ii) follows from applying Corollary 1 with $q = 2$, and noting that when $m \geq nr$ we have

$$2^{m+s} \left(\frac{2}{2^n} \right)^r \geq 2 \cdot 2^s$$

so that, automatically, at least $2^{s+1} - 2^s = 2^s$ colliding inputs are obtained by the adversary. \square

In light of Theorem 1, our remaining results are restricted to the case ($s/2 \geq (nr - m)/(r + 1) \wedge m \leq nr$). It is worth noting that $2^{(nr-m)/(r+1)} \geq 1$ when $m \leq nr$, since then $(nr - m)/(r + 1) \geq 0$.

To state and discuss our main result it will be convenient to define the function

$$\gamma(r, c) = 2e^{-c^2/5760} + \sum_{i=1}^{r-1} 2e^{-\frac{1}{32}(\frac{c}{80})^i}$$

where $r \geq 1$ is an integer and $c > 0$ is an arbitrary real number. We keep this definition of $\gamma(r, c)$ for the rest of the paper.

Our main result is the following:

Theorem 2. *Let F be an (m, n, r, s) compression function with $1 \leq m \leq nr$ and $s/2 \geq (nr - m)/(r + 1)$. Let $c > 0$ be a real number such that $c2^{\frac{nr-m}{r+1}}$ is an integer. Then there exists an adversary making at most*

$$q = 2c2^{\frac{nr-m}{r+1}}$$

queries to each f_i and obtaining at least

$$2^{s - \frac{2(nr-m)}{r+1}}$$

colliding inputs, with probability at least $1 - \gamma(r, c)$.

For $r = 1$ one can compute that $\gamma(1, 90) < 0.5$, whereas $\gamma(1, 100) < 0.36$ and $\gamma(1, 1000) < e^{-170}$. Thus $180\lceil 2^{\frac{nr-m}{r+1}} \rceil$ queries suffice to obtain a collision with probability at least 0.5. For $r > 1$, one can make the observation that

$$\gamma(r, c) \leq 2e^{-c^2/5760} + \sum_{i=1}^{\infty} 2e^{-\frac{1}{32}(\frac{c}{80})^i}$$

where the right-hand side does not depend on r , and where the right-hand side is less than 0.5 for $c \geq 8000$. Thus $16000r\lceil 2^{\frac{nr-m}{r+1}} \rceil$ queries (in total to all f_i 's) suffice to find a collision with probability at least 0.5 when $r > 1$.

The proof of Theorem 2 is left to the paper's full version. However the main ideas behind the proof are presented in the next Section, with supporting lemmas in Appendix A.

4 Proof Overview

In this section we give an overview of the proof of Theorem 2. We emphasize that this section's contents constitute *intuition only* and have little mathematical value. An independent, fully self-contained proof of Theorem 2 appears in the paper's full version. Nonetheless, the more technical lemmas needed to implement the ideas described below are proved in this version, in Appendix A.

A central ingredient in our proof is a lemma on collisions (Lemma 5 in Appendix A) that we start by paraphrasing here in order to facilitate the following discussion. Let T_1, \dots, T_k be disjoint sets whose sizes are upper bounded by some constant M , let $F : T \rightarrow R$ be some function where $T = T_1 \cup \dots \cup T_k$, and let C be the total number of colliding inputs in T with respect to F . Note that if we select q of the k sets T_1, \dots, T_k at random, and form a set T' as the union of the q selected sets, then each point of T has probability $p := q/k$ of ending up in T' . Since a colliding input $x_0 \in T$ has probability at least

$$\frac{q}{k} \frac{q-1}{k-1} \approx p^2$$

of winding up as a colliding input in T' (because x_0 must be selected for T' and also at least one of the other points⁷ in T with which x_0 collides must be selected for T'), we can therefore expect T' to have approximately at least

$$p^2 C$$

colliding inputs. Roughly speaking, Lemma 5 states that as long as this expectation is a fair amount larger than M (the maximum size of the T_i 's) then this intuition is borne out, and the number of colliding inputs in T' is not much less than $p^2 C$ with high probability. We point out that Lemma 5 does not “know”

⁷ If such an other point comes from the same set T_i that contains x_0 , this only helps us, in the sense that x_0 then has chance exactly p (the chance that T_i is selected) of becoming a colliding input in T' .

how to take advantage of multi-collisions: if a colliding input $x_0 \in T$ collides with very many other points in T , coming from many different T_j 's, then x_0 's chance of being a colliding input in T' will be significantly greater than p^2 . Thus Lemma 5 does not give a sharp result in all situations. This lack plays a role in the proof sketch below (as well as in the proof itself).

For the proof sketch we start by reviewing some specific settings of the parameters and explain, in each case, how our collision-finding adversary operates, and why it can hope to find the desired number of collisions within the limits of Stam's bound. We call these "case studies". We later abstract more general observations from these case studies. (The first two case studies concern parameter settings that are already covered by Steinberger's [9] results. However, the point is to flesh out our line of attack, which is completely different from Steinberger's birthday-based approach.)

Conceptually we emphasize that, unlike for a typical collision resistance analysis in the provable security setting, it is more useful to view the primitives f_1, \dots, f_r as being sampled (from whatever distribution) *before* the start of the collision resistance experiment, rather than as being lazy sampled. Thus one can think of the primitives f_1, \dots, f_r as functions that are "fixed but arbitrary", and to which the adversary has oracle access.

First Case Study: $(m, n, r, s) = (0.5n, n, 1, n)$. Let $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^n$ be a compression function making a single call to an n -bit primitive f_1 , where $(m, n, r, s) = (0.5n, n, 1, n)$. Thus $F : \{0, 1\}^{1.5n} \rightarrow \{0, 1\}^n$. We note this setting of parameters coincides, for example, with the parameters of the JH compression function (discussed in the introduction).

Stam's bound indicates that

$$q = 2^{\frac{nr-m}{r+1}} = 2^{\frac{n-0.5n}{2}} = 2^{n/4}$$

queries to f_1 should suffice to find collisions for F . Let $S_0 = \{0, 1\}^{m+s}$ be F 's domain, and write

$$S_0 = \bigcup_{y \in \{0, 1\}^n} U_y^0$$

where

$$U_y^0 = \{x \in S_0 : g_1(x) = y\}$$

where g_1 is F 's first and only intermediate processing function (see Section 2). We note that S_0 is the disjoint union of the sets U_y^0 . Moreover, the collision adversary knows each set U_y^0 , since g_1 is public. We also note that the *average size* of the sets $\{U_y^0 : y \in \{0, 1\}^n\}$ is

$$\frac{|S_0|}{2^n} = \frac{2^{m+s}}{2^n} = \frac{2^{1.5n}}{2^n} = 2^{n/2}.$$

For simplicity we start by assuming that $|U_y^0| = 2^{n/2}$ for all $y \in \{0, 1\}^n$. We will discuss later how to lift this assumption.

The adversary's most natural strategy is to make $2^{n/4}$ random queries to f_1 . Let $\mathcal{B} \subseteq 2^{n/4}$ denote the set of values so queried to f_1 , and set

$$S_1 = \bigcup_{y \in \mathcal{B}} U_y^0.$$

Then $S_1 \subseteq \{0, 1\}^{m+s}$ is the set of inputs for which the relevant query to f_1 is known. Note that $|S_1| = 2^{n/4} \cdot 2^{n/2} = 2^{3n/4}$ by our assumption that each set U_y^0 has size $2^{n/2}$.

We could try, at this point, to estimate the number of colliding inputs in S_1 using Lemma 5, applied with $T = S_0$ and $T' = S_1$, where the sets T_1, \dots, T_k correspond to the family of sets $\{U_y^0 : y \in \{0, 1\}^n\}$ (which form a disjoint partition of S_0), and where $M = 2^{n/2}$ is the upper bound on the size of the T_i 's. Here $k = 2^n$ and, therefore, $p = q/k = 2^{n/3}/2^n = 2^{-2n/3}$. The number of colliding inputs C in $T = S_0$ is at least $|S_0| - 2^s = 2^{1.5n} - 2^n \approx 2^{1.5n}$. We therefore find that

$$p^2 C \approx 2^{-4n/3} 2^{1.5n} = 2^{n/6}.$$

Unfortunately, this number is not as large as $M = 2^{n/2}$ and, in such a case, Lemma 5 does not deliver anything meaningful. We are running up against the afore-mentioned shortcoming of Lemma 5, since we are in a case where the average colliding input in $T = S_0$ does not only collide with $O(1)$ other elements in T , but with very many other elements (or more precisely with $|S_0|/2^s = 2^{n/2}$ other elements).

We overcome this obstacle with a trick. We divide the adversary's querying process into two phases. In the first phase, the adversary selects (deterministically, say) a subset I of $\{0, 1\}^n$ of size $2^{n/2+1}$. In the second phase, the adversary selects a set $\mathcal{B} \subseteq I$ of size $2^{n/4}$ uniformly from all such subsets of I , and queries the elements of \mathcal{B} to f_1 . We emphasize that the elements of I not in \mathcal{B} are not queried to f_1 . Clearly, applying this two-step process is equivalent to directly selecting $2^{n/4}$ values \mathcal{B} uniformly at random from $\{0, 1\}^n$ and querying them to f_1 .

Let

$$S_I = \bigcup_{y \in I} U_y^0.$$

Thus $S_0 \supseteq S_I \supseteq S_1$. Moreover $|S_I| = 2^{n/2+1} \cdot 2^{n/2} = 2^{n+1} = 2^{s+1}$, so S_I contains at least $2^{s+1} - 2^s = 2^s$ colliding inputs. (Note, crucially, that every colliding input in S_I *might* very well collide with only one other input in S_I , so that we are no longer in a case in which Lemma 5 is ignoring a key statistic.) We now apply Lemma 5 with $T = S_I$ and $T' = S_1$, where the sets T_1, \dots, T_k correspond, this time, to the family of sets $\{U_y^0 : y \in I\}$ (which form a partition of S_I). Thus $k = |I| = 2^{n/2+1}$. We have $q/k = 2^{n/4}/2^{n/2+1} \approx 2^{-n/4}$ and

$$p^2 C \approx 2^{-2n/4} 2^s = 2^{n/2}$$

where $C \geq 2^s$ is the number of colliding inputs in S_I . Thus, this time, $p^2 C$ is commensurate with the upper bound $M = 2^{n/2}$ on the size of the T_i 's, and so

Lemma 5 can be effectively applied. (To be a little more precise, by making, say, $200 \cdot 2^{n/2}$ queries to f_1 instead of $2^{n/2}$ queries to f_1 , we can push C and p^2C to significantly higher than M , which remains capped at $2^{n/2}$. Moreover we can make $\frac{1}{20}p^2C \geq 2^{n/2}$ so that, by Lemma 5, we actually obtain $2^{n/2}$ colliding inputs with high probability.)

We emphasize that the above argument does not require f_1 to be “random” at all; f_1 can be *any* fixed function. The only randomness occurs in the selection of the set \mathcal{B} of queries to f_1 .

Finally, the “well-balancedness” assumption on the sets U_y^0 can be removed by using a common refinement of these sets. More precisely, by Lemma 6 in Appendix A, one can always refine the collection of sets $\{U_y^0 : y \in \{0,1\}^n\}$ into a collection of sets each of size at most $2^{n/2}$, at the cost of increasing the number of sets by a factor of at most 2. We then view the adversary as “querying” sets in this refinement (each such set is a subset of a particular U_y^0 and, therefore, associated to a particular value of y). This process may result in redundant queries to f_1 (when two or more subsets of the same U_y^0 are chosen to be queried), but this is harmless. In particular, we do not care about the fact that such redundant queries to f_1 produce dependent results—indeed, from the proof’s standpoint, f_1 is anyway an arbitrary fixed function containing no entropy.

Second Case Study: $(m, n, r, s) = (n, n, 2, n)$. Let $F : \{0,1\}^{m+s} \rightarrow \{0,1\}^n$ be a compression function making calls to two n -bit primitives f_1 and f_2 in fixed-order mode, where $(m, n, r, s) = (n, n, 2, n)$. Thus $F : \{0,1\}^{2n} \rightarrow \{0,1\}^n$. As usual, let g_1 and g_2 be the intermediate processing functions for F .

Stam’s bound forecasts a collision resistance of

$$q = 2^{\frac{nr-m}{r+1}} = 2^{\frac{2n-n}{3}} = 2^{n/3}$$

queries to each f_1 and f_2 .

Let $S_0 = \{0,1\}^{m+s}$ and let

$$U_y^0 = \{x \in S_0 : g_1(x) = y\}.$$

The adversary starts by querying $f_1(y)$ for the q values y for which $|U_y^0|$ is largest. (Note this is a deterministic step.) Then

$$|S_1| \geq 2^{m+s} \left(\frac{q}{2^n} \right) = 2^{2n} \frac{2^{n/3}}{2^n} = 2^{4n/3}$$

where $S_1 \subseteq \{0,1\}^{m+s}$ is the set of inputs for which the relevant query to f_1 has been made. Note that $2^{4n/3} \gg 2^n = 2^s$, so S_1 contains many colliding inputs (more precisely, at least $2^{4n/3} - 2^n \approx 2^{4n/3}$) with probability 1. Moreover, depending on the structure of F and of f_1 , there is no reason one could expect to beat this number of colliding inputs (in S_1) by using a randomized query strategy to f_1 instead of a greedy query strategy to f_1 .

For simplicity, we will assume that S_1 has size exactly $2^{4n/3}$ (anyway the adversary could choose to “throw out” or ignore elements of S_1 to reduce the effective size of S_1 to $2^{4n/3}$, if desired).

At this point, before queries to f_2 are made, note that we are essentially reduced to attacking a compression function F' with parameters $(m', n', r', s') = (n/3, n, 1, n)$ whose domain is S_1 , where $|S_1| = 2^{m'+s'} = 2^{4n/3}$. For such a compression function, Stam's bound quotes a collision resistance of

$$2^{\frac{n'r'-m'}{r'+1}} = 2^{\frac{n-n/3}{2}} = 2^{n/3}$$

queries, which is exactly our budget query for f_2 . We have thus reduced the parameter setting $(m, n, r, s) = (n, n, 2, n)$ of Stam's conjecture to the parameter setting $(n/3, n, 1, n)$, namely to a case of Stam's conjecture where $r = 1$. (This type of reduction was first brought to attention by Steinberger [9].) What follows is therefore fairly similar to the first case study for the parameters $(m, n, r, s) = (0.5n, n, 1, n)$.

Let

$$U_y^1 = \{x \in S_1 : g_2(x, y_1) = y\}$$

for each $y \in \{0, 1\}^n$, where, above, $y_1 = f_1(g_1(x))$ is the first intermediate chaining variable for x (implicitly dependent on x). For simplicity, we can assume that $|U_y^1| = |S_1|/2^n = 2^{n/3}$ for all $y \in \{0, 1\}^n$ (this assumption can be lifted by using a refinement of the U_y^1 's, as in the previous case study). To make his queries to f_2 , the adversary starts by (deterministically) selecting a set $I \subseteq \{0, 1\}^n$ of size $2^{2n/3+1}$. Let

$$S_I = \bigcup_{y \in I} U_y^1$$

so that $|S_I| = 2^{2n/3+1} \cdot 2^{n/3} = 2^{n+1} = 2^{s+1}$. Thus S_I contains at least 2^s colliding inputs. The adversary then selects a random subset \mathcal{B} of I of size $q = 2^{n/3}$, and queries f_2 at all the points in \mathcal{B} . Let

$$S_2 = \bigcup_{y \in \mathcal{B}} U_y^1$$

so that $S_2 \subseteq S_1$ is the set of inputs for which the relevant queries to f_1 and f_2 are both known. Applying Lemma 5 with $T = S_1$, $T' = S_2$, and with sets T_1, \dots, T_k corresponding to $\{U_y^1 : y \in I\}$, where $k = |I| = 2^{2n/3+1}$ and $|T_i| \leq M := 2^{n/3}$ for all i , we find that $p = q/k = 2^{n/3}/2^{2n/3+1} \approx 2^{-n/3}$ and

$$p^2 C \approx 2^{-2n/3} 2^s = 2^{n/3}$$

where $C \geq 2^s$ is the number of colliding inputs in S_I . Since the latter quantity is commensurate with M , we can effectively apply Lemma 5 to conclude that we will obtain $2^{n/3}$ colliding inputs in S_2 with high probability (by making some constant factor more queries than $2^{n/3}$).

Third Case Study: $(m, n, r, s) = (1.25n, n, 4, 2n)$. Let $F : \{0, 1\}^{m+s} \rightarrow \{0, 1\}^n$ be a compression function making calls to four n -bit primitives f_1, \dots, f_4 in fixed-order mode, where $(m, n, r, s) = (1.25n, n, 4, 2n)$. In this case, therefore, $F : \{0, 1\}^{3.25n} \rightarrow \{0, 1\}^{2n}$.

Stam's bound places collision resistance at

$$q = 2^{\frac{nr-m}{r+1}} = 2^{\frac{4n-1.25n}{5}} = 2^{0.55n}$$

queries to each of the primitives f_1, f_2, f_3 and f_4 (which, we note, is less than the cost of a birthday attack).

Let $S_0 = \{0, 1\}^{m+s}$ and let $U_y^0 = \{x \in S_0 : g_1(x) = y\}$ for all $y \in \{0, 1\}^n$. The adversary starts by querying $f_1(y)$ for the q values y for which $|U_y^0|$ is largest. Then

$$|S_1| \geq 2^{m+s} \left(\frac{q}{2^n} \right) = 2^{3.25n} \frac{2^{0.55n}}{2^n} = 2^{2.8n}$$

where S_1 is the set of inputs for which queries to f_1 have been made.

After the queries to f_1 are completed, let $U_y^1 = \{x \in S_1 : g_2(x, y_1) = y\}$ where y_1 is the first intermediate chaining variable for x . For f_2 the adversary again makes greedy queries, i.e. queries $f_2(y)$ for the q values y for which $|U_y^1|$ is largest. Then

$$|S_2| \geq |S_1| \left(\frac{q}{2^n} \right) = 2^{2.8n} \frac{2^{0.55n}}{2^n} = 2^{2.35n}$$

where $S_2 \subseteq S_1$ is the set of inputs for which queries to both f_1 and f_2 have been made. Note S_2 , like S_1 is still larger than $2^s = 2^{2n}$; thus we are “automatically” assured the presence of colliding inputs in S_1 and S_2 by virtue of the size of these sets, which accounts for the sufficiency of the greedy approach.

After the queries to f_2 are completed, let $U_y^2 = \{x \in S_2 : g_3(x, y_1, y_2) = y\}$, where y_1, y_2 are the first two intermediate chaining variables for x . At this point, if the adversary were again to apply a greedy strategy for f_3 , we would find a lower bound of

$$|S_2| \left(\frac{q}{2^n} \right) = 2^{2.35n} \frac{2^{0.55n}}{2^n} = 2^{1.9n}$$

on the size of S_3 , which is no longer larger than 2^s . Applying a (deterministic) greedy strategy would therefore be a very bad idea, since one could easily set up F and its primitives f_1, \dots, f_4 so that S_3 contains no colliding inputs with probability 1, and the adversary finds collisions with probability 0.

Instead, at this stage we revert to using Lemma 5 and the two-step “trick” involving the set I . Assume for simplicity (and in fact without loss of generality) that $|U_y^2| = 2^{2.35n}/2^n = 2^{1.35n}$ for all $y \in \{0, 1\}^n$. The adversary starts by deterministically selecting a set $I \subseteq \{0, 1\}^n$ of size $2^{0.65n+1}$. Let

$$S_I = \bigcup_{y \in I} U_y^2.$$

Thus $|S_I| = 2^{0.65n+1} 2^{1.35n} = 2^{2n+1} = 2^{s+1}$. (We note the adversary has been deterministic up to now—namely the adversary remains deterministic as long as the underlying set of known inputs contains colliding inputs simply by virtue of its size. Now the adversary is about to switch to, and stick with, a randomized

strategy.) The adversary then randomly selects a set $\mathcal{B} \subseteq I$ of size $q = 2^{0.55n}$, and queries these values to f_3 . We set

$$S_3 = \bigcup_{y \in \mathcal{B}} U_y^2.$$

We apply Lemma 5 with $T = S_I$, $T' = S_3$, $\{T_i : 1 \leq i \leq k\} = \{U_y^2 : y \in I\}$, $k = |I| = 2^{0.65n+1}$, $M = 2^{1.35n}$, $p = q/k = 2^{0.55n}/2^{0.65n+1} \approx 2^{-0.1n}$ and $C \geq |S_I| - 2^s \geq 2^s = 2^{2n}$, so that

$$p^2 C \geq 2^{-0.2n} 2^{2n} = 2^{1.8n}.$$

In particular, $p^2 C \gg M$, so Lemma 5 can be effectively applied to show that the number of colliding inputs in S_3 is not much less than $p^2 C = 2^{1.8n}$. For simplicity, we will assume the number of colliding inputs in S_3 is exactly $2^{1.8n}$. Moreover, note that $|S_3| = 2^{1.35n} 2^{0.55n} = 2^{1.9n}$ by virtue of our assumption that $|U_y^2| = 2^{1.35n}$ for each $y \in \{0, 1\}^n$.

For queries to f_4 , the adversary directly continues with a randomized strategy and an application of Lemma 5—no need for a preliminary selection of inputs I , here, because the number of colliding inputs in S_3 is already less than 2^s .

More precisely, let $U_y^3 = \{x \in S_3 : g_4(x, y_1, y_2, y_3) = y\}$ for all $y \in \{0, 1\}^n$, where y_1, y_2, y_3 are the intermediate chaining variables for x . Assume for simplicity that $|U_y^3| = |S_3|/2^n = 2^{1.9n}/2^n = 2^{0.9n}$ for all y . The adversary selects a random set $\mathcal{B} \subseteq \{0, 1\}^n$ of size $q = 2^{0.55n}$, and queries these values to f_4 . Set

$$S_4 = \bigcup_{y \in \mathcal{B}} U_y^3.$$

We apply Lemma 5 with $T = S_3$, $T' = S_4$, $\{T_i : 1 \leq i \leq k\} = \{U_y^3 : y \in \{0, 1\}^n\}$, $k = |I| = 2^n$, $M = 2^{0.9n}$, $p = q/k = 2^{0.55n}/2^n = 2^{-0.45n}$ and $C = 2^{1.8n}$, where the latter equality comes from our simplifying assumption that S_3 contains exactly $2^{1.8n}$ colliding inputs. Then

$$p^2 C = 2^{-0.9n} 2^{1.8n} = 2^{0.9n}$$

so that $p^2 C$ is commensurate with $M = 2^{0.9n}$, and Lemma 5 can be effectively applied (after, potentially, multiplying the number of queries by some small constant) to show that at least

$$p^2 C = 2^{0.9n} = 2^{2n-1.1n} = 2^{s-\frac{2(nr-m)}{r+1}}$$

colliding inputs can be obtained with good probability.

DIGEST. The last case study exhibits more or less all the features of the general case. In the general case, the adversary's querying strategy has two phases. The first phase is a “deterministic” phase where the adversary makes greedy queries to maximize the yield. This phase lasts as long as the next set S_i obtained is guaranteed to be larger than 2^s . This phase also “spills over” into the (still deterministic) selection of the set I . The second phase then commences, consisting

of purely random queries. (First q random queries selected from I and then, for subsequent f_i 's, q random queries selected from $\{0, 1\}^n$.) It so turns out that the “phase change” occurs exactly when it is time to make queries to f_{r_0+1} where

$$r_0 = \left\lfloor \frac{m(r+1)}{m+n} \right\rfloor.$$

Thus, in the general case, the two-phase strategy determines a sequence of sets

$$S_0 \supseteq S_1 \supseteq \cdots \supseteq S_{r_0} \supseteq S_I \supseteq S_{r_0+1} \supseteq \cdots \supseteq S_r$$

where $S_0 = \{0, 1\}^{m+s}$ is F 's domain and S_i , $i \geq 1$, is the set of inputs for which the queries to f_1, \dots, f_i have been made. (When $\frac{m(r+1)}{m+n}$ happens to be an integer—which does not occur in any of the case studies above—then $r_0 = \frac{m(r+1)}{m+n} \geq 1$ and, by adding a constant factor to the number of queries, one finds $|S_{r_0}| \geq 2^{s+1}$ instead of $|S_{r_0}| = 2^s$, so that there is “still room” for S_I to be selected.)

One can point out that the number of colliding inputs in the sets S_0, \dots, S_r evolves differently during the first and second phases. During the first phase, each colliding input in S_i collides on average with a very large number of other points, so that the key factor determining whether a colliding input makes it from S_i to S_{i+1} (assuming $i+1 \leq r_0$) is just whether that particular point makes it to S_{i+1} (since it is very likely that at least one of the myriad other points it collides with has made it to S_{i+1} as well). The “rate of attrition” of colliding inputs is therefore $p = q/2^n$ in going from S_i to S_{i+1} , for $i+1 \leq r_0$, and, similarly, is $|I|/2^n$ in going from S_{r_0} to S_I . During the second phase, on the other hand, both a colliding input *and the (on average unique) other input it collides with* must simultaneously survive the selection process, so that the rate of attrition of colliding inputs in going from S_I to S_{r_0+1} is $(q/|I|)^2$ whereas the rate of attrition in going from S_i to S_{i+1} is $(q/2^n)^2$ for $r_0+1 \leq i \leq r-1$. It is possible to compute that these rates of attrition lead to a final expected number of colliding inputs equal to $2^s - \frac{2(nr-m)}{r+1}$. The latter also equals, by no coincidence, $|S_{r-1}|/2^n$.

COMPARISON WITH [9]. The proof of Lemma 5—our paper’s “key lemma”—uses ideas from Steinberger’s “MECMAC lemma” [9] (a lemma which is actually unused in the main result of [9]), and more precisely recycles the nice idea of using a bipartition of sets to overcome dependencies between collision events. Our work also uses Steinberger’s parameter reduction idea (as discussed in the second case study). However, these are essentially the only similarities with [9]. In particular, our proof does not consist in a generalization of Steinberger’s techniques, since our proof, as restricted to $r=1$, does not reduce to a birthday attack, but instead uses Lemma 5 which itself relies on Martingale concentration results. Moreover, the key idea of focusing on the number of colliding inputs (as opposed to the more usual “number of colliding pairs of inputs”) as the correct metric for measuring the progress of an attack is an original contribution of this paper.

5 Future Work

Many related interesting open problems remain. One of the basic questions that remains is to show Stam's bound is tight. This would require exhibiting an infinite class of compression functions (parameterized by m , n , r , s , where m , r and s are linear functions of n) whose collision resistance is provably in the vicinity of

$$\min(2^{s/2}, \lceil 2^{\frac{nr-m}{r+1}} \rceil).$$

Another remaining open question concerns parallelism. Could better attacks be found for compression functions that call their primitives in parallel? So far, rather amazingly, we are not aware of any provable separation between the power of parallel and sequential compression functions. A third type of question concerns adapting results like those in this paper to compression functions with primitives of not-all-equal input lengths and, maybe more interestingly, to primitives with small output lengths. Indeed, primitives with small output lengths constitute a vulnerability, as pointed out by Stam [8], though a classification and quantification of such vulnerabilities still awaits.

References

1. Bellare, M., Kohno, T.: Hash Function Balance and Its Impact on Birthday Attacks. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 401–418. Springer, Heidelberg (2004)
2. Black, J., Cochran, M., Shrimpton, T.: On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 526–541. Springer, Heidelberg (2005)
3. Chung, F., Lu, L.: Concentration Inequalities and Martingale Inequalities: A Survey. *Internet Mathematics* 3(1), 79–127
4. McDiarmid, C.: Concentration. In: Habib, M., McDiarmid, C., Ramier-Alfonsin, J., Reed, B. (eds.) *Probabilistic Methods for Algorithmic Discrete Mathematics. Algorithms and Combinatorics*, vol. 16, pp. 195–248. Springer (1998)
5. Rogaway, P., Steinberger, J.: Constructing Cryptographic Hash Functions from Fixed-Key Blockciphers. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 433–450. Springer, Heidelberg (2008)
6. Rogaway, P., Steinberger, J.: Security/Efficiency Tradeoffs for Permutation-Based Hashing. In: Smart, N.P. (ed.) *EUROCRYPT 2008*. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008)
7. Shrimpton, T., Stam, M.: Building a Collision-Resistant Compression Function from Non-compressing Primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *ICALP 2008, Part II*. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008); Also available at the Cryptology ePrint Archive: Report 2007/409
8. Stam, M.: Beyond Uniformity: Better Security/Efficiency Tradeoffs for Compression Functions. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 397–412. Springer, Heidelberg (2008)

9. Steinberger, J.: Stam's Collision Resistance Conjecture. In: Gilbert, H. (ed.) EU-ROCRYPT 2010. LNCS, vol. 6110, pp. 597–615. Springer, Heidelberg (2010)
10. Wiener, M.: Bounds on birthday attack times. Cryptology ePrint archive (2005)
11. Wu, H.: The JH hash function. NIST SHA-3 competition submission (October 2008)

A Supporting Lemmas

We recall that for random variables X, Y , a notation such as $\text{Var}(X|Y = s)$ means the variance of X conditioned on the event $Y = s$, whereas $\text{Var}(X|Y)$ is a *function* from the range of Y to \mathbb{R} , that assigns $\text{Var}(X|Y = s)$ to each s in the range of Y (or more precisely, to each s such that $\Pr[Y = s]$ is nonzero). The notation “ $\text{Var}(X|Y) \leq c$ ” indicates this function is upper bounded by c : $\text{Var}(X|Y = s) \leq c$ for all s such that $\Pr[Y = s] > 0$.

We use, as a starting point, the following concentration result for Martingales. See Theorem 6.1 of [3] for a proof. (We note that our notation is slightly modified from standard in order to avoid discussion of filters.)

Lemma 2 (Folklore [3, 4]). *Let Y_1, \dots, Y_n be a sequence of random variables of range R , $f : R^n \rightarrow \mathbb{R}$ be a function and let $Y = f(Y_1, \dots, Y_n)$. Let*

$$X_i = E[Y|Y_1, \dots, Y_i]$$

for $0 \leq i \leq n$. Then if

1. $\text{Var}(X_i|Y_1, \dots, Y_{i-1}) \leq \sigma_i^2$ for $1 \leq i \leq n$, and
2. $|X_i - X_{i-1}| \leq M$, for every $1 \leq i \leq n$,

we have

$$\Pr[Y \leq E[Y] - \lambda] \leq e^{-\frac{\lambda^2}{2(\sum_{i=1}^n \sigma_i^2 + M\lambda/3)}}$$

for any $\lambda \geq 0$.

Lemma 3. *Let k, q be integers such that $1 \leq q \leq k$. Let \mathcal{B} be random a subset of $[k] = \{1, \dots, k\}$ of size q . Let M and c_1, \dots, c_k be nonnegative constants such that $M \geq c_i$ for $1 \leq i \leq k$. Put $Y = \sum_{i \in \mathcal{B}} c_i$. Then*

$$\Pr[Y \leq E[Y] - t] \leq e^{-\frac{t^2}{2M(3E[Y]+t/3)}}$$

for all $t \geq 0$.

Proof. Note that if $q = k$ the lemma is obviously true, and so we can assume $q < k$.

We view the elements of \mathcal{B} as being selected sequentially, with the i -th element of \mathcal{B} coming uniformly at random from a set of size $k - i + 1$ (the complement of the currently selected elements). Let s_i be the i -th chosen element, and define $f : [k]^q \rightarrow \mathbb{R}$ by $f(s_1, \dots, s_q) = \sum_{i=1}^q c_{s_i}$. Note $Y = f(s_1, \dots, s_q)$. In view of applying Lemma 2 (with $Y_i = s_i$), we define

$$X_i = E[Y|s_1, \dots, s_i]$$

for $0 \leq i \leq q$. Thus, X_i is the expected “value” of \mathcal{B} after the first i elements have been chosen.

Note that for any values $t_1, \dots, t_q \in [k]$ and $t'_i \in [k]$,

$$|f(t_1, \dots, t_q) - f(t_1, \dots, t_{i-1}, t'_i, t_{i+1}, \dots, t_q)| \leq M.$$

That is, changing the i -th input of f (i.e. the i -th element chosen) can only change f 's output by M , at most. It follows (by a short but standard argument) that $|X_i - X_{i-1}| \leq M$ for $1 \leq i \leq q$.

We next want to upper bound $\text{Var}(X_{i+1} | s_1, \dots, s_i)$ independently of s_1, \dots, s_i . We have:

$$\begin{aligned} X_{i+1} &= c_{s_{i+1}} + \frac{q-i}{k-i-1} \sum_{h \notin \{s_1, \dots, s_{i+1}\}} c_h + \sum_{j=1}^i c_{s_j} \\ &= c_{s_{i+1}} \left(1 - \frac{q-i}{k-i-1}\right) + \frac{q-i}{k-i-1} \sum_{h \notin \{s_1, \dots, s_i\}} c_h + \sum_{j=1}^i c_{s_j} \\ &= c_{s_{i+1}} \left(1 - \frac{q-i}{k-i-1}\right) + K \end{aligned}$$

where K is a constant depending only on s_1, \dots, s_i . Therefore,

$$\begin{aligned} \text{Var}(X_{i+1} | s_1, \dots, s_i) &= \left(1 - \frac{q-i}{k-i-1}\right)^2 \cdot \text{Var}(c_{s_j} | s_1, \dots, s_i) \\ &\leq \left(1 - \frac{q-i}{k-i-1}\right)^2 \cdot \frac{1}{k-i} \sum_{j \notin \{s_1, \dots, s_i\}} c_j^2 \\ &\leq \frac{(k-q)^2}{(k-i)(k-i-1)^2} \sum_{j=1}^k c_j^2. \end{aligned}$$

We set σ_{i+1}^2 to this last expression, $0 \leq i < q$, so that $\text{Var}(X_{i+1} | s_1, \dots, s_i) \leq \sigma_{i+1}^2$.

Let $p = q/k$. Note that $p < 1$ since we are assuming $q < k$ and that $X_0 = E[Y] = p \sum_{i=1}^k c_i$. We have

$$\begin{aligned} \sum_{i=1}^q \sigma_i^2 &= \sum_{j=1}^k c_j^2 \cdot (k-q)^2 \sum_{i=0}^{q-1} \frac{1}{(k-i)(k-i-1)^2} \\ &\leq \sum_{j=1}^k c_j^2 \cdot (k-q)^2 \sum_{i=1}^q \frac{1}{(k-i)^3} \\ &= \sum_{j=1}^k c_j^2 \cdot (k-q)^2 \left(\sum_{i=0}^{q-1} \frac{1}{(k-i)^3} + \frac{1}{(k-q)^3} - \frac{1}{k^3} \right) \\ &\leq \sum_{j=1}^k c_j^2 \cdot (k-q)^2 \left(\int_0^q \frac{1}{(k-x)^3} dx + \frac{1}{(k-q)^3} - \frac{1}{k^3} \right) \end{aligned}$$

$$\begin{aligned}
&= \sum_{j=1}^k c_j^2 \cdot \left((k-q)^2 \cdot \frac{1}{2} \left(\frac{1}{(k-q)^2} - \frac{1}{k^2} \right) + \frac{1}{k-q} - \frac{(k-q)^2}{k^3} \right) \\
&= \left(\frac{1}{2} \left(1 - \frac{(k-q)^2}{k^2} \right) + \frac{1}{k-q} - \frac{(k-q)^2}{k^3} \right) \sum_{j=1}^k c_j^2 \\
&= \left(\frac{1}{2} (1 - (1-p)^2) + \frac{1}{k(1-p)} - \frac{(1-p)^2}{k} \right) \sum_{j=1}^k c_j^2 \\
&\leq \left(\frac{1}{2} (2p - p^2) + \frac{1}{k(1-p)} \right) \sum_{j=1}^k c_j^2 \\
&= p \left(\frac{1}{2} (2-p) + \frac{1}{q(1-p)} \right) \sum_{j=1}^k c_j^2 \\
&= p \left(\frac{1}{2} (2-p) + \frac{1}{q} + \frac{1}{k(1-p)} \right) \sum_{j=1}^k c_j^2 \\
&\leq p \left(\frac{1}{2} (2-p) + \frac{1}{q} + \frac{1}{k^{\frac{1}{k}}} \right) \sum_{j=1}^k c_j^2 \\
&\leq 3p \sum_{j=1}^k c_j^2 \\
&\leq 3p \sum_{j=1}^k c_j M \\
&= 3E[Y]M
\end{aligned}$$

Then by Lemma 2, we have

$$\Pr[Y < E[Y] - t] \leq e^{-\frac{t^2}{2(\sum_{i=1}^q \sigma_i^2 + Mt/3)}} \leq e^{-\frac{t^2}{2(3E[Y]M + Mt/3)}} = e^{-\frac{t^2}{2M(3E[Y] + t/3)}}. \quad \square$$

Lemma 4. Let k, q be integers such that $1 \leq q \leq k$. Let \mathcal{B} be random a subset of $[k] = \{1, \dots, k\}$ of size q . Let M and c_1, \dots, c_k be nonnegative constants such that $M \geq c_i$ for $1 \leq i \leq k$. Put $Y = \sum_{i \in \mathcal{B}} c_i$. Then

$$\Pr[Y < \phi - t] \leq e^{-\frac{t^2}{2M(3\phi + t/3)}} \tag{4}$$

for any t, ϕ such that $0 \leq t \leq \phi \leq E[Y]$.

Proof. Let $u = E[Y] - \phi$. Then by Lemma 3,

$$\begin{aligned}
\Pr[Y < \phi - t] &= \Pr[Y < E[Y] - u - t] \\
&\leq e^{-\frac{(t+u)^2}{2M(3(u+\phi)+(t+u)/3)}}
\end{aligned} \tag{5}$$

Let $f(u) = (t+u)^2$, $g(u) = 3(u+\phi) + (t+u)/3$, we find that

$$\begin{aligned}
\left(\frac{f(u)}{g(u)}\right)' &\geq 0 \iff f'(u)g(u) \geq g'(u)f(u) \\
&\iff 2g(u) \geq g'(u)(t+u) \\
&\iff 2g(u) \geq (3+1/3)(t+u) \\
&\iff g(u) \geq (3+1/3)(t+u) \\
&\iff 3(u+\phi) + (t+u)/3 \geq (3+1/3)(t+u) \\
&\iff 3(u+t) + (t+u)/3 \geq (3+1/3)(t+u)
\end{aligned}$$

where we use $\phi \geq t$ for the last implication. Thus (5), considered as a function of u and restricted to $u \geq 0$, takes its maximum at $u = 0$, which establishes (4). \square

Lemma 5. *Let k, q be integers such that $1 \leq q \leq k$ and such that q is even. Let $M > 0$ be a constant and let T be the disjoint union of sets T_1, \dots, T_k such that $|T_i| \leq M$ for $1 \leq i \leq k$. Let $F : T \rightarrow U$ be some function and let*

$$C_i = |\{x \in T_i : \exists y \in T, y \neq x, \text{ s.t. } F(x) = F(y)\}|$$

Let $C = C_1 + \dots + C_k$. Let \mathcal{B} be a random subset of $[k]$ of size q . Let

$$\overline{C}_i = |\{x \in T_i : \exists j \in \mathcal{B}, y \in T_j, y \neq x, F(x) = F(y)\}|$$

and let

$$\overline{C} = \sum_{i:i \in \mathcal{B}} \overline{C}_i$$

then

$$\Pr[\overline{C} < t] \leq 2e^{-\frac{t}{16M}}$$

where $t = \frac{1}{20}p^2C$ and $p = q/k$.

Proof. We use the following equivalent selection process for \mathcal{B} : we first select, independently and uniformly at random, two subsets L and R of $[k]$ of size $q/2$ each, then select an additional set H of size $q - |L \cup R|$ uniformly at random from $[k]$, and finally set $\mathcal{B} = L \cup R \cup H$. Clearly, this process yields a set \mathcal{B} of size q that is uniformly distributed at random among all subsets of $[k]$ of size q .

Define random variables Y_1, \dots, Y_k by putting $Y_i = C_i$ if $i \in L$, $Y_i = 0$ otherwise. Let $Y = \sum_{i=1}^k Y_i$. We have $|Y_i| \leq |T_i| \leq M$. Note that $E[Y] = \frac{q/2}{k}C = \frac{1}{2}pC$. Let $t_0 = \frac{1}{4}pC = \frac{1}{2}E[Y]$. By Lemma 3,

$$\begin{aligned}
\Pr[Y < E[Y] - t_0] &\leq e^{-\frac{t_0^2}{2M(3E[Y]+t_0/3)}} \\
&= e^{-\frac{p^2C^2/16}{2M(3pC/2+pC/12)}} \\
&= e^{-\frac{pC}{32M(3/2+1/12)}} \\
&\leq e^{-\frac{pC}{51M}}.
\end{aligned}$$

For the rest of the proof we assume that $Y \geq E[Y] - t_0 = \frac{1}{4}pC$. For $1 \leq i \leq k$, let

$$C_i^L = |\{x \in T_i : \exists y \in T_j, j \in L, y \neq x, \text{ s.t. } F(x) = F(y)\}|.$$

Recall that $t = \frac{1}{20}p^2C$. It is not difficult to see that if $\sum_{i \in [k]} C_i^L \leq \sum_{i \in L} C_i - t$, then

$$\sum_{i \in L} |\{x \in T_i : \exists y \in T_j, j \in L, y \neq x, \text{ s.t. } F(x) = F(y)\}| \geq t + 1,$$

implying that $\overline{C} \geq t$. We can therefore assume that $\sum_{i \in [k]} C_i^L \geq \sum_{i \in L} C_i - t \geq \frac{1}{4}pC - t$.

Define random variables Z_1, \dots, Z_k by putting $Z_i = C_i^L$ if $i \in R$, $Z_i = 0$ otherwise, and let $Z = \sum_{i=1}^k Z_i$. Then $|Z_i| \leq |T_i| \leq M$ for all i . Let $\phi = (p/2)(\frac{1}{4}pC - t) \leq E[Z]$ (the latter equality follows from the fact that each set is added to R with probability $p/2$, and from the fact that $\sum_{i \in [k]} C_i^L \geq \frac{1}{4}pC - t$). Then

$$\phi = \frac{1}{8}p^2C - \frac{1}{2}pt = (2.5 - \frac{1}{2}p)t \geq 2t.$$

Since $0 \leq t \leq \phi \leq E[Z]$, $t \leq \phi - t$, and $\phi \leq 2.5t$, we have by Lemma 4 that

$$\begin{aligned} \Pr[Z < t] &\leq \Pr[Z < \phi - t] \\ &\leq e^{-\frac{t^2}{2M(3\phi+t/3)}} \\ &\leq e^{-\frac{t^2}{2M(3 \cdot 2.5t+t/3)}} \\ &= e^{-\frac{t}{M(15+2t/3)}} \\ &\leq e^{-\frac{t}{16M}} \end{aligned}$$

Since $\overline{C} \geq Z$ and since $e^{-\frac{t}{16M}} = e^{-\frac{p^2C}{20 \cdot 16M}} \geq e^{-\frac{p^2C}{51M}} \geq e^{-\frac{pC}{51M}}$, a sum bound on the two bad events (these being the event that either $Y < E[Y] - t_0$, or that $Z < t$) concludes the lemma. \square

Lastly, Lemma 6 below notes an following elementary result related to refinements of a set of disjoint sets, defined next.

Definition 2. Let U_1, \dots, U_ℓ be a collection of finite disjoint sets. Another collection T_1, \dots, T_k of finite disjoint sets is a refinement of U_1, \dots, U_ℓ if $\bigcup_{i=1}^k T_i = \bigcup_{i=1}^\ell U_i$ and if either $T_i \subseteq U_j$ or $T_i \cap U_j = \emptyset$ for all $1 \leq i \leq k$, $1 \leq j \leq \ell$.

Lemma 6. Let U_1, \dots, U_ℓ be disjoint finite sets. Let $M \geq 1$ be a positive integer upper bounding the average size of the U_i 's. (That is, $M \geq (\sum_i |U_i|)/\ell$.) Then there exists a refinement T_1, \dots, T_k of the sets U_1, \dots, U_ℓ such that $|T_i| \leq M$ for all i and such that $k \leq 2\ell$.

Proof. We can refine each set U_i into at most $\lceil \frac{|U_i|}{M} \rceil$ sets of size at most M each⁸. Thus we can find a refinement T_1, \dots, T_k of U_1, \dots, U_ℓ where $|T_i| \leq M$ for all $1 \leq i \leq k$ and where

$$k \leq \sum_{i=1}^{\ell} \left\lceil \frac{|U_i|}{M} \right\rceil \leq \sum_{i=1}^{\ell} \left(\frac{|U_i|}{M} + 1 \right) \leq 2\ell. \quad \square$$

⁸ Note this actually requires M to be an integer.

Universal Composability from Essentially Any Trusted Setup

Mike Rosulek^{*}

Department of Computer Science, University of Montana
`mikero@cs.umt.edu`

Abstract. It is impossible to securely carry out general multi-party computation in arbitrary network contexts like the Internet, unless protocols have access to some trusted setup. In this work we classify the power of such trusted (2-party) setup functionalities. We show that nearly every setup is either **useless** (ideal access to the setup is equivalent to having no setup at all) or else **complete** (composably secure protocols for *all* tasks exist in the presence of the setup). We further argue that those setups which are neither complete nor useless are highly unnatural.

The main technical contribution in this work is an almost-total characterization of completeness for 2-party setups. Our characterization treats setup functionalities as black-boxes, and therefore is the first work to classify completeness of *arbitrary setup functionalities* (*i.e.*, randomized, reactive, and having behavior that depends on the global security parameter).

1 Introduction

When a protocol is deployed in a vast network like the Internet, it may be executed in the presence of concurrent instances of other arbitrary protocols with possibly correlated inputs. A protocol that remains secure in such a demanding context is called *universally composable*. This security property is highly desirable; unfortunately, it is simply too demanding to be achieved for every task. Canetti's UC framework [5] provides the means to formally reason about universal composability in a tractable way, and it is widely regarded as the most realistic model of security for protocols on the Internet. A sequence of impossibility results [8,30,9] culminated in a complete characterization for which tasks are securely realizable in the UC framework [35]. Indeed, universal composability is impossible for almost all tasks of any cryptographic interest, under any intractability assumption.

For this reason, there have been many attempts to slightly relax the UC framework to permit secure protocols for more tasks, while still keeping its useful composition properties. Many of these variants are extremely powerful, permitting composably-secure protocols for *all* tasks; for example: adding certain trusted setup functionalities [8,11,1,12,17,23,21,31], allowing superpolynomial simulation

^{*} Supported by NSF grant CCF-1149647.

[34,36,2,32,10], assuming bounded network latency [22], considering uniform adversaries [29], and including certain global setups [7], to name a few (for a more comprehensive treatment, see the survey by Canetti [6]). Other variants of the UC framework turn out to be no more powerful than the original framework; for example, adding certain setup functionalities [35,25] or global setups [7], and requiring only self-composition rather than universal composition [30]. A natural question is, therefore: under what circumstances can universal composability be achieved?

1.1 Our Results

In this work we study the power of 2-party trusted setups. In other words, given access to a particular trusted setup functionality (e.g., a common random string functionality), what tasks have UC-secure protocols? In particular, two extremes are of interest. First, we call a trusted setup \mathcal{F} **useless** if having ideal access to \mathcal{F} is equivalent to having no trusted setup at all. More precisely, \mathcal{F} is useless if it already has a UC-secure protocol in the plain (no trusted setups) model. A complete characterization for such functionalities was given in [35].

At the other extreme, we call a trusted setup \mathcal{F} **complete** if *every* well-formed task has a UC-secure protocol given ideal access to \mathcal{F} . As mentioned above, many setups are known to be complete (e.g., a common random string functionality), but the methods for demonstrating their completeness have been quite *ad hoc*. Our major contribution is to give a new framework for understanding when a trusted setup is complete.

Informal statement of the results. Our characterization is based on the concept of *splittability* from [35]. To give a sense of splittability, imagine the following two-party game between a “synchronizer” and a “distinguisher.” The parties connect to two independent instances of \mathcal{F} , each party playing the role of Alice in one instance and the role of Bob in the other. They are allowed to arbitrarily interact with these two instances of \mathcal{F} . The synchronizer’s goal is to make the two instances behave like a single instance of \mathcal{F} , from the distinguisher’s point of view. The distinguisher’s goal is to make the two instances act noticeably different from a single instance of \mathcal{F} , from his point of view.

Then, informally, we say that \mathcal{F} is **splittable** if the synchronizer has a winning strategy in this game, and \mathcal{F} is **strongly unsplittable** if the distinguisher has a winning strategy.¹ Importantly, these splittability-based conditions are relatively easy to check for a candidate setup, and apply uniformly to *completely arbitrary* functionalities in the UC framework (i.e., possibly randomized, reactive, with behavior depending on the security parameter). Prabhakaran & Rosulek [35] showed that \mathcal{F} is useless if and only if \mathcal{F} is splittable. Analogously, our main result is the following:

Theorem (Informal). *If \mathcal{F} is strongly unsplittable*, then \mathcal{F} is complete.*

¹ In the formal definition, both players are computationally bounded, so it may be that neither party has a feasible winning strategy in the 2-party game. See Section 3.2.

The asterisk after “strongly unsplittable” indicates that the precise statement of our result involves a variant of strong unsplittability, in which the “synchronizer” is allowed to obtain the internal state of one of the instance of \mathcal{F} . However, in the case that \mathcal{F} is a *nonreactive* functionality, the informal statement above is correct; the asterisk can be safely ignored.

Our completeness theorem is proved under the assumption that there exists a semi-honest secure oblivious transfer protocol (the SHOT assumption), an intractability assumption we show is necessary. We also show that a very slight modification of strong unsplittability is *necessary* for a setup to be complete, and so our characterization is nearly tight.

We argue that setups which are neither splittable nor strongly unsplittable exploit low-level technical “loopholes” of the UC framework or are otherwise highly unnatural. Thus, combining our result with that of [35], we can informally summarize the power of trusted setups by saying that every “natural” setup is either useless or complete.

Our notion of completeness. Many prior completeness results place restrictions on how protocols are allowed to access a setup functionality. A common restriction is to allow protocols to access only one instance of the setup — typically only at the beginning of a protocol. In these cases, we say that the protocols have only *offline access* to the setup. Additionally, some completeness results construct a *multi-session* commitment functionality from such offline access to the setup, modeling a more globally available setup assumption.

In contrast, the completeness results in this work are of the following form. We say that a functionality \mathcal{F} is a **complete** setup if there is a UC-secure protocol for the ideal (single-session) commitment functionality in the \mathcal{F} -hybrid model. This corresponds to the complexity-theoretic notion of completeness, under the reduction implicit in the conventional UC security definition. As is standard for protocols in the \mathcal{F} -hybrid model, we place no restrictions on how or when protocols may access \mathcal{F} or how many instances of \mathcal{F} they may invoke. However, we point out that completeness for offline access can be achieved by simply using our construction to generate a common random string in an offline phase, then applying a result such as [11].

Technical overview. To prove that a functionality \mathcal{F} is complete, it suffices to show that there is a UC-secure protocol for bit commitment, given ideal access to \mathcal{F} . This follows from the well-known result of Canetti *et al.* [11] that commitment is complete, under the SHOT assumption. We construct a commitment protocol in several steps: In Sections 4 & 5 we construct (for the two cases in our main theorem) commitment protocols that have a straight-line UC simulator for corrupt receivers (*i.e.*, an equivocating simulator) but have only a rewinding simulator for corrupt senders (*i.e.*, an extracting simulator). Then, in Section 6 we show how to use such a “one-sided” UC-secure commitment protocol to build a full-fledged UC-secure commitment protocol.

In Section 7 we show that several variants of strong unsplittability are necessary for a setup to be complete. These variants involve only very minor technical modifications to the strong unsplittability definition.

1.2 Related Work

Dichotomies in the cryptographic power of functionalities are known in several settings [13,28]. Our work follows a rich line of research exhibiting dichotomies specifically between *useless* and *complete* functionalities, for various security models and restricted to various classes of functionalities [3,26,27,18,31,24]. Among these results, only [31] considered *reactive* functionalities, and only [26] considered *randomized* functionalities. In comparison, ours is the first work to consider the full range of arbitrary functionalities allowed by the UC framework (the class of functionalities is stated explicitly in Section 2.2).

Among the results listed above, two [31,24] are cast in the UC framework; for these we give a more detailed comparison to our own results. Maji, Prabhakaran, and Rosulek [31] showed a uselessness/completeness dichotomy among *deterministic* functionalities whose internal state and input/output alphabets are constant-sized (*i.e.*, independent of the security parameter). Their approach relies heavily on deriving combinatorial criteria for such functionalities, expressed as finite automata, whereas our classification achieves much greater generality by treating functionalities essentially as black-boxes. Concurrently and independently of this work, Katz *et al.* [24] showed a similar result for deterministic, non-reactive (secure function evaluation) functionalities.² They construct UC-puzzles [29] for classes of SFE functionalities deemed impossible in the characterization of Canetti, Kushilevitz, and Lindell [9]. Our result achieves greater generality by being based not on the CKL characterization but the splittability characterization of [35]. Furthermore, we show completeness by directly constructing a commitment protocol rather than a UC puzzle (see below).

Lin, Pass, and Venkatasubramaniam [29] developed a framework for proving completeness results in the UC framework and many variants. Their framework is based on “UC-puzzles” — protocols with an explicit *trapdoor* and satisfying a *statistical simulation* property. Using UC-puzzles, they construct round-optimal protocols for all functionalities. Our work focuses on completeness in terms of *feasibility* rather than the efficiency. To explicitly highlight the uselessness/completeness dichotomy, our completeness criterion is closely tied to the existing notion of splittability. Consequently, our criterion is less demanding (requiring only a *distinguisher*) and is tailored exclusively towards setup functionalities (not more fundamental modifications to the UC framework). We leave it as an open problem whether strong unsplittability can be used to construct a UC puzzle to give an efficiency improvement for our result. In particular, the requirement of a statistical simulation seems incompatible with strong unsplittability, which gives only computational properties.

² Our result and that of [24] use different formulations for SFE functionalities. See the discussion in the full version.

In the full version we show that strong unsplittability can be used to understand many previous (*ad hoc*) completeness results involving trusted setups. However, not all setups considered in previous works are within the class of functionalities we consider for the general result here (in particular, many rely crucially on the setup interacting with the adversary).

2 Preliminaries

A function $f : \mathbb{N} \rightarrow [0, 1]$ is *negligible* if for all $c > 0$, we have $f(n) < 1/n^c$ for all but finitely many n . A function f is *noticeable* if there exists some $c > 0$ such that $f(n) \geq 1/n^c$ for all but finitely many n . We emphasize that there exist functions that are neither negligible nor noticeable (e.g., $f(n) = n \bmod 2$). A probability $p(n)$ is *overwhelming* if $1 - p(n)$ is negligible.

2.1 Universal Composability

We assume some familiarity with the framework of universally composable (UC) security; for a full treatment, see [5]. We use the notation $\text{EXEC}[\mathcal{Z}, \mathcal{F}, \pi, \mathcal{A}, k]$ to denote the probability that the environment outputs 1, in an interaction involving environment \mathcal{Z} , a single instance of an ideal functionality \mathcal{F} , parties running protocol π , adversary \mathcal{A} , with global security parameter k . All entities in the system must be PPT interactive Turing machines (see [19] for a complete treatment of “polynomial time” definitions for the UC framework). We consider security only against *static* adversaries, who corrupt parties only at the beginning of a protocol execution. π_{dummy} denotes the *dummy protocol* which simply relays messages between the environment and the functionality.

A protocol π is a *UC-secure* protocol for functionality \mathcal{F} in the \mathcal{G} -hybrid model if for all adversaries \mathcal{A} , there exists a simulator \mathcal{S} such that for all environments \mathcal{Z} , we have that $|\text{EXEC}[\mathcal{Z}, \hat{\mathcal{G}}, \pi, \mathcal{A}, k] - \text{EXEC}[\mathcal{Z}, \mathcal{F}, \pi_{\text{dummy}}, \mathcal{S}, k]|$ is negligible in k . Here, $\hat{\mathcal{G}}$ denotes the multi-session version of \mathcal{G} , so that the protocol π is allowed to access multiple instances of \mathcal{G} in the \mathcal{G} -hybrid model. The former interaction (involving π and \mathcal{G}) is called the **real process**, and the latter (involving π_{dummy} and \mathcal{F}) is called the **ideal process**.

We consider a communication network for the parties in which the adversary has control over the timing of message delivery. In particular, there is no guarantee of *fairness* in output delivery.

2.2 Class of Functionalities

Our results apply to essentially the same class of functionalities considered in the feasibility result of Canetti *et al.* [11]. First, the functionality must be *well-formed*, meaning that it ignores its knowledge of which parties are corrupt.

Second, the functionality must be represented as a (uniform) circuit family $\{C_k \mid k \in \mathbb{N}\}$, where C_k describes a single activation of the functionality when the security parameter is k . For simplicity, we assume that C_k receives k bits of

the functionality's internal state, k bits of randomness (independent randomness in each activation), a k -bit input from the activating party, and the identity of the activating party as input, and then outputs a new internal state and k bits of output to each party (including the adversary). Note that all parties receive outputs; in particular, all parties are informed of every activation. We focus on 2-party functionalities and refer to these parties as Alice and Bob throughout.

Finally, we require that the functionality do nothing when activated by the adversary.³

2.3 The SHOT Assumption and Required Cryptographic Primitives

The SHOT assumption is that there exists a protocol for $\binom{2}{1}$ -oblivious transfer that is secure against semi-honest PPT adversaries (equivalently, standalone-secure, by standard compilation techniques). From the SHOT assumption it also follows that there exist *standalone-secure* protocols for every functionality in the class defined above [15,11]. We require a simulation-based definition of standalone security, equivalent to the restriction of UC security to environments that do not interact with the adversary during the execution of the protocol.

The SHOT assumption implies the existence of one-way functions [20], which in turn imply the existence of standalone-secure statistically-binding commitment schemes [33] and zero-knowledge proofs of knowledge [16,4] that we use in our constructions. One-way functions also imply the existence of **non-malleable secret sharing schemes** (NMSS) [21]. An NMSS consists of two algorithms, Share and Reconstruct. We require that if $(\alpha, \beta) \leftarrow \text{Share}(x)$, then the marginal distributions of α and β are each independent of x , but that $\text{Reconstruct}(\alpha, \beta) = x$. The non-malleability property of the scheme is that, for all x and PPT adversaries \mathcal{A} the following probability is negligible:

$$\Pr \left[(\alpha, \beta) \leftarrow \text{Share}(x); \beta' \leftarrow \mathcal{A}(\beta, x) : \beta' \neq \beta \wedge \text{Reconstruct}(\alpha, \beta') \neq \perp \right].$$

Furthermore, an NMSS has the property that given α , β , x , and x' , where $(\alpha, \beta) \leftarrow \text{Share}(x)$, one can efficiently sample a correctly distributed β' such that $\text{Reconstruct}(\alpha, \beta') = x'$.

3 Splittability and Our Characterization

Our result is based on the alternative characterization of UC-realizability called *splittability*, introduced by Prabhakaran & Rosulek [35]. Intuitively, a two-party functionality \mathcal{F} is splittable if there is a strategy to coordinate two independent

³ In [11], this convention is also used. However, in the context of a *feasibility* result such as theirs, it is permissible (even desirable) to construct a protocol for a *stronger* version of \mathcal{F} that ignores activations from the adversary. By contrast, in a *completeness* result, we must be able to use the given \mathcal{F} as-is. Since we cannot reason about the behavior of an honest interaction if an external adversary could influence the setup, we make the requirement explicit.

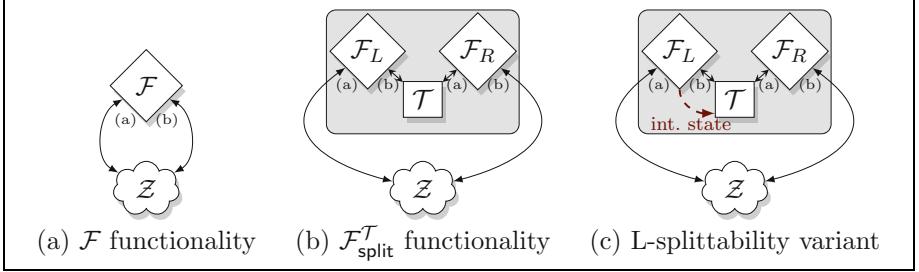


Fig. 1. Interactions considered in the splittability definitions. Small “a” and “b” differentiate a functionality’s communication links for Alice and Bob, respectively

instances of \mathcal{F} , so that together they behave as a single instance of \mathcal{F} . More formally, let \mathcal{T} be an interactive Turing machine, and define $\mathcal{F}_{\text{split}}^{\mathcal{T}}$ to be the 2-party functionality which behaves as follows (Figure 1b):

$\mathcal{F}_{\text{split}}^{\mathcal{T}}$ internally simulates two independent instances of \mathcal{F} , denoted \mathcal{F}_L and \mathcal{F}_R . $\mathcal{F}_{\text{split}}^{\mathcal{T}}$ associates its input-output link for Alice with the Alice-input/output link of \mathcal{F}_L , and similarly the Bob-input/output link with that of \mathcal{F}_R . $\mathcal{F}_{\text{split}}^{\mathcal{T}}$ also internally simulates an instance of \mathcal{T} , which is connected to the Bob- and adversary-input/output links of \mathcal{F}_L and the Alice- and adversary-input/output links of \mathcal{F}_R . \mathcal{T} receives immediate delivery along its communication lines with \mathcal{F}_L and \mathcal{F}_R . The $\mathcal{F}_{\text{split}}^{\mathcal{T}}$ functionality does not end its activation until all three subprocesses cease activating. Finally, the instances \mathcal{T} , \mathcal{F}_L , and \mathcal{F}_R are each given the global security parameter which is provided to $\mathcal{F}_{\text{split}}^{\mathcal{T}}$. We say that \mathcal{T} is **admissible** if $\mathcal{F}_{\text{split}}^{\mathcal{T}}$ is a valid PPT functionality. For an environment \mathcal{Z} , we define

$$\Delta_{\text{split}}(\mathcal{Z}, \mathcal{F}, \mathcal{T}, k) := |\text{EXEC}[\mathcal{Z}, \mathcal{F}, \pi_{\text{dummy}}, \mathcal{A}_0, k] - \text{EXEC}[\mathcal{Z}, \mathcal{F}_{\text{split}}^{\mathcal{T}}, \pi_{\text{dummy}}, \mathcal{A}_0, k]|,$$

where \mathcal{A}_0 denotes the dummy adversary that corrupts no one.

Definition 1 ([35]). Call an environment **suitable** if it does not interact with the adversary except to immediately deliver all outputs from the functionality.⁴ Then a functionality \mathcal{F} is **splittable** if there exists an admissible \mathcal{T} such that for all suitable environments \mathcal{Z} , $\Delta_{\text{split}}(\mathcal{Z}, \mathcal{F}, \mathcal{T}, k)$ is negligible in k .

Splittability provides a complete characterization of *uselessness*:

Theorem 1 ([35]). Call a functionality **useless** if it has a (non-trivial) UC-secure protocol in the plain model. Then \mathcal{F} is useless if and only if \mathcal{F} is splittable.

3.1 Our Main Theorem

Our classification is based on the following variant of splittability:

⁴ The restriction on delivering outputs is analogous to the restriction to so-called “non-trivial protocols” in [9], which is meant to rule out the protocol which does nothing. Similarly, this definition of splittability rules out the trivial splitting strategy \mathcal{T} which does nothing.

Definition 2. A functionality \mathcal{F} is **strongly unsplittable** if there exists a suitable, uniform environment \mathcal{Z} and noticeable function δ such that for all admissible \mathcal{T} , $\Delta_{\text{split}}(\mathcal{Z}, \mathcal{F}, \mathcal{T}, k) \geq \delta(k)$.

Due to technical subtleties (described in Section 4) involving the internal state of functionalities, we also consider the following variants of splittability. If in $\mathcal{F}_{\text{split}}^{\mathcal{T}}$ we let \mathcal{T} obtain the internal state of \mathcal{F}_L (resp. \mathcal{F}_R) after every activation (Figure 1c), then we obtain the notions of **L-splittability** and **L-strong-unsplittability** (resp. R-splittability, R-strong-unsplittability). We emphasize that \mathcal{T} is only allowed *read-only access* to the internal state of \mathcal{F}_L (resp. \mathcal{F}_R). In fact, most natural functionalities that are strongly unsplittable also appear to be also either L- or R-strongly-unsplittable. For example, the 3 notions are equivalent for *secure function evaluation (SFE)* functionalities — those which evaluate a (possibly randomized) function of the two parties' inputs (definitions and details deferred to the full version). As another example, the ideal commitment functionality \mathcal{F}_{com} is R-strongly-unsplittable (assuming we identify Alice as the sender), since the sender already knows the entire internal state of \mathcal{F}_{com} at all times.

Main Theorem. \mathcal{F} is complete if the SHOT assumption is true and either of the following conditions is true:

1. \mathcal{F} is L-strongly-unsplittable or R-strongly-unsplittable, or
2. \mathcal{F} is strongly unsplittable and L-splittable and R-splittable, and at least one of the \mathcal{T} machines from the L- and R-splittability conditions is uniform.

Given the equivalence of these notions for SFE functionalities, we have:

Corollary 2 If the SHOT assumption is true, and \mathcal{F} is a strongly unsplittable, SFE functionality, then \mathcal{F} is complete.⁵

3.2 Interpreting (L-/R-) Splittability and Strong Unsplittability

Nearly all functionalities of interest can be quite easily seen to be either splittable or strongly unsplittable, and thus we informally summarize our results by stating that every “natural” functionality is either useless or complete. However, there are functionalities with intermediate cryptographic power, which are neither splittable or strongly unsplittable. We give concrete examples of such functionalities in the full version, and here briefly describe properties of such functionalities:

⁵ Though similar, this corollary is somewhat incomparable to the main result of [24]. The two works use fundamentally different formulations of SFE functionalities. In ours (following our convention of Section 2.2) the functionality gives an empty output to both parties after receiving the first party's input. In [24], the functionality gives no output (except to the adversary) until receiving both party's inputs. Our result also applies to randomized functions, whereas the results of [24] involve only deterministic functionalities.

First, a functionality’s behavior may fluctuate in an unnatural way with respect to the security parameter. This may force every environment’s distinguishing probability in the splittability definition to be neither *negligible* (as required for splittability) nor *noticeable* (as required for strong unsplittability). Second, a functionality may have cryptographically useful behavior that can only be accessed by non-uniform computations, while uniform computations (such as a hypothetical commitment protocol using such a functionality) can elicit only trivial behavior. Both of these properties heavily rely on low-level technical restrictions used in the UC framework, and can easily be mitigated by relaxing these restrictions — for example, by considering notions of “infinitely-often useless/complete” or by allowing protocols to be nonuniform machines. We point out that analogous gaps also appear in other contexts involving polynomial-time cryptography [18].

Finally, as in the introduction, interpret the splittability definitions as a 2-party game between \mathcal{T} and \mathcal{Z} . Then splittability corresponds to a winning strategy for \mathcal{T} (i.e., a fixed \mathcal{T} which fools all \mathcal{Z}), and strong unsplittability corresponds to a winning strategy for \mathcal{Z} (i.e., a fixed \mathcal{Z} which detects all \mathcal{T}). Yet some functionalities may not admit winning strategies for either player. (Similarly, there may exist an environment which can distinguish \mathcal{F} from $\mathcal{F}_{\text{split}}^{\mathcal{T}}$ with noticeable probability for every \mathcal{T} , but the distinguishing bias may necessarily depend on \mathcal{T} .) An example of such a functionality is presented in the full version; however, the functionality is outside the class considered in this work (it cannot be expressed as a circuit family with *a priori* bound on input length). We do not know whether similar behavior is possible within the scope of our results.

4 UC-Equivocal Commitment from R-Strong Unsplittability

In this section we show that any R-strongly unsplittable (symmetrically, L-strongly unsplittable) functionality \mathcal{F} can be used to construct a certain kind of commitment protocol. The resulting protocol has a UC simulation only in the case where the receiver is corrupt (i.e., an equivocating simulator). Its other properties are of the standalone-security flavor. We call such a protocol a *UC-equivocal commitment*. Later in Section 6 we show that such a protocol can be transformed into a fully UC-secure protocol for commitment. From this it follows that \mathcal{F} is complete. The other case of our main theorem is simpler, similar in its techniques, and presented in Section 5.

Simplest instantiation. We first motivate the general design of the protocol with a concrete example. Suppose \mathcal{F} is a functionality which takes input x from Bob and gives $f(x)$ to Alice, where f is a one-way function. Then our UC-equivocal commitment using \mathcal{F} is as follows:

Commit phase; Alice has input b . Alice commits to b using a standalone-secure commitment protocol COM. Let C denote the commitment transcript, and let σ denote the noninteractive decommitment (known only to Alice).

Reveal phase Alice sends b to Bob. Bob chooses a random string x , and sends it to \mathcal{F} ; Alice receives $y = f(x)$. Both parties engage in a *standalone-secure protocol* for the following functionality, with common input (C, b) :

“On input (σ, z) from Alice, if σ is a valid COM-opening of C to value b , then give output z to Bob; otherwise give output $f(z)$ to Bob.”

Alice uses (σ, y) as input to this subprotocol. Bob accepts iff he receives output $f(x)$.

The protocol has a straight-line simulator for a corrupt Bob. The simulator commits to a junk value, then obtains Bob’s input x in the reveal phase. As such, it can give (\perp, x) as input to the subprotocol, and Bob will receive output $f(x)$ just as in the real interaction. Note that the subprotocol need only be standalone-secure — the simulator runs the subprotocol honestly, and in particular does not need to rewind the subprotocol.

The protocol is also binding in a standalone-security sense. Intuitively, for an equivocating Alice to succeed, she must provide an input (σ, z) to the subprotocol, where either σ is a valid COM-opening of C to $1 - b$, or $y = f(z)$. The former is infeasible by the standalone binding property of COM; the latter is infeasible because it requires Alice to invert the one-way function f .

Connection to splittability. How does this simple protocol relate to the notion of splittability? Observe that Bob’s strategy in our protocol is the obvious strategy for the *environment* when showing that \mathcal{F} is strongly unsplittable; namely, choose a random x , send it as Bob, and see whether Alice receives $f(x)$. An honest Alice and the simulator are able to succeed because they effectively “bypass” one of the two applications of f — either the one that happens within \mathcal{F} (by virtue of the simulation) or the one that happens within the subprotocol (by knowing the correct σ value). These interactions are analogous to the environment interacting with a *single* instance (not a split instance) of \mathcal{F} in the splittability game. However, a cheating Alice is “stuck” between two applications of f , analogous to the role of \mathcal{T} in the splittability game.

Generalizing to our final protocol. Following the ideas from the previous paragraph, we generalize the protocol as follows. As before, the commit phase of our final UC-equivocal protocol is essentially just a commitment under a statistically binding, standalone-secure commitment protocol (for technical reasons, it must have a rewinding extracting simulator). Again, the non-trivial part of our protocol is the reveal phase. To be UC-equivocal, the honest sender and the simulator (who can each cause the receiver to accept) must each have some *advantage* over a cheating sender (who should not be able to force the receiver to accept).

Imagine the sender and receiver both connected to two instances of \mathcal{F} , in opposite roles (similar to the splittability interaction in Figure 1b). Further imagine that the receiver is running the code of $\mathcal{Z}_{\mathcal{F}}^*$, the distinguishing environment from the definition of strong unsplittability.

Let us denote one instance of \mathcal{F} (the one associated with \mathcal{F}_L) as $\mathcal{F}_{\text{ideal}}$. In our protocol, this instance will indeed be an ideal instance of \mathcal{F} , as our protocol

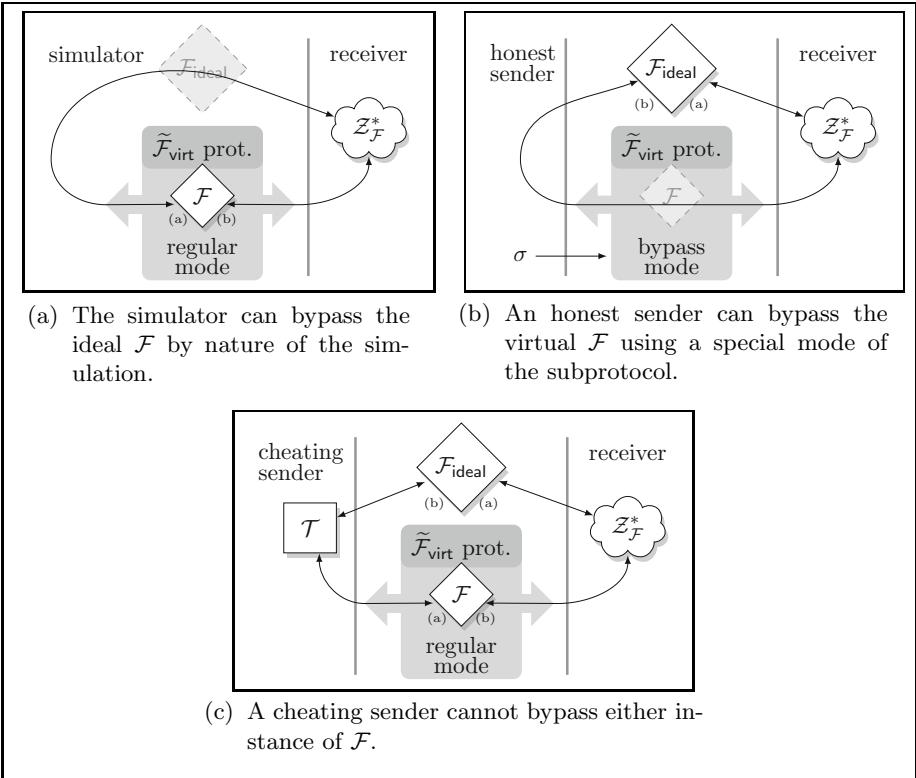


Fig. 2. Intuition behind the reveal phase of Π_{EqCom}^F and its security properties

is in the \mathcal{F} -hybrid model. As such, the simulator for a corrupt receiver is able to *bypass* this instance — by which we mean that the simulator can directly obtain the receiver’s inputs and set its outputs on behalf of $\mathcal{F}_{\text{ideal}}$. The simulator then simply “re-routes” the receiver’s connection with $\mathcal{F}_{\text{ideal}}$ directly into the second instance of \mathcal{F} . Thus, from the receiver’s point of view, $\mathcal{Z}_{\mathcal{F}}^*$ appears to be interacting with only a *single* instance of \mathcal{F} (Figure 2a).

An honest sender’s only advantage is that the underlying standalone commitment can indeed be opened to the value being claimed, whereas a cheating sender cannot generate a valid decommitment to the false value it claims. We would like to translate this advantage into a similar “bypassing” capability as the simulator. Let us denote the other instance of \mathcal{F} (the one associated with \mathcal{F}_R) as $\tilde{\mathcal{F}}_{\text{virt}}$ (virtual- \mathcal{F}), and modify it as follows. It now takes as input the commitment-phase transcript C and the purported value b . It also takes as input a candidate decommitment string σ from the sender. If σ is indeed a valid decommitment of C to b , then $\tilde{\mathcal{F}}_{\text{virt}}$ allows the sender to *bypass* just as above (directly giving the receiver’s inputs to the sender and allowing the sender to directly fix the receiver’s outputs). Otherwise, the functionality simply acts exactly like \mathcal{F} . Now the honest sender can bypass $\tilde{\mathcal{F}}_{\text{virt}}$ so that, again, from the receiver’s point of

view, $\mathcal{Z}_{\mathcal{F}}^*$ is interacting with a single instance of \mathcal{F} (Figure 2b). This advantage for the honest sender holds even if we have just a **standalone-secure** protocol for $\tilde{\mathcal{F}}_{\text{virt}}$ (importantly, since constructing a *UC-secure protocol* for $\tilde{\mathcal{F}}_{\text{virt}}$ in the \mathcal{F} -hybrid model might even be harder than our goal of constructing UC-secure commitment in the \mathcal{F} -hybrid model).

Finally, a cheating (equivocating) sender cannot provide such a value σ to $\tilde{\mathcal{F}}_{\text{virt}}$, so $\tilde{\mathcal{F}}_{\text{virt}}$ behaves just like an instance of \mathcal{F} . Thus the cheating sender can bypass neither instance and is “stuck” between two instances of \mathcal{F} (Figure 2c). The receiver’s environment $\mathcal{Z}_{\mathcal{F}}^*$ is specifically designed to detect this difference, no matter what the cheating sender does. The distinguishing bias of $\mathcal{Z}_{\mathcal{F}}^*$ is guaranteed to be noticeable, so by repeating this basic interaction a polynomial number of times $\mathcal{Z}_{\mathcal{F}}^*$ can distinguish with overwhelming probability. The receiver will therefore accept the decommitment if $\mathcal{Z}_{\mathcal{F}}^*$ believes it is interacting with instances of \mathcal{F} rather than instances of $\mathcal{F}_{\text{split}}^T$.

Technical subtleties. We outline some important technical considerations that affect the final design of our UC-equivocal commitment protocol. First, our $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol only has standalone security, so to apply any of its properties may require using a rewinding simulation. If the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol is ongoing while the parties interact with $\mathcal{F}_{\text{ideal}}$, or while the receiver is executing its $\mathcal{Z}_{\mathcal{F}}^*$ instance, then these instances may also be rewound. Since rewinding $\mathcal{Z}_{\mathcal{F}}^*$ and $\mathcal{F}_{\text{ideal}}$ would jeopardize our ability to apply the splittability condition, we let our subprotocol only perform a *single activation* of the virtual \mathcal{F} per execution. We allow \mathcal{F} to be reactive, so we need a way to maintain the internal state of the virtual- \mathcal{F} between activations of $\tilde{\mathcal{F}}_{\text{virt}}$. For this purpose we have the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol share \mathcal{F} ’s internal state between the two parties using a non-malleable secret sharing (NMSS) scheme, which was proposed for precisely this purpose [21].

The other important technicality is that activations of the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol and of $\mathcal{F}_{\text{ideal}}$ are decoupled, meaning that the receiver can observe the relative timings of these activations. This differs from the splittability interaction, in which successive activations of \mathcal{F}_L and \mathcal{F}_R within $\mathcal{F}_{\text{split}}^T$ are atomic from the environment’s perspective. This difference makes the “bypassing” technique more subtle. For instance, when the receiver gives an input to the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol, the sender must obtain this input, then make a “round trip” to $\mathcal{F}_{\text{ideal}}$ to obtain the output that it forces to the receiver through the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol. For this reason, and to avoid having the subprotocol running while $\mathcal{F}_{\text{ideal}}$ is accessed, we split such an activation (initiated by the receiver activating the virtual \mathcal{F}) of the virtual \mathcal{F} into two phases: one for input-extraction and one for output-fixing.

Similarly, consider the case where the simulator is bypassing $\mathcal{F}_{\text{ideal}}$. In the real-process interaction, every activation of $\mathcal{F}_{\text{ideal}}$ is instantaneous, so the simulator must make such activations appear instantaneous as well. In particular, the simulator has no time to make a “round-trip” to the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol to determine the appropriate response to simulate on behalf $\mathcal{F}_{\text{ideal}}$. A moment’s reflection shows that the only way for the simulator to *immediately* give the correct response is if it already knows the internal state of the virtual- \mathcal{F} . For this reason, we let the subprotocol give this information to the sender, even

in its normal (non-bypassing) mode. Since this internal state information then becomes available to a cheating sender as well, we require that \mathcal{F} be strongly unsplittable even when \mathcal{F}_R leaks its internal state in this way (i.e., R-strongly unsplittable).

We use only indistinguishability properties of the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol to prove the soundness of the straight-line equivocating simulator. Indeed, the simulation is valid even against arbitrary corrupt receivers, even though for simplicity we have portrayed here all receivers to be running $\mathcal{Z}_{\mathcal{F}}^*$ as the protocol prescribes. We use the splittability condition to prove only the (standalone) binding property of the commitment, where the receiver is honest and indeed runs $\mathcal{Z}_{\mathcal{F}}^*$.

Lemma 1. *If the SHOT assumption is true, then $\Pi_{\text{EqCom}}^{\mathcal{F}}$ has a UC simulator in the case that the receiver is corrupt (i.e., an equivocating simulator). If \mathcal{F} is R-strongly-unsplittable, then $\Pi_{\text{EqCom}}^{\mathcal{F}}$ has a rewinding simulator in the case that the sender is corrupt (i.e., an extracting simulator).*

5 UC-Equivocal Commitment from L/R-Splittability

In this section we show that if \mathcal{F} is strongly unsplittable, L-splittable, and R-splittable, then \mathcal{F} can be used to construct a UC-equivocal commitment protocol. This is the second case in our main theorem.

5.1 Overview

Our approach for this case is quite similar to that of Section 4 — our protocol involves an ideal instance of \mathcal{F} along with a “virtual” instance of \mathcal{F} within a standalone-secure subprotocol. The receiver runs instances of $\mathcal{Z}_{\mathcal{F}}^*$, the environment guaranteed by the strong unsplittability condition. The receiver accepts the commitment if $\mathcal{Z}_{\mathcal{F}}^*$ believes it is interacting with a single instance of \mathcal{F} as opposed to some $\mathcal{F}_{\text{split}}^T$. The primary difference from the previous section is in the “bypass mode” of the virtual- \mathcal{F} subprotocol. In this subprotocol, the bypass mode does not completely bypass the virtual \mathcal{F} , but instead it simply leaks the internal state of the virtual \mathcal{F} to the receiver. Intuitively, leaking the internal state is sufficient to “fool” $\mathcal{Z}_{\mathcal{F}}^*$, since \mathcal{F} is L/R-splittable. We have the following observations about the protocol (Figure 3):

- An honest sender who can activate the bypass mode of the $\tilde{\mathcal{F}}_{\text{virt}}$ subprotocol is situated between an ideal instance of \mathcal{F} and (intuitively) an instance of \mathcal{F} that leaks its internal state. By the R-splittability of \mathcal{F} , there exists a \mathcal{T}_2 that the sender can execute to make two such instances behave to the sender as a single instance of \mathcal{F} . Note that \mathcal{T}_2 must be a uniform machine, since it is used as a subroutine in the description of the protocol.
- The simulator can honestly simulate $\mathcal{F}_{\text{ideal}}$ while also having access to its internal state. The remainder of the simulator is intuitively between this simulated instance of \mathcal{F} and a (normal, non-bypassed) instance of \mathcal{F} . By the

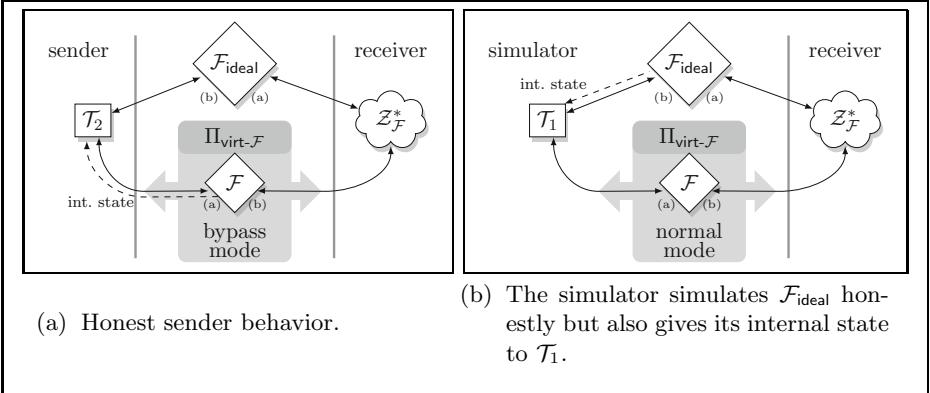


Fig. 3. Interactions in $\Pi_{\text{EqCom}}^{\mathcal{F}}$ and its security proof

L-splittability of \mathcal{F} , there exists a \mathcal{T}_1 that the simulator can execute to make these two instances behave to the sender as a single instance of \mathcal{F} . Note that \mathcal{T}_1 need not be uniform, since it is used only by the simulator. Finally, since the simulator is designed to interact with a corrupt receiver, \mathcal{T}_1 and \mathcal{T}_2 must be able to “fool” every environment, not just the environment $\mathcal{Z}_{\mathcal{F}}^*$ from the strong unsplittability condition.

- A cheating sender cannot obtain the internal state from either the ideal or the virtual instance of \mathcal{F} . As such, it plays the role of the machine \mathcal{T} in the (normal) splittability interaction. By the strong unsplittability of \mathcal{F} , the receiver’s environment $\mathcal{Z}_{\mathcal{F}}^*$ can detect a noticeable deviation from the expected behavior.

In this section, we never have need to completely bypass an instance of \mathcal{F} . The technical complications described in Section 4 are not present here, and the actual construction is a much more straight-forward implementation of the intuition described above.

Lemma 2. *If the SHOT assumption is true and \mathcal{F} is L- & R-splittable, then $\Pi_{\text{EqCom}}^{\mathcal{F}}$ has a UC simulator in the case that the receiver is corrupt (i.e., an equivocating simulator). If \mathcal{F} is strongly-unsplittable, then $\Pi_{\text{EqCom}}^{\mathcal{F}}$ has a rewinding simulator in the case that the sender is corrupt (i.e., an extracting simulator).*

6 Full-Fledged UC Commitment from Equivocal Commitment

We now show how the UC-equivocal commitment protocol from the previous sections can be used to construct a full-fledged UC commitment protocol. Due to space limitations, we give only a high-level overview of the protocol $\Pi_{\text{com}}^{\mathcal{F}}$ here; the full details and security proof are deferred to the full version.

Commit phase:

1. The receiver first uses $\Pi_{\text{EqCom}}^{\mathcal{F}}$ to commit to a random string r containing half 0s and half 1s.
2. To commit to a bit b , the sender uses $\Pi_{\text{EqCom}}^{\mathcal{F}}$ to commit (bitwise) to a random string x , and gives $b \oplus (\bigoplus_i x_i)$ to the receiver.
3. The two parties engage in a standalone-secure subprotocol in which the receiver learns a random half of the bits of x . The sender does not learn which positions of x the receiver obtains.
4. The receiver opens his commitment to r . The receiver opens her commitments to all bits x_i such that $r_i = 1$.

Intuitively, a malicious receiver can expect to learn no more than about 3/4 of the bits of x in this way (a random half in step 3 and an independent random half in step 4), so the secret bit b remains hidden. However, the simulator for a corrupt sender can equivocate (in $\Pi_{\text{EqCom}}^{\mathcal{F}}$) while revealing r so that it learns *all* the bits of x , and hence can extract b .

Reveal phase: The parties engage in a standalone-secure subprotocol in which the sender learns a position i^* for which the receiver did not learn the bit x_{i^*} in steps 3 and 4 the commit phase. Then the sender opens all the remaining commitments to the bits of x .

Intuitively the information i^* is useless to a corrupt sender, but the simulator for a corrupt receiver can equivocate (in $\Pi_{\text{EqCom}}^{\mathcal{F}}$) on its opening of the bit x_{i^*} , and hence open its commitment to either value of b .

Lemma 3. $\Pi_{\text{com}}^{\mathcal{F}}$ is a UC-secure protocol for commitment, in the \mathcal{F} -hybrid model.

7 Necessity of SHOT Assumption and Strong Unsplittability

The SHOT assumption is necessary for many strongly unsplittable functionalities (e.g., coin-tossing, commitment) to be complete under static corruption [14,31]. Thus the SHOT assumption is the minimal assumption for a completeness result such as ours.

In this work we showed that strong unsplittability (and its variants) is a sufficient condition for completeness. Ideally, we would like to prove that it is also a necessary condition; currently we are able to prove that several minor modifications of strong unsplittability are necessary.

To prove necessity of some kind of strong unsplittability, we show that \mathcal{F} is not complete by showing that there is no UC-secure protocol for coin-tossing in the \mathcal{F} -hybrid model. Consider an interaction involving a purported coin-tossing protocol in the \mathcal{F} -hybrid model. This protocol invokes possibly many instances of \mathcal{F} , and it is likely that we will have to apply some property of \mathcal{F} to each of them (indeed, this is what we must do in the proofs below). If \mathcal{F} is not strongly unsplittable, then for every suitable \mathcal{Z} , there is a machine \mathcal{T} so that

$\Delta_{\text{split}}(\mathcal{Z}, \mathcal{F}, \mathcal{T}, k)$ is non-noticeable — that is, negligible only *for infinitely many* values of the security parameter. In the following proofs, we must apply this condition in a hybrid argument, each time with a slightly different environment and thus a potentially different subset of security parameter values. There may not be an infinite number of security parameter values for which *every* step of the hybrid argument succeeds, and thus we must settle for slight variants of strong unsplittability in the following:

Lemma 4. *If \mathcal{F} is complete, then all of the following are true:*

1. \mathcal{F} is infinitely-often strongly unsplittable with respect to uniform \mathcal{T} .
2. The multi-session version of \mathcal{F} is strongly unsplittable.
3. \mathcal{F} is strongly unsplittable via an environment with multi-bit output.

In item (1), “infinitely-often” refers to the relaxation in which $\Delta_{\text{split}}(\mathcal{Z}, \mathcal{F}, \mathcal{T}, k)$ is non-negligible (rather than noticeable, as required in the usual definition).

The complete proofs are given in the full version, but for a flavor of the techniques we give a proof of item (1) here:

Proof (of item 1). We prove the contrapositive. Suppose that \mathcal{F} is **not** as in item (1), then for every suitable \mathcal{Z} there is a *uniform* machine \mathcal{T} so that $\Delta_{\text{split}}(\mathcal{Z}, \mathcal{F}, \mathcal{T}, k)$ is negligible. It suffices to show that there is no protocol for coin-tossing in the \mathcal{F} -hybrid model.

Suppose for contradiction that π is such a secure protocol for coin-tossing. Let \mathcal{Z} be the environment that invokes one session of coin tossing and outputs 1 if both parties output the same coin. Then we inductively define sequences of machines \mathcal{T}_i and \mathcal{Z}_i as follows. \mathcal{Z}_i is the environment that internally simulates \mathcal{Z} and two honest parties running π . For $j > i$, the j th instance of \mathcal{F} invoked by π is simulated internally to \mathcal{Z}_i . The i th instance of \mathcal{F} is routed to an external ideal instance of \mathcal{F} . And for $j < i$, the j th instance of \mathcal{F} is simulated internally as $\mathcal{F}_{\text{split}}^{\mathcal{T}_j}$ rather than \mathcal{F} . The machine \mathcal{T}_i is defined as the uniform machine that “fools” environment \mathcal{Z}_i . By a straight-forward hybrid argument, we have that \mathcal{Z} outputs 1 with overwhelming probability when invoking an instance of π in which the i th instance of \mathcal{F} is replaced by $\mathcal{F}_{\text{split}}^{\mathcal{T}_i}$.

By applying the UC-security of π , we can replace Alice (who is honestly running π) and the collection of \mathcal{F}_L instances in this interaction with an ideal instance of coin-tossing (and appropriate simulator). Similarly, we can replace Bob and the set of \mathcal{F}_R instances in the interaction with another ideal instance of coin-tossing. Both of these changes have only a negligible effect on the outcome of the interaction, by the security of π . Now \mathcal{Z} receives two coins from *totally independent* instances of an ideal coin-tossing functionality, and outputs 1 with overwhelming probability. Since \mathcal{Z} only outputs 1 if its two inputs agree, this is a contradiction. Thus, no such protocol π can exist.

Acknowledgements. The author would like to thank Tal Malkin for helpful discussions surrounding Lemma 4, and Manoj Prabhakaran, Hong-Sheng Zhou, and several anonymous referees for various constructive suggestions.

References

1. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195. IEEE Computer Society (2004)
2. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552. IEEE Computer Society (2005)
3. Beimel, A., Malkin, T., Micali, S.: The All-or-Nothing Nature of Two-Party Secure Computation. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 80–97. Springer, Heidelberg (1999)
4. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Naor, M. (ed.) FOCS, pp. 136–145. IEEE Computer Society (2001), Revised version on Cryptology ePrint Archive (2005), <http://eprint.iacr.org/2000/067>
6. Canetti, R.: Obtaining Universally Composable Security: Towards the Bare Bones of Trust. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 88–112. Springer, Heidelberg (2007)
7. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Security with Global Setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
8. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
9. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology* 19(2), 135–167 (2006)
10. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: Trevisan, L. (ed.) FOCS, pp. 541–550. IEEE Computer Society (2010)
11. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503. ACM (2002)
12. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: FOCS, pp. 249–259. IEEE Computer Society (2007)
13. Chor, B., Kushilevitz, E.: A zero-one law for boolean privacy. *SIAM J. Discrete Math.* 4(1), 36–47 (1991)
14. Damgård, I., Nielsen, J.B., Orlandi, C.: On the Necessary and Sufficient Assumptions for UC Computation. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 109–127. Springer, Heidelberg (2010)
15. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229. ACM (1987)
16. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: STOC, pp. 291–304. ACM (1985)
17. Groth, J., Ostrovsky, R.: Cryptography in the Multi-string Model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
18. Harnik, D., Naor, M., Reingold, O., Rosen, A.: Completeness in two-party secure computation: A computational view. *J. Cryptology* 19(4), 521–552 (2006)

19. Hofheinz, D., Unruh, D., Müller-Quade, J.: Polynomial runtime and composability. Cryptology ePrint Archive, Report 2009/023 (2009), <http://eprint.iacr.org/2009/023>
20. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography (extended abstract). In: FOCS, pp. 230–235. IEEE (1989)
21. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
22. Kalai, Y.T., Lindell, Y., Prabhakaran, M.: Concurrent general composition of secure protocols in the timing model. In: Gabow, H.N., Fagin, R. (eds.) STOC, pp. 644–653. ACM (2005)
23. Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
24. Katz, J., Kiayias, A., Kumaresan, R., Shelat, A., Zhou, H.-S.: From impossibility to completeness for deterministic two-party SFE (2011) (unpublished manuscript)
25. Kidron, D., Lindell, Y.: Impossibility results for universal composability in public-key models and with fixed inputs. *J. Cryptology* 24(3), 517–544 (2011)
26. Kilian, J.: More general completeness theorems for secure two-party computation. In: STOC, pp. 316–324. ACM (2000)
27. Kilian, J., Kushilevitz, E., Micali, S., Ostrovsky, R.: Reducibility and completeness in private computations. *SIAM J. Comput.* 29(4), 1189–1208 (2000)
28. Kreitz, G.: A Zero-One Law for Secure Multi-party Computation with Ternary Outputs. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 382–399. Springer, Heidelberg (2011)
29. Lin, H., Pass, R., Venkatasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: Mitzenmacher, M. (ed.) STOC, pp. 179–188. ACM (2009)
30. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
31. Maji, H.K., Prabhakaran, M., Rosulek, M.: A Zero-One Law for Cryptographic Complexity with Respect to Computational UC Security. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 595–612. Springer, Heidelberg (2010)
32. Malkin, T., Moriarty, R., Yakovenko, N.: Generalized Environmental Security from Number Theoretic Assumptions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 343–359. Springer, Heidelberg (2006)
33. Naor, M.: Bit commitment using pseudorandomness. *J. Cryptology* 4(2), 151–158 (1991)
34. Pass, R.: Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
35. Prabhakaran, M., Rosulek, M.: Cryptographic Complexity of Multi-Party Computation Problems: Classifications and Separations. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 262–279. Springer, Heidelberg (2008)
36. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: Babai, L. (ed.) STOC, pp. 242–251. ACM (2004)

Impossibility Results for Static Input Secure Computation

Sanjam Garg¹, Abishek Kumarasubramanian¹,
Rafail Ostrovsky¹, and Ivan Visconti²

¹ UCLA, Los Angeles, CA

{sanjamg,abishekk,rafael}@cs.ucla.edu

² University of Salerno, Italy

visconti@dia.unisa.it

Abstract. Consider a setting of two mutually distrustful parties Alice and Bob who want to securely evaluate some function on *pre-specified* inputs. The well studied notion of *two-party secure computation* allows them to do so in the *stand-alone* setting. Consider a deterministic function (e.g., 1-out-of-2 bit OT) that Alice and Bob can not evaluate trivially and which allows only Bob to receive the output. We show that Alice and Bob can not securely compute any such function in the *concurrent* setting even when their inputs are *pre-specified*. Our impossibility result also extends to all deterministic functions in which both Alice and Bob get the same output. Our results have implications in the *bounded-concurrent* setting as well.

1 Introduction

Consider a setting of two mutually distrustful parties Alice and Bob who want to securely evaluate a function f . The well studied notion of *two-party secure computation* [1, 2] allows them to do so. However this notion is only relevant to the *stand-alone* setting where security holds only if a single protocol session is executed in isolation. Additionally these secure computation protocols are *interactive* and Alice and Bob are expected to preserve *state* information during the protocol execution.

What if Alice and Bob want to evaluate multiple functions concurrently? This problem has drawn a lot of attention in the literature. A large number of secure protocols (in fact under an even stronger notion of security called UC security) based [3–12] on various trusted setup assumptions have been proposed. To address the problem of concurrent security for secure computation in the *plain model*, a few candidate definitions have been proposed, including input-indistinguishable security [13, 14] and super-polynomial simulation [15–17, 9, 18]. Both of these notion, although very useful in specialized settings, *do not* suffice in general. Additionally other models that limit the level of concurrency have also been considered [19, 20] or allow simulation using additional outputs from

the ideal functionality [21]. Among these models the model of m -bounded concurrency [22, 19] which allows for m different protocol executions to overlap has received a lot of attention in the literature [22, 19, 23, 24]. Unbounded concurrent oblivious transfer in the restricted model where all the inputs in all the executions are assumed to be independent has been constructed [25].

At the same time, impossibility results ruling out the existence of secure protocols in the concurrent setting have been shown. UC secure protocols for most functionalities of interest have been ruled out in [3, 26]. Concurrent self-composability [23] for a large class of interesting functionalities (i.e., bit transmitting functionalities) has been ruled out however only in a setting in which the honest parties choose their inputs *adaptively* (i.e., “on the fly”). Unfortunately, in the very natural setting of *static* (pre-specified) inputs only the \mathcal{F}_{ZK+OT} functionality (i.e., a mix of the zero-knowledge and the oblivious transfer functionalities) and the functionality that evaluates a pseudorandom function on a committed key [20] have been ruled out.

This leaves a large gap between the very few functionalities (e.g., $ZK + OT$) for which the impossibility [27, 20] has been proved, and the achieved results on concurrent ZK [28–31, 27]. In this paper, we ask the following *basic* question:

What functions can Alice and Bob concurrently securely compute in the plain model in the natural setting of static (i.e., pre-specified) inputs?

1.1 Our Results

In this paper we give the following two results.

Impossibility Results for Concurrent Self Computation. We show (assuming one-way functions) that no two-party protocol concurrent securely realizes any symmetric (where both parties get the same output) or asymmetric (only one party gets the output) deterministic functionality [32–34] that is *complete*¹ in the stand-alone setting. Our impossibility results hold even in the *very restricted* setting of *static* inputs (inputs of honest parties are pre-specified) and *fixed roles* (i.e, the adversary can corrupt only one party who plays the same role across all executions).

We additionally show that an m -bounded concurrently secure protocol for any symmetric deterministic functionality must have a communication complexity of at least $\frac{m}{k^c}$ bits where $c \geq 0$ (depending on the functionality) is a constant and k is the security parameter. This bound corresponds to m bits for the case of string OT.

Independent of our work, exciting results concerning the impossibility of concurrently secure computation have been obtained recently by Agrawal et. al. We refer the reader to [35], in these proceedings, for more details.

¹ A functionality is said to be complete if it can be used to securely realize any other functionality.

Impossibility Results for Stateless Two-party Secure Computation. The question of *general* secure computation with stateless parties has been studied in [36, 37]. It might seem that secure computation with stateless parties should imply secure computation in the concurrent but stateful setting. However interestingly, this is not true as the definition of secure computation among stateless parties *only* requires the existence of a simulator that can additionally benefit from the stateless nature of honest parties (technically speaking, the ideal-world adversary can rewind the functionality). Goyal and Maji [37] showed a positive result for stateless secure computation for a large class of (in particular for all functionalities except the ones which behave as a weak form of pseudorandom generators) functionalities. In this work, we show unconditionally that there exists a functionality that can not be securely realized between two stateless parties. Similar results have been obtained concurrently and independently by Goyal and Maji [38].

1.2 Technical Overview - Impossibility of Concurrent Self Composition

Consider two parties Alice and Bob executing a two-party secure computation protocol. Now consider a real-world adversary that corrupts either Alice or Bob and is allowed to participate in any arbitrary (still polynomial) number of executions of the protocol. In this setting we construct a real-world adversary that interacts with the honest party in an execution of the protocol, referred to as the *main* execution that can not be simulated in the ideal world. The starting point for realizing such an adversary is the idea that the adversary has secure computation at its disposal and it can use it to its advantage. More specifically, an adversary can at will interact with the honest party in multiple *additional* executions of the secure computation protocol and use the “information” obtained in these additional executions to complete the *main* execution. In order to realize the impossibility we need to establish what this “information” is and how can this be obtained. Looking ahead this “information” is going to be the very messages that the adversary needs to send in the main execution and it is going to be obtained via secure computation with the honest party. In this paper we extend this intuition to show impossibility of concurrent secure computation for a very general class of functionalities. However, in order to build intuition we start by considering the impossibility for the special case of 1-out-of-2 string oblivious transfer (OT).

String OT. Consider two parties Alice and Bob executing an instance of the string OT protocol in which Alice plays the role of the sender and Bob plays the role of the receiver. We refer to this execution as the *main* execution. Now in order to complete the main execution adversary needs to execute some function securely with Alice.

First we begin by addressing the issue of providing Bob with the ability to execute some function exactly once. Observe that providing malicious Bob with a garbled circuit allows him to evaluate (once) any function securely as long as

he can obtain the right secret keys for evaluating the garbled circuit. However obtaining such keys is not a problem as malicious Bob has access to a string OT channel with Alice. More specifically, if we set garbled circuit keys as the inputs of Alice then malicious Bob can obtain the keys of his choice and evaluate any function securely. Very roughly we have established that malicious Bob can evaluate any function securely allowing it with an access to a source of “information” of its choice.

The next question to ask is “What is the right information?” Observe that malicious Bob needs to send messages as the receiver in the main execution of the protocol.² Now note that the “information” that malicious Bob receives may very well be the very messages that it needs to send to Alice in the main execution. Bob obtains the messages it sends to Alice using secure evaluation. This allows us to argue that no ideal-world simulator can simulate the view of this “virtual” receiver. In arguing this we crucially rely of the fact that the ideal-world simulator is limited and can obtain *only* one key corresponding to each input wire of the garbled circuit. Additionally, we would like to stress *two* points:

- Observe that malicious Bob obtains the keys from Alice in a very straightforward and well specified manner. On the other hand a simulator trying to simulate malicious Bob is not required to follow this strategy. In particular the simulator could potentially obtain keys in an *adaptive* manner that depends on the garbled circuit it gets as input. To deal with this problem, we use a construction of Yao’s garbled circuit secure against *malicious* adversaries [39].
- Since we are in the *static* input setting all the inputs of Alice must be specified before any execution of the protocol is started. In our setting the inputs of the honest Alice consist of the keys for the garbled circuit, and can always be specified before the execution of the protocol. On the other hand malicious Bob can follow any arbitrary polynomial-time strategy. In particular the adversary can choose its inputs adaptively.

Bit OT. The intuition described above crucially relies on the fact that the adversary can execute multiple instances of the string OT protocol in order to obtain the desired keys for input wires of the garbled circuit and thereby evaluate the circuit. Note that the adversary described in the setting of string OT will continue to work when provided with an access to a protocol that can be used to evaluate bit OTs. However unfortunately, we will not be able to rule out simulators that obtain potentially different bits of the key strings. We solve this problem using the string OT construction from bit OT of [40]. The key idea of the construction in [40] is to “encode” the keys of the garbled circuit in a manner such that the partial information obtained by any simulator on the encoded keys is essentially “useless.” An important feature of the [40] technique is that it allows us to specify inputs of Alice before any bit OT protocol is executed.

² Our proof builds upon the intuitive application of *chosen protocol attack* as in [27].

Extending to general functionalities. In extending the impossibility result to general functionalities the challenge lies in realizing a form of bit OT but without the use of adaptive inputs for honest parties. The key idea in achieving this is to exploit the “uncertainty” in the input of honest Alice that must exist in the eyes of any simulator that is constrained to keep the outputs of the real-world execution and the ideal-world execution indistinguishable. In the case of *symmetric* functionalities (which are complete in the stand-alone malicious setting) the outputs of honest Alice plays a crucial role in ensuring this. In particular in order to make sure that Alice gets the correct output a simulator simulating malicious Bob is forced into leaving some “uncertainty” in the input of honest Alice. On the other hand *asymmetric* functionalities (again, which are complete in the stand-alone malicious setting) do not provide any output to honest Alice but posses a more general structure. In particular these functionalities have a structure that leaves some “uncertainty” in the input of the honest Alice regardless of the input of malicious Bob. This allows us to argue our impossibility result.

1.3 Technical Overview - Impossibility of Stateless Two-Party Computation

Consider three stateless parties Alice, Bob and Charlie executing two instances of a two-party secure computation protocol. Malicious Bob³ interacts with an honest Alice in the first instance and with an honest Charlie in the second instance. In this setting malicious Bob can evaluate any function securely with Charlie and use it as source of “information.” Just like in the concurrent security case we can ask the question: “What is the right information?” Observe that the adversary needs to send messages in the execution of the protocol with Alice. Now note that the “information” that malicious Bob receives from Charlie may very well be these next messages that it needs to send to Alice. This allows us to construct a “virtual” party that malicious Bob securely implements with the help of honest Charlie. Now observe that a simulator might have non-black box access to malicious Bob, however, this is essentially useless because malicious Bob acts just like a “message forwarding machine.” Therefore simulator only has black-box access to malicious Bob which alone of course does not help the simulator because a malicious Bob on being rewound can always reset honest Charlie as it is stateless. Therefore we can conclude that no ideal-world adversary can simulate the view of malicious Bob because an ability to do so will contradict the security of the underlying protocol itself.

2 Preliminaries and Definitions

Let k denote a security parameter. We say that a function is *negligible* in the security parameter k if it is asymptotically smaller than the inverse of any fixed polynomial. Otherwise, the function is said to be *non-negligible* in k . We say that

³ We stress that a malicious Bob is not required to be stateless.

an event happens with *overwhelming* probability if it happens with a probability $p(k) = 1 - \nu(k)$ where $\nu(k)$ is a negligible function of k . In this section, we recall the definitions of basic primitives studied in this paper. Let $\stackrel{c}{\equiv}$ stand for computational indistinguishability of two distributions.

Our definitions of concurrent security are judiciously borrowed from the work of Lindell [24]. Some of the text has been taken verbatim from [24], however, the definitions have been tailored and simplified to suit our requirements.

Two-Party Computation. A two-party protocol problem is cast by specifying a function that maps pairs of inputs to pairs of outputs (one input/output for each party). We refer to such a map as a *functionality* and denote it by $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{0, 1\}^*$, where $f = (f_1, f_2)$, i.e. for every pair of inputs (x, y) , we have a pair of outputs $(f_1(x, y), f_2(x, y))$. Thus for example, the oblivious transfer functionality is denoted by $\mathcal{F}_{OT}((m_0, m_1), b) = (\perp, m_b)$.

In the context of concurrent composition each party actually uses many inputs. These are represented by vectors of inputs \bar{x}, \bar{y} for the two parties. Furthermore, let $\ell = |\bar{x}| = |\bar{y}|$ denote the number of sessions the two parties interact in. In this work we consider the setting of *static* inputs as opposed to *adaptive* inputs. More specifically in the setting of static inputs, the inputs of both parties are specified before the execution of any protocol. This differs from the more general setting of adaptive inputs, in which honest parties choose inputs for sessions on the fly (possibly using the information obtained in previous sessions).

Adversarial Behavior. Throughout this paper we only consider a malicious and static adversary. In a two-party interaction, such an adversary chooses one of the parties before the protocol begins (who is referred to as a corrupted party) and may then interact with the honest party on behalf of the corrupted party while arbitrarily deviating from the specified protocol. Furthermore, in this work, we consider secure computation protocols with *aborts* and no *fairness*. This notion is well studied in literature [41]. More specifically, the adversary can abort at any point and the adversary can decide when (if at all) the honest parties will receive their output even when it receives its own output. The scheduling of the message delivery is decided by the adversary.

Note that we study the security of the protocols in a setting where multiple instances of a two-party protocol (between P_1 and P_2) are being executed. In order to obtain stronger impossibilities, we look at the setting in which the same party plays the role of P_1 (and similarly P_2) across all executions of the protocol. We refer to this as the setting of *fixed roles*. In this setting, an adversary can corrupt only one party that consistently plays the role of P_1 (or P_2) across all sessions.

Observe that we consider a very restricted adversary (in terms of what it can do) and provide impossibility results in this paper. Furthermore, all the impossibility results in this setting directly translate to more demanding settings.

Security of Protocols (Informal). We give an informal description of the definition here and refer the reader to the full version for formal definitions. We start by giving the definition in the *stand-alone* case, in which the parties execute only

one instance of the protocol. The security of a protocol is defined by comparing what an adversary can do in the real-world execution of the protocol to what it could have done in an ideal scenario. The real-world execution of the protocol corresponds to the actual interaction of the adversary with honest parties. The real execution of ρ (with security parameter k , initial inputs (x, y) , and auxiliary input z to the adversary \mathcal{A}), denoted $\text{EXEC}_{\rho, \mathcal{A}}(k, x, y, z)$, is defined as the output pair of the honest party and \mathcal{A} , resulting from the above process. The ideal scenario is formalized by considering an ideal incorruptible trusted third party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Then the **ideal execution** of f (with security parameter k , initial inputs (x, y) and auxiliary input z to \mathcal{S}), denoted by $\text{EXEC}_{f, \mathcal{S}}(k, x, y, z)$ is denoted as the output pair of the honest party and the adversary \mathcal{S} from the above execution. Informally, we require that executing a protocol in the real world roughly emulates the ideal process.

Next we extend the definition to the concurrent setting. Unlike in the case of stand-alone computation, in the concurrent setting the real-world protocol can be executed multiple times. Correspondingly, in the ideal world, the trusted party computes the functionality many times, each time upon the received inputs. Loosely speaking, a protocol is secure if any adversary interacting in the real-world protocol (where no trusted third party exists) can do no more harm than if it was involved in a corresponding ideal computation, as described above.

Bounded Concurrency. The notion of concurrent self composition allows for an unbounded (though, still polynomial) number of executions of a protocol. We sometimes refer to this notion of concurrent self composition as *unbounded* concurrency, in order to distinguish it from m -bounded concurrency that we describe next. In the setting of m -bounded concurrency, the number of concurrent executions is a priori bounded by m (a fixed polynomial). More specifically, we require that in the interval between the beginning and termination of any given execution, the number of different sessions executed is at most m . Furthermore, the protocol design and hence the running time of each party (in a single session) can depend on this bound.

Resetability. The notion of resetably secure computation [36, 37] is also defined by comparing what an adversary can do in the real-world execution of the protocol to what it could have done in an ideal scenario, however there are two crucial differences. The real-world adversary in the resettable setting is allowed to reset (or, “rewind”) honest parties. In a similar way, the ideal-world adversary in the resettable setting can query the ideal functionality on behalf of the adversary multiple times, each time with a different input of its choice (while the input of the honest party remains the same).

3 Impossibility of Static Input Concurrency

In this section we prove that the string OT functionality can not be concurrently and securely realized even in the setting of *static* inputs and against adversaries

that corrupt only one party that plays a *fixed role*. Next we generalize this and provide a general theorem allowing us to argue impossibility for a broader class (we do this in Section 4) of functionalities.

3.1 The Case of String OT

In this section we start by first giving an impossibility result for the string OT functionality. Roughly speaking, string OT is a two-party functionality between a Sender S , with input (m_0, m_1) and a Receiver R with input b which allows R to learn m_b without learning anything about m_{1-b} . At the same time the Sender S learns nothing about b . More formally string OT functionality $\mathcal{F}_{OT} : (\{0, 1\}^{p(k)} \times \{0, 1\}^{p(k)}) \times \{0, 1\} \rightarrow \{0, 1\}^{p(k)}$ is defined as, $\mathcal{F}_{OT}((m_0, m_1), b) = m_b$, where $p(\cdot)$ is any polynomial and only R gets the output. We show that for some polynomial $p(\cdot)$ (to be fixed later), there does not exist a protocol π that concurrently securely realizes the \mathcal{F}_{OT} functionality. More specifically we show that there exists an adversary \mathcal{A} who starts $\ell(k)$ sessions (to be fixed later) of the protocol π with honest parties (with pre-specified inputs drawn from a particular distribution \mathcal{D}) such that no ideal-world adversary whose output is computationally indistinguishable from the output of real-world adversary \mathcal{A} exists. We stress that the adversary (we construct) corrupts only one of the two parties – the Sender S or the Receiver R in all the $\ell(k)$ sessions. In other words we are in the setting of *fixed roles* (and our results of course extend also to the setting where the adversary plays different roles).

Theorem 1. (*impossibility of static input concurrent-secure string OT*) *Let π be any protocol which implements⁴ the \mathcal{F}_{OT} functionality for a particular (to be determined later) polynomial $p(k)$. Then, (assuming one-way functions exist) there exists a polynomial $\ell(k)$ and a distribution \mathcal{D} over $\ell(k)$ -tuple of inputs and an adversarial strategy \mathcal{A} such that for every probabilistic polynomial-time simulation strategy \mathcal{S} , definition of concurrent security cannot be satisfied when the inputs of the parties are drawn from \mathcal{D} .*

Proof Intuition. We start by giving an informal outline of the proof. We postpone the full proof to the full version.

1. **Setting Up a Chosen Protocol Attack:** As stated earlier our goal is to construct a real-world adversary that can achieve something in the real world that is infeasible for any ideal-world adversary to accomplish in the ideal world. The starting point for realizing such an adversary is to consider a *chosen protocol attack* [42, 43, 27] and adapt it to our setting. Let us start by considering a protocol π that concurrently securely realizes the \mathcal{F}_{OT} functionality. The key attack methodology of chosen protocol attack (first considered in the context of zero-knowledge protocols) is to consider a specific protocol π' (which may depend on π and hence the name “chosen protocol

⁴ We say that a protocol implements a functionality if the protocol allows two parties to evaluate the desired function. This protocol however may not be secure.

attack") which when concurrently executed with π renders π completely insecure. Next we present the chosen protocol attack in our specific setting. Consider a setting with four parties $A, \tilde{B}, \tilde{C}, D$. (Looking ahead parties \tilde{B} and \tilde{C} will be corrupted by the adversary.) In this setting A and \tilde{B} execute an instance of the OT protocol π where A plays the role of the Sender (with inputs say (m_0, m_1)) and \tilde{B} plays the role of the Receiver (with some choice bit c as input). At the same time, parties \tilde{C} (with inputs say (m'_0, m'_1)) and D (with a bit $b \in \{0, 1\}$ and values m_b, w as input) execute the following chosen protocol π' (which depends on π) and is meant to render π insecure. We now describe this protocol π' . π' involves first an execution of π between \tilde{C} and D in which \tilde{C} plays the role of the sender (with inputs say (m'_0, m'_1)) and D plays the role of the receiver. Now note that D playing the role of the receiver obtains the value m'_b in the execution of π . D now checks if $m'_b = m_b$ and sends w to \tilde{C} if this is indeed the case. Observe that in the above setting A is provided with (m_0, m_1) as input and D is provided with b and the corresponding m_b as input.

Next we show how concurrent execution of protocol π with the protocol π' renders π insecure. Towards this goal, consider a real-world adversary \mathcal{A} that corrupts \tilde{B} and \tilde{C} , ignores their inputs and proceeds as follows⁵. Our adversary \mathcal{A} merely acts as an intermediary who forwards the messages he receives from honest A to honest D on behalf of \tilde{C} and similarly forwards the messages it receives from honest D to honest A on behalf of \tilde{B} ⁶. Our adversary \mathcal{A} will be able to complete the execution of π with Sender A and the execution of π (executed as a part of π') with receiver D . This setting roughly amounts to A and D executing a protocol π in which A has inputs (m_0, m_1) and D has input b . Additionally in this execution, D will always obtain the value m_b which will match its input m_b and therefore it will always send the value w to \mathcal{A} . Our adversary outputs this value w as its output. On the other hand, executing merely an ideal version of π and obtaining its output will not help \tilde{B}, \tilde{C} to successfully complete an interaction with D and thus the adversary does not output w . This results in breaking the indistinguishability between the real world and the ideal world, contradicting the security definition of π .

Now that, we have described an adversarial strategy in a four party scenario where \mathcal{A} interacts in an execution of π and an execution of π' . We would like to move to a setting with only concurrent executions of the protocol π .

⁵ Recall that we are in setting of static inputs. This setting requires that the inputs of the honest parties be pre-specified before any execution of the protocol happens. On the other hand our adversary \mathcal{A} may deviate arbitrarily from the protocol specification based on his input and auxiliary input. In particular, it could adaptively decide on the messages it sends in the protocol ignoring the input it obtains completely.

⁶ We remark that in case the underlying protocol uses identities, then we will think of \tilde{C} as using the identity of A and D as using the identity B . As we will see later, this continues to work in our setting.

2. **Using Yao [1] for Simulating Parties \tilde{C}, D in the Head:** With the goal of completely removing the execution of π' between the adversary and an external party D we provide the adversary with a garbled circuit to simulate the execution of the protocol π' “in the head”. Just like the previous scenario consider an adversary \mathcal{A} , who interacts with A in an execution of π . In this execution, A with input (m_0, m_1) acts as the Sender and \mathcal{A} acts as the Receiver. We will refer to this execution as the *main* execution of π . However, we need this adversary \mathcal{A} to send messages to A on behalf of a receiver. Our approach for generation of these messages is provide the adversary \mathcal{A} with a garbled circuit as an auxiliary input that it can use to evaluate the next messages on behalf of D in π , in effect simulating the interaction of \tilde{C} with D “in its head”. In other words \mathcal{A} now acts on behalf of the party \tilde{B} and interacts with A in an execution of protocol π and executes the chosen protocol via evaluating the garbled circuit. This allows us to simulate the protocol π' . However, note that in order to securely evaluate the garbled circuit, the adversary \mathcal{A} will need keys corresponding to the input wires. Furthermore, since we will rely on the security of the garbled circuits itself we will need to ensure that only one key per wire can be obtained. Looking ahead this will be crucial when considering ideal-world adversaries and try to reach a contradiction. Loosely speaking this will guarantee that garbled circuits will be useful for a one-time evaluation only, revealing only the input/output behavior of the underlying function.

Next, we show how to achieve this task.

Garbled circuit keys: In the above setting note that \mathcal{A} needs to obtain keys for secure evaluation of the garbled circuit. We achieve this by using other invocations of the OT protocol π at our disposal. Note that corresponding to each input wire of the garbled circuit we will have two keys and the adversary \mathcal{A} needs to learn one of them in order to evaluate the garbled circuit correctly. Corresponding to each input wire of the garbled circuit we will provide the two keys to honest party A . The adversary \mathcal{A} can choose and obtain the appropriate keys in multiple executions of the OT protocol π . We refer to these executions as the *additional* executions of π . Note that this will involve an executions of the OT protocol that is interleaved with the main invocation of the OT protocol. The benefits achieved here are two fold. Firstly, we are now able to transfer the appropriate garbled circuit keys to the adversary \mathcal{A} via executions of π . This makes the chosen protocol attack in the setting where only multiple instance of π are executed concurrently possible. Secondly, loosely speaking, the adversary \mathcal{A} for each input wire can obtains only one of the two keys that the honest party A holds. This intuitively follows from the sender security of the OT protocol π . Further note here that since the honest party A always plays the role of the sender, the adversary \mathcal{A} is corrupting only the receiver in all executions of π . This concludes the construction of our real-world adversary.

Remark on static nature of inputs. Observe that the inputs provided to the honest party A include the values m_0, m_1 and the garbled circuit keys which can all be fixed in advance. Recall that since we are in the *static* input setting this is required for our adversary. Additionally a garbled circuit, generated as above, is provided as an auxiliary input to the adversary.

3. **The Contradiction:** Now that we have roughly specified the details of our real-world adversary \mathcal{A} we will now provide the key idea behind why no simulator (or the ideal-world adversary) \mathcal{S} can simulate the view of the adversary \mathcal{A} in all executions of π given access to the ideal functionality \mathcal{F}_{OT} only. Observe that the real-world adversary \mathcal{A} is a deterministic procedure that uses garbled circuits to securely evaluate the messages it needs to send to A . From the security of garbled circuits, the messages sent by the adversary \mathcal{A} in the main session roughly amount to be the messages generated by an external honest party. In particular this means that \mathcal{S} essentially has only black-box access to the source of these messages and it can not rewind it. Therefore the simulator \mathcal{S} can not simulate the view of the adversary in the main session allowing us to reach a contradiction.

Implications for Bounded Concurrency. Observe that the attack described in the above proof (in the unboundend concurrent setting) has natural implications in the bounded setting as well. In particular, the number of sessions that our adversary executes, or the “extent” of concurrency used by the adversary in the proof above in order to arrive at a contradiction is bounded by the communication complexity of the protocol. More specifically the adversary needs to make one additional OT call for every bit that the Sender sends in the protocol. This yields the following corollary:

Corollary 1. *Let π be any protocol that securely realizes the \mathcal{F}_{OT} functionality under m -bounded concurrent self-composition. Then (assuming one-way functions) the communication complexity of (a single session of) π is at least m bits.*

Implications in the Setting of Very Limited Concurrency. Observe that the attack described in the above proof (in the setting of arbitrary concurrent composition) has natural implications even if we restrict ourselves to a very limited concurrent composition. In particular, the adversary in the proof above only interleaves the main session with the rest of the sessions all of which are executed just sequentially.

3.2 General Theorem for Impossibility Results

In order to extend our results to more general functionalities we present a general theorem that allows us to argue impossibility for any functionality which satisfies certain special properties. We next state our theorem. In Section 4 we will use this theorem to argue impossibility for large classes of functionalities.

Theorem 2. Let \mathcal{F} be any two-party functionality between a Sender, S and a Receiver, R . Consider a protocol π that securely realizes \mathcal{F} in the concurrent setting. Then (assuming one-way functions) at least one of the following is not true.

1. **Key Transmission.** There exists a real-world adversary \mathcal{A}_1 and a distribution $(\bar{x}, z) \leftarrow \mathcal{D}_1(X_0, X_1)$ with inputs X_0, X_1 (each one bit long) such that the following holds. \mathcal{A}_1 on input b, z interacting with an honest S with input \bar{x} in concurrent executions of π outputs X_b such that every ideal-world adversary \mathcal{S}_1 running on input (b, z) that simulates \mathcal{A}_1 can be simulated⁷ by querying an oracle that holds X_0 and X_1 for only one of the values except with negligible probability.
2. **Chosen Protocol Attack.** There exists a chosen protocol π' for π and a distribution $(x, y) \leftarrow \mathcal{D}_2$ such that there exists a real-world adversary \mathcal{A}_2 (with auxiliary input z) that interacts with an honest sender A of π with input x and a honest receiver D of π' with input y and that there exists no corresponding simulator \mathcal{S}_2 . Namely, for every hybrid-world adversary \mathcal{S}_2 interacting with the ideal functionality \mathcal{F} (that talks to A) and a honest receiver D of π' we have

$$\text{EXEC}_{\mathcal{F}, \pi', \mathcal{S}_2}(k, x, y, z) \stackrel{\mathcal{C}}{\not\equiv} \text{EXEC}_{\pi, \pi', \mathcal{A}_2}(k, x, y, z)$$

We start by giving the intuition. The key difference between the theorem as stated above (beside the natural generalization) from Theorem 1 is that the above theorem requires transmission of bits only. More specifically, the adversary gets to choose one bit among X_0 and X_1 and gets X_b as the response. On the other hand in the case of string OT these values were actually strings. We deal with this problem by using a specialized garbled circuit (see the full version for more details) construction in which all the keys are bits. This construction involves applying an encoding function to the keys of the garbled circuit thereby obtaining encoded key bits. These encoded key bits are then transferred via bit OTs. The security guarantee is that each key for every input wire is encoded in a manner such that even an adversary that obtains bits of its choice can not learn more than one key per input wire. We prove the above theorem in the full version.

3.3 Example: The Case of Bit OT

In this section we give an impossibility result for the bit OT functionality. Roughly speaking, bit OT is a two-party functionality between a sender S , with input bits (m_0, m_1) and a receiver R with input b which allows R to learn m_b without learning

⁷ Note that there are two levels of simulation happening here. First, any adversary \mathcal{A}_1 who runs π with honest party input drawn from $\mathcal{D}(X_0, X_1)$ is being simulated by an ideal world adversary \mathcal{S}_1 with access to only the ideal functionality \mathcal{F} . Second, such a simulator \mathcal{S}_1 learns only one of the two values X_0 or X_1 . That is, his ideal-world interaction can actually be simulated by querying for only one of the two values (to generate the honest party input).

anything about m_{1-b} . At the same time the sender S learns nothing about b . More formally bit OT functionality is defined as $\mathcal{F}_{\text{Bit-OT}} : (\{0, 1\} \times \{0, 1\}) \times \rightarrow \{0, 1\}$ where $\mathcal{F}_{\text{Bit-OT}}((m_0, m_1), b) = m_b$ and only R gets the output. We stress that we are in the setting of *static inputs* and *fixed roles*.

Lemma 1. (*impossibility of static input concurrency - bit OT*) *Let π be any protocol which implements the $\mathcal{F}_{\text{Bit-OT}}$ functionality. Then, (assuming one-way functions) there exists a polynomial $\ell(k)$ and a distribution \mathcal{D} over $\ell(k)$ -tuple of inputs and an adversarial strategy \mathcal{A} such that for every probabilistic polynomial-time simulation strategy \mathcal{S} , definition of concurrent security cannot be satisfied when the inputs of the parties are drawn from \mathcal{D} .*

Proof (Informal). Lemma 1 follows by a direct application of Theorem 2 and the ideas developed in Theorem 1. Observe that Theorem 2 requires us to prove two properties, namely, key transmission and chosen protocol attack. Key transmission property requires us to construct an adversary \mathcal{A}_1 and a distribution \mathcal{D}_1 that generates inputs for the honest party S and auxiliary input z for the adversary \mathcal{A}_1 . Note that our adversary will run only one execution of π . In particular this means that $\nu(k) = 1$. We first specify our distribution \mathcal{D}_1 . Distribution \mathcal{D}_1 on input X_0, X_1 outputs X_0, X_1 for the honest party S and it outputs an empty string z for \mathcal{A}_1 . Observe that \mathcal{A}_1 on input b can easily obtain X_b in one execution of the protocol π playing as the receiver.

Next note that the chosen protocol attack for the case of bit OT functionality is the same as the string OT functionality. Note that our goal here is to just give intuition for Theorem 2 and since we prove general theorems for all functionalities (in Section 4) we skip the details here. ■

4 The Case of General Functionalities

In this section we give impossibility results for general functionalities. We first consider symmetric functionalities in which both parties get the same output and then we consider functionalities in which the two parties get different outputs. We stress that all our results are in the setting of *static inputs* and *fixed roles*. Surprisingly, completeness theorems (and constructions) for secure computation of such functions are known in the stand-alone two-party setting [33, 32, 34, 2].

4.1 The Case of Symmetric General Functionalities

Consider a two-party functionality \mathcal{F}_{sym} between a sender S with input x and a receiver R with input y which allows both S and R to learn $f(x, y)$ (and nothing else). More formally, let $f : X \times Y \rightarrow Z$ be any finite function⁸ then a symmetric functionality $\mathcal{F}_{sym} : X \times Y \rightarrow Z \times Z$ is defined as, $\mathcal{F}_{sym}(x, y) = (f(x, y), f(x, y))$ where both S and R get $f(x, y)$. For any *complete* symmetric function f , as

⁸ A function is said to be finite if both the domain and the range are of finite size.

defined below, we show that there does not exist a protocol π that concurrently securely realizes the \mathcal{F}_{sym} functionality.

Loosely speaking, a symmetric functionality that contains an *embedded-OR* is complete. We know that \mathcal{F}_{sym} is *complete* [44] (both in the setting of *semi-honest* and *malicious* adversaries) in the stand-alone setting of information-theoretically secure computation⁹ iff $\exists a_0, a_1, b_0, b_1$ such that $f(a_0, b_0) = f(a_0, b_1) = f(a_1, b_0) \neq f(a_1, b_1)$.

Theorem 3. (*impossibility of static input concurrent security for symmetric complete functionalities*) Let \mathcal{F}_{sym} be a functionality that is complete in the stand-alone setting and π be any protocol which implements \mathcal{F}_{sym} . Then, (assuming one-way functions) there exists a polynomial $\ell(k)$ and a distribution \mathcal{D} over $\ell(k)$ -tuple of inputs and an adversarial strategy \mathcal{A} such that definition of concurrent security cannot be satisfied when the inputs of the parties are drawn from \mathcal{D} .

Our argument in the proof of the theorem above specifically relies on the fact that the output distribution of the honest party between a real-world and an ideal-world execution cannot change. We give a proof of the above theorem in the full version. The impossibility of secure computation for complete, symmetric functionalities in the unbounded concurrent setting has natural implications in the bounded setting as well. In particular, the number of sessions, or the “extent” of concurrency used by the adversary in the proof of theorem above in order to arrive at a contradiction is polynomially related to the communication complexity of the protocol. This yields the following corollary:

Corollary 2. Let π be any protocol that securely realizes \mathcal{F}_{sym} functionality under m -bounded concurrent self-composition for any polynomial m . Then (assuming one-way functions) there exists a constant $c \geq 0$ such that for sufficiently large k , the communication complexity of (a single session of) π is at least $\frac{m}{k^c}$ -bits.

4.2 The Case of Asymmetric General Functionalities

Consider a two-party functionality \mathcal{F}_{asym} between a sender S , with input x and a receiver R with input y which allows R to learn $f(x, y)$ (and nothing else) and where S learns nothing. More formally, let $f : X \times Y \rightarrow Z$ be any finite function then an asymmetric functionality \mathcal{F}_{asym} is defined as, $\mathcal{F}_{asym}(x, y) = (\perp, f(x, y))$ where S gets nothing and R gets $f(x, y)$. We show that there does not exist a protocol π that concurrently securely realizes any *complete* \mathcal{F}_{asym} functionality as defined below.

We know that \mathcal{F}_{asym} is *complete* [33] in the setting of stand-alone two-party computation in the presence of *malicious* adversaries iff $\forall b_0, \exists b_1, a_0, a_1$ such that

$$f(a_0, b_0) = f(a_1, b_0) \wedge f(a_0, b_1) \neq f(a_1, b_1).$$

⁹ Note that the setting of stand-alone and information-theoretic security is used only to define the class of functions. We deal with the concurrent and computational security of this class of functions in this work.

Theorem 4. (*impossibility of static input concurrent security for asymmetric complete functionalities*) Let π be any protocol which implements any $\mathcal{F}_{\text{asym}}$ functionality that is complete in the stand-alone setting. Then, (assuming one-way functions) there exists a polynomial $\ell(k)$ and a distribution \mathcal{D} over $\ell(k)$ -tuple of inputs and an adversarial strategy \mathcal{A} such that definition of concurrent security, cannot be satisfied when the inputs of the parties are drawn from \mathcal{D} .

The key difference when considering asymmetric functionalities as opposed to the case of symmetric functionalities is that for asymmetric functionalities only one party gets the output. Observe that our arguments in Theorem 3 specifically relied on the fact that the output distribution of either of the parties between a real-world and ideal-world execution (note that this output contains the outputs of both the honest party and the adversarial party) cannot change. However, complete asymmetric functionalities possess a larger structure than what is available to complete symmetric functionalities. We crucially use this additional structure in our proof. We give the full proof of the above theorem in the full version. We also note that a direct analogue of Corollary 2 holds in the asymmetric setting as well.

5 Impossibility of Stateless Two-Party Computation

In this section, we show the existence of a deterministic two-party functionality for which there does not exist any stateless secure protocol. We start by giving some intuition about our impossibility result. The key observation behind our impossibility result is that even a deterministic adversary in interaction with an honest party can come up with honest looking messages on behalf of an adversarial party by obtaining them from other honest interactions. However, loosely speaking, since these messages are obtained from other honest parties, the simulator only has black box access to the source of these messages. The simulator does have non-black box access to the adversary itself, however, it is essentially useless because the adversary acts just like a “message forwarding machine.” Therefore simulator essentially only has black-box access to the adversary which alone of course does not help the simulator because a real-world adversary can also reset honest parties.

Functionalities considered. Before we move on to a formal claim, we introduce description of some notation that will be useful in this work. Let us start by describing a general functionality. Let $\mathcal{F}_{\text{universal}} : (\{0, 1\}^{p_1(k)} \times \{0, 1\}^{q_1(k)}) \times (\{0, 1\}^{p_2(k)} \times \{0, 1\}^{q_2(k)}) \rightarrow (\{0, 1\}^{r(k)} \times \{0, 1\}^{r(k)})$ be the following two-party (between P_1 and P_2) functionality, $\mathcal{F}_{\text{universal}}((C, x), (C', y))$, where $\mathcal{F}_{\text{universal}}$ obtains the input (C, x) from P_1 and (C', y) from P_2 . The functionality outputs \perp to both P_1 and P_2 if $C \neq C'$ and it outputs $C(x, y)$ to both P_1 and P_2 otherwise, where $p_1(\cdot), q_1(\cdot), p_2(\cdot), q_2(\cdot), r(\cdot)$ are polynomials. We stress that we consider a functionality which outputs the same value to both parties. This is consistent with the definition of [37]. Let π be a protocol that resetably securely

realizes the functionality $\mathcal{F}_{universal}$. Next we describe two circuits that will be useful in our context.

Equality Testing: Let $C_{eq}(x, y)$ be a circuit that outputs 1 if $x = y$ and 0, otherwise. In an execution of the ideal functionality $\mathcal{F}_{universal}$, in which the parties P_1 and P_2 input (C_{eq}, x) and (C_{eq}, y) respectively corresponds to the *equality testing* functionality.

Next Message Function of P_2 : Let C_π be the next message function of P_2 in the protocol π . In other words, C_π is a non-interactive algorithm that gets as an input, the input and random tape of P_2 and the history of messages sent by P_1 , and outputs the next message that P_2 would send in a real execution of π in which it sees this message history. More formally, C_π takes as input a sequence of messages $(h_1, h_2 \dots h_t)$ and P_2 's input (C, y) and random coins r and generates the next message of P_2 , i.e. the message that P_2 sends in the execution with the history $h_1, h_2 \dots h_t$.

Theorem 5. (*impossibility of static input resetability*) *There does not exist any stateless secure protocol for the $\mathcal{F}_{universal}$ functionality. (unconditionally)*

Now we give a brief outline of the proof. The full proof is provided in the full-version. For the sake of contradiction, assume that there exists a protocol π , with round complexity r , that resetably securely realizes the functionality $\mathcal{F}_{universal}$. We give an outline of the proof before giving all the details.

1. We consider the “first scenario” of two parties P_1 and P_2 executing π . In this setting we consider an adversary that corrupts P_2 and interacts honestly with P_1 , in the first incarnation. However, it does not generate the honest responses on its own. In fact it obtains these responses from P_1 itself (via different incarnations of P_1). In this setting, as per the definition of resetability there must exist a simulator (plausibly non-black box in this adversary), which can “extract” the input used by the adversary, on behalf of P_2 , in interaction with the first incarnation of P_1 .
2. Next we consider the “second scenario” of two parties P_1 and P_2 executing π . In this setting we construct a real-world adversary that corrupts P_1 and interacts with an honest P_2 . This adversary internally uses the ideal-world adversary constructed in the “first scenario” to extract the input of P_2 . Finally, we observe that no ideal-world adversary in this “second scenario” can achieve the same, thereby reaching a contradiction.

Acknowledgements. The work is supported in part by NSF grants 0830803, 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the authors and do not reflect the official policy or position of

the Department of Defense or the U.S. Government. The work of the fourth author has been done while visiting UCLA and is supported in part by the European Commission through the FP7 programme under contract 216676 ECRYPT II.

We thank Hemanta K. Maji, Akshay Wadia, Vipul Goyal and Amit Sahai for valuable discussions.

References

1. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167. IEEE Computer Society (1986)
2. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) STOC, pp. 218–229. ACM (1987)
3. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
4. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
5. Barak, B., Canetti, R., Nielsen, J., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)
6. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: FOCS, pp. 249–259 (2007)
7. Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
8. Chandran, N., Goyal, V., Sahai, A.: New Constructions for UC Secure Computation Using Tamper-Proof Hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)
9. Lin, H., Pass, R., Venkatasubramaniam, M.: A unified framework for concurrent security: universal composable from stand-alone non-malleability. In: STOC, pp. 179–188 (2009)
10. Groth, J., Ostrovsky, R.: Cryptography in the Multi-string Model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
11. Goyal, V., Katz, J.: Universally Composable Multi-party Computation with an Unreliable Common Reference String. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 142–154. Springer, Heidelberg (2008)
12. Garg, S., Goyal, V., Jain, A., Sahai, A.: Bringing People of Different Beliefs Together to Do UC. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 311–328. Springer, Heidelberg (2011)
13. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: FOCS, pp. 367–378 (2006)
14. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently Secure Computation in Constant Rounds. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (2012)
15. Pass, R.: Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
16. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal compositability without trusted setup. In: STOC, pp. 242–251 (2004)

17. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552. IEEE Computer Society (2005)
18. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: FOCS, pp. 541–550 (2010)
19. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority, pp. 232–241 (2004)
20. Goyal, V.: Positive results for concurrently secure computation in the plain model. Cryptology ePrint Archive, Report 2011/602 (2011), <http://eprint.iacr.org/>
21. Goyal, V., Jain, A., Ostrovsky, R.: Password-Authenticated Session-Key Generation on the Internet in the Plain Model. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 277–294. Springer, Heidelberg (2010)
22. Pass, R., Rosen, A.: Bounded-concurrent secure two-party computation in a constant number of rounds (2003)
23. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
24. Lindell, Y.: Lower bounds and impossibility results for concurrent self composition. J. Cryptology 21(2), 200–249 (2008)
25. Garay, J.A., MacKenzie, P.D.: Concurrent oblivious transfer. In: FOCS, pp. 314–324 (2000)
26. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. J. Cryptology 19(2), 135–167 (2006)
27. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS, pp. 345–354 (2006)
28. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: STOC, pp. 409–418 (1998)
29. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–431. Springer, Heidelberg (1999)
30. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in polylogarithm rounds. In: STOC, pp. 560–569 (2001)
31. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS, pp. 366–375 (2002)
32. Kilian, J.: Founding cryptography on oblivious transfer. In: Simon, J. (ed.) STOC, pp. 20–31. ACM (1988)
33. Kilian, J.: More general completeness theorems for secure two-party computation. In: STOC 2000, pp. 316–324. ACM, New York (2000)
34. Beimel, A., Malkin, T., Micali, S.: The All-or-Nothing Nature of Two-Party Secure Computation. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 80–97. Springer, Heidelberg (1999)
35. Agrawal, S., Goyal, V., Jain, A., Prabhakaran, M., Sahai, A.: New Impossibility Results for Concurrent Composition and a Non-Interactive Completeness Theorem for Secure Computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 443–460. Springer, Heidelberg (2012)
36. Goyal, V., Sahai, A.: Resettable Secure Computation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 54–71. Springer, Heidelberg (2009)
37. Goyal, V., Maji, H.K.: Stateless cryptographic protocols. In: FOCS (2011), <http://research.microsoft.com/en-us/people/vipul/gm11.pdf>
38. Goyal, V., Maji, H.K.: Personal communication (2012)

39. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-Time Programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
40. Brassard, G., Crépeau, C., Santha, M.: Oblivious transfers and intersecting codes. IEEE Transactions on Information Theory 42(6), 1769–1780 (1996)
41. Goldwasser, S., Lindell, Y.: Secure Computation Without Agreement. In: Malkhi, D. (ed.) DISC 2002. LNCS, vol. 2508, pp. 17–32. Springer, Heidelberg (2002)
42. Kelsey, J., Schneier, B., Wagner, D.: Protocol Interactions and the Chosen Protocol Attack. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 91–104. Springer, Heidelberg (1998)
43. Lindell, Y.: General composition and universal composability in secure multi-party computation. In: FOCS, pp. 394–403. IEEE Computer Society (2003)
44. Kilian, J.: A general completeness theorem for two-party games. In: Koutsougeras, C., Vitter, J.S. (eds.) STOC, pp. 553–560. ACM (1991)

New Impossibility Results for Concurrent Composition and a Non-interactive Completeness Theorem for Secure Computation

Shweta Agrawal¹, Vipul Goyal², Abhishek Jain¹,
Manoj Prabhakaran³, and Amit Sahai¹

¹ UCLA

² Microsoft Research, India

³ UIUC

Abstract. We consider the *client-server* setting for the concurrent composition of secure protocols: in this setting, a single server interacts with multiple clients concurrently, executing with each client a specified protocol where only the client should receive any nontrivial output. Such a setting is easily motivated from an application standpoint. There are important special cases for which positive results are known – such as concurrent zero knowledge protocols – and it has been an open question whether other natural functionalities such as Oblivious Transfer (OT) are possible in this setting.

In this work:

- We resolve this open question by showing that unfortunately, even in this very limited concurrency setting, **broad new impossibility results** hold, ruling out not only OT, but in fact all nontrivial finite asymmetric functionalities. Our new negative results hold even if the inputs of all honest parties are fixed in advance, and the adversary receives no auxiliary information.
- Along the way, we establish a new **unconditional completeness result** for asymmetric functionalities, where we characterize functionalities that are *non-interactively complete* secure against active adversaries. When we say that a functionality \mathcal{F} is non-interactively complete, we mean that every other asymmetric functionality can be realized by parallel invocations of several copies of \mathcal{F} , with no other communication in any direction. Our result subsumes a completeness result of Kilian [STOC'00] that uses protocols which require additional interaction in both directions.

1 Introduction

Consider the following scenario: Goldman Sachs, which has collected in-depth market research and analysis on various companies, wishes to offer a paid service to high-profile investors who are interested in using this market research. For this purpose, it has set up a large server to which potential clients can connect in order to obtain answers to queries of an authorized kind (for which they have

paid). Potential investors, however, are wary of Goldman Sachs learning about their business plans, and would want to hide the queries from Goldman Sachs – indeed, Goldman Sachs should learn nothing at all except that the client performed an authorized query. On the other hand, Goldman Sachs would like to ensure that in one session, a client can only learn the answer to a single query of a kind that it is authorized to ask. In particular, it would like to ensure that multiple clients who may connect to the server simultaneously, cannot perform a *coordinated attack* to learn more information than what each of them paid for (or even carry out an unauthorized query).

Can we satisfy these requirements? While well-known two-party computation protocols (e.g., [35,14]) offer remarkably powerful simulation-based security guarantees, these guarantees hold only in the stand-alone model, where only one protocol execution takes place. Our scenario is slightly more complex, as it has some mild concurrency in that multiple clients may interact with the server concurrently. At the same time, we are making the plausible assumption that the server is programmed only to interact with clients using the prescribed protocol, and we do not seek to guarantee security of any other protocols that the clients may be engaged in while they are communicating with the server¹. Arguably, such a *client-server* setting (formally, asymmetric functionalities in a “fixed-role concurrent self composition” setting) is of great practical relevance. But apart from the practical significance, from a theoretical point of view, it is an important question as to whether restricting to such a model of concurrent executions of a single protocol, allows us to recover strong security guarantees for two-party computation (at least for some functionalities).

In this work, we consider secure computation in the client-server setting, and show that, even in this highly restricted concurrency setting, broad impossibility results for secure computation hold.

- We establish **new and broad impossibility results** for achieving security under fixed-roles concurrent self composition, ruling out *all finite functionalities* (except “trivial” ones which have universally composable protocols), including many natural functionalities such as oblivious transfer. Our results hold even if the inputs to the parties in all sessions are fixed before the protocols commence.
- Along the way, we establish a new **unconditional completeness result** for asymmetric functionalities, where we characterize functionalities that are *non-interactively complete*² for secure computation against active adversaries. This subsumes a result of Kilian [22] that used a protocol which had additional interaction and, for its security, relied on the properties of a Nash equilibrium of a zero-sum game.

¹ If we did consider the security of other protocols that the clients were engaged in, then known sweeping impossibility results would apply. See further details below.

² We say that a functionality \mathcal{F} is non-interactively complete if every other asymmetric functionality can be realized by parallel invocations of several copies of \mathcal{F} , with no other communication in any direction.

Background: Security under Concurrent Composition. With the proliferation of the network setting, in particular the *Internet*, the last decade has seen a push towards constructing protocols that have strong concurrent composability guarantees. For example, we could require security under *concurrent self-composition* (which is the focus of this work): a protocol should remain secure even when there are multiple copies executing concurrently. The framework of universal composability (UC) [4] was introduced to capture the more general setting of *concurrent general composition*, where a protocol may be executed concurrently with not only several copies of itself but also with other arbitrary protocols.

General positive results for UC secure computation have been obtained based on various *trusted* setup assumptions such as a common random string [6,9,1,10,20,25]. Whether a given set of players is actually willing to trust an external entity, however, is debatable. Indeed a driving goal in cryptographic research is to eliminate the need to trust other parties. Ideally, we would like to achieve security under concurrent composition in the *plain model* (which is the main focus of this work).

The Dark Side of Concurrency. Unfortunately, in the plain model, by and large, most of the results have been negative.³ UC secure protocols for most functionalities of interest were ruled out in [6,7,32]. (Recently, these were extended to various public key models in [21].) These impossibility results were extended to the setting of general composition by Lindell [26] who proved that security under concurrent general composition implies UC security. Later, Lindell [27] established broad negative results even for the setting of concurrent self-composition by showing equivalence of concurrent self-composition and general composition for functionalities where each party can “communicate” to the other party via its output (referred to as *bit transmitting functionalities*). Barak et al. [2] (and more recently, [16]) obtained negative results for very specific functions in the “static-input” setting (i.e., where the inputs of honest parties are fixed in advance for all the protocol sessions).

On the positive side, there has been much success in obtaining construction for zero-knowledge and related functionalities, with security in similar models (e.g., [11,34,23,31,2,28]). Very recently, Goyal [16] has been able to obtain positive results for a large class of functionalities in this setting with the restriction that an honest party uses the *same*, static input in all of the sessions. However these results do not translate to the more general setting where the server may choose different inputs in different sessions. Indeed, in a scenario such as the one discussed earlier, if the server is required to use the same static input in all sessions, it will have to allow all clients the same level of access, and further use its *entire* database in every computation (which may be impractical).

³ We remark that several works have obtained positive results for “non-standard” simulation-based security, in which the simulator, for example, has access to super-polynomial computational power [30,33,3,29,18,8,12]. In this work, we will focus on security w.r.t. standard (polynomial time) simulation.

Our Question: Static-Input, Fixed-Roles Concurrent Self-composition?

In this work, we study the (im)possibility of achieving secure two-party computation with respect to a very weak notion of concurrency as occurs in the *client-server* setting: one server interacts with many clients using the same protocol, always playing the same role, while each (honest) client interacts only with the server. Further, the functionalities being computed are *asymmetric*, in that only the client gets any output.⁴ The adversarial model is that either the clients or the server may be adversarial – this is referred to as the *fixed roles* setting in the literature [27].^{5,6} We note that this is a very natural setting and captures several important functionalities such as **oblivious transfer**. However, despite extensive prior work on concurrent security, this setting has remained largely unstudied for general secure two-party computation (with the exception of [2,16] as discussed above). Indeed, concurrent security of oblivious transfer under (unbounded) concurrent composition was left as an explicit open problem by Lindell (see [28, pg. 6]).

The importance of the above question stems from the fact that if the answer is positive, then it would enable security of many application scenarios, such as the one discussed at the beginning. On the other hand, if the answer is negative, then the situation would indeed be quite stark, and reaffirm the need for relaxing the standard security definitions. The recent positive results of Goyal [16] may give hope that the answer is positive. Unfortunately, in this work, we show that the latter case holds. We now proceed to describe our results.

1.1 Our Results

We obtain negative results regarding two-party computation with security under concurrent self-composition in the *fixed-roles* setting, along with positive results on UC-secure reductions that were used to get the negative results. (These are formally stated in Section 5.)

- We give a **full-characterization** of security under concurrent self-composition for deterministic *asymmetric* finite functionalities (in which only one party receives output). (Theorem 4.) Specifically, we prove that no *non-trivial* deterministic asymmetric finite functionality can be computed securely under concurrent self composition, while *trivial* ones can be computed UC securely against active adversaries. (A deterministic asymmetric

⁴ Functionalities in which both parties receive output (which is not entirely a function of their local input) are more “complex” and already ruled out by Lindell [27], when the inputs could be adaptively chosen.

⁵ One could consider the corruption model where one or more clients and the server are corrupted. We note that if communication pattern is “bipartite” (i.e., honest clients do not talk to each other), then this is also covered in the fixed roles setting.

⁶ Note that in the setting of *inter-changeable roles* (i.e., where parties may assume different roles in different protocol executions), essentially all non-trivial functionalities allow bit-transmission, and hence the negative results of Lindell [27] are applicable.

functionality is said to be non-trivial if there does not exist a single input for the receiver that “dominates” all other inputs.) Our results are *unconditional* and hold in the *static-input* setting, and thus also rule out the more general case where a party may choose its input in a session *adaptively*, based on the outcomes of the previous sessions. Further, our results are “robust” in the sense that the impossibility is not because honest parties could choose their inputs by reacting to signals sent by corrupt parties over subliminal channels.

In particular, our results rule out concurrent security of 1-out-of-2 oblivious transfer and thus settle the open question of Lindell [28]. Furthermore, to the best of our knowledge, these are the first broad impossibility results in the *static-input* setting. (In contrast, prior works which considered static inputs [2,16] only ruled out very specific functionalities.)

- To prove the above, we first construct a new UC-secure *asynchronous non-interactive* protocol for 1-out-of-2 oblivious transfer (\mathcal{F}_{OT}) using any given non-trivial deterministic asymmetric functionality \mathcal{F} , thereby subsuming a result of Kilian [22]. By a non-interactive protocol we mean that the only step in the protocol is to invoke, in parallel, several copies of the given functionality \mathcal{F} ; we say that the protocol is asynchronous if it remains secure even if the adversary can adaptively schedule the different invocations of \mathcal{F} . (Theorem 1.)
 - Combining the above protocol with a UC-secure non-interactive protocol from [19, Full version] for any asymmetric functionality \mathcal{F} given \mathcal{F}_{OT} , we obtain a full characterization of completeness among deterministic asymmetric functionalities with respect to non-interactive reductions. (Theorem 6.)
- We further devise a *composition theorem* for static-input fixed-role concurrent security. (Theorem 3.) This theorem holds only for asynchronous non-interactive protocols. Given this theorem and our asynchronous non-interactive OT protocol, we complete the proof of our main result by establishing the impossibility of static-input, fixed-role concurrently secure protocols for \mathcal{F}_{OT} . (Theorem 2.)

Independent Work. Independent of our work, exciting results concerning the impossibility of concurrently secure computation have been obtained recently by Garg *et al.* We refer the reader to [13], in these proceedings, for more details.

1.2 Our Techniques

Asynchronous Non-interactive Protocol for \mathcal{F}_{OT} . As mentioned above, the first ingredient in our main result, and a contribution of independent interest, is an asynchronous non-interactive protocol for \mathcal{F}_{OT} given any non-trivial deterministic asymmetric functionality \mathcal{F} . To obtain the impossibility result in the static-input setting it is vital that this protocol is non-interactive *and asynchronous*.

Let \mathcal{F} be a *non-trivial* asymmetric functionality that takes inputs x and y from Alice and Bob respectively, and outputs $f(x, y)$ to Bob. The intuitive reason why

such a functionality \mathcal{F} can yield \mathcal{F}_{OT} is that Bob has no input which will let it learn Alice's input to \mathcal{F} exactly (up to equivalent inputs), whereas Alice does not learn anything about Bob's input to \mathcal{F} . In particular, one can find two inputs \hat{y}^0 and \hat{y}^1 for Bob, such that if Alice chooses her input x at random, the only way Bob can learn $f(x, \hat{y}^0)$ with full certainty is if he chooses \hat{y}^0 as his input; similarly, unless he deterministically chooses \hat{y}^1 as his input, Bob will be left with some entropy regarding $f(x, \hat{y}^1)$. Thus Bob can choose to learn at most one of $f(x, \hat{y}^0)$ and $f(x, \hat{y}^1)$ exactly. There are two main challenges in turning this idea into an implementation of \mathcal{F}_{OT} :

- Bob learns some information about $f(x, \hat{y}^0)$ when he uses \hat{y}^1 as an input, and vice versa, whereas in \mathcal{F}_{OT} , he should not learn any information about one of Alice's two inputs (and learn the other one completely).
- Alice and Bob may deviate from the prescribed distributions when choosing their inputs to \mathcal{F} . In particular, any solution to the above issue should work even if Bob uses an input other than \hat{y}^0 and \hat{y}^1 .

Kilian [22] handles the issue of active adversaries using properties of a Nash equilibrium of an appropriate zero-sum game. However, this approach (apart from being not asynchronous) appears suitable only for constructing an erasure channel, and does not permit Bob to use an input. (Converting the erasure channel to \mathcal{F}_{OT} requires interaction.) The way [24] handles active corruption using cut-and-choose checks is highly interactive and again inadequate for our purposes.

One natural approach one could consider is to use an extractor to amplify the uncertainty Bob has about $f(x, \hat{y}^0)$ or $f(x, \hat{y}^1)$ from many invocations of \mathcal{F} (with x independently randomly chosen each time). That is, if Bob has some uncertainty about at least one of the strings, R^0 and R^1 where R^0 (respectively, R^1) is defined as the string of outputs Bob would have received if he chose \hat{y}^0 (respectively, \hat{y}^1) as his input in all invocations of \mathcal{F} , then Alice can choose two seeds for a strong extractor and obtain two masks r_0 and r_1 as the strings extracted from R^0 and R^1 respectively, using these seeds. Unfortunately, *this is not secure in an asynchronous protocol*. Alice must transmit the extractor seeds she picked to Bob (for which she can use instances of \mathcal{F}). However, in an asynchronous non-interactive protocol, a corrupt Bob can *first receive the extractor seeds* before picking its inputs for the other instances of \mathcal{F} ; hence the seeds are not independent of the information Bob obtains about R^0 and R^1 , and the guarantees of extraction no more hold.

We get around this by avoiding using *the full power of extractors*, and in fact using a deterministic function in its place. For instance, when f is a boolean function, consider defining r_0 as simply the XOR of all the bits in R^0 (and similarly r_1). Since Alice picks her input x to \mathcal{F} independently for each invocation, this still guarantees that if Bob has uncertainty about sufficiently many bits in R^0 , then he has almost no information about r_0 .

But using a linear function in place of the extractor will not suffice when Bob can use inputs other than \hat{y}^0 and \hat{y}^1 . In particular, for boolean functions again, if there is an input \hat{y}^2 such that $f(x, \hat{y}^2) = f(x, \hat{y}^0) \oplus f(x, \hat{y}^1)$, then using this input in all invocations, Bob can learn $R^0 \oplus R^1$, and $r_0 \oplus r_1$. Our solution to this is to use a simple “quadratic” function, after appropriately mapping Bobs outputs to a field. This lets us ensure that one of the two “extracted” strings r_0 and r_1 remains almost uniformly random, even given the other string.

Impossibility for \mathcal{F}_{OT} . As the next step towards our final impossibility result, we rule out a protocol for concurrent OT even for the case where both parties have fixed roles. The basic idea behind this impossibility result builds on the techniques from [2,16]. The proof proceeds in the following high-level steps; we refer the reader to Section 4 for details. Suppose, towards contradiction, we are given a protocol Π_{OT} that securely realizes OT in our setting.

1. First, we will construct an instance of the *chosen protocol attack* for this protocol. More precisely, we will construct a protocol $\widehat{\Pi}_{\text{OT}}$ such that the protocols Π_{OT} and $\widehat{\Pi}_{\text{OT}}$ are *insecure* when executed concurrently. We will have three parties in the system: Alice and Eve running Π_{OT} (as sender and receiver respectively); Eve and David running $\widehat{\Pi}_{\text{OT}}$ (as sender and receiver respectively). In our chosen protocol attack, Eve will be the corrupted party acting as man-in-the-middle between Alice and David and violating security of Π_{OT} (see Section 4 for more details of this step).
2. Next, we will use *one time programs* (OTPs) [15] to eliminate David. In more detail, Eve simply gets a set of one-time programs implementing the next message function of David.
3. To execute these one-time programs, the (possibly adversarial) Eve is required to carry out a number of oblivious transfer invocations. In these invocations, Alice can be given the required key and can act as the sender. Fortunately, oblivious transfer is exactly the functionality that Π_{OT} provides! So these OT invocations can be executed using the protocol Π_{OT} .
4. Thus, now there are only Alice and Eve running a number of concurrent executions of Π_{OT} . Hence, the chosen protocol attack we started with can now be carried out in our setting of concurrent self-composition with static-inputs and fixed-roles.

2 Preliminaries

Below we present some of the important definitions we need. Through out this paper, we denote the security parameter by κ .

Secure Computation under Concurrent Self-composition. In this section, we present the definition for concurrently secure two-party computation.

The definition we give below is an adaptation of the definition of security under concurrent self-composition from [27], but with two further restrictions to the model — *fixed-roles* and *static-inputs* — i.e., there are only two parties engaged in multiple executions of a given protocol, with each playing the same “role” in

all the sessions, and the inputs of the honest parties for each session is fixed in advance. (Recall that since the focus in this work is on obtaining impossibility results, our results are stronger by adding these restrictions.) Some parts of the definition below have been taken almost verbatim from [27].

A two-party functionality⁷ \mathcal{F} with input domains X and Y for the two parties is defined by two functions $f_1 : X \times Y \rightarrow Z_A$ and $f_2 : X \times Y \rightarrow Z_B$, where Z_A, Z_B are the output domains. For most part we shall consider the input and output domains to be finite sets. Such functionalities are called *finite functionalities*. (Again, since our focus is on impossibility results, it is more interesting to show impossibility of finite functionalities than of infinite functionalities.)

We will denote the two parties that wish to jointly instantiate \mathcal{F} as Alice and Bob (or sometimes P_1 and P_2). If Alice's input is $x \in X$ and Bob's input is $y \in Y$, then the functionality would output $f_1(x, y)$ to Alice and $f_2(x, y)$ to Bob (unless aborted by a corrupt party). A functionality is called *asymmetric* if only Bob receives any output; more precisely, in an asymmetric functionality f_1 is the constant function (Alice does receive this fixed output to indicate the termination of execution). An asymmetric functionality will be defined using a single function $f : X \times Y \rightarrow Z$.

In this work, we consider a malicious, static adversary. The scheduling of the messages across the concurrent executions is controlled by the adversary. The security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario, where a trusted party computes the function output on the inputs of the parties. We do not require fairness (i.e., our impossibility is not a consequence of requiring fairness) and hence in the ideal model, we allow a corrupt party to receive its output in a session and then optionally block the output from being delivered to the honest party, in that session. Unlike in the case of stand-alone computation, in the setting of concurrent executions, the trusted party computes the functionality many times, each time upon different inputs. We refer the reader to the full version of the paper, or [27] for further details of the real and ideal model executions.

The output pair of the ideal-model adversary \mathcal{S} and the honest party (or both parties, when neither corrupt) in an ideal-model execution of a functionality \mathcal{F} with security parameter κ , input vectors \mathbf{x}, \mathbf{y} and auxiliary input z to \mathcal{S} , will be denoted as $\text{IDEAL}_{\mathcal{F}, \mathcal{S}}(\kappa, \mathbf{x}, \mathbf{y}, z)$. Similarly, the output pair of the real-model adversary \mathcal{A} and the honest party (or parties) in a real-model concurrent execution of a protocol Π with security parameter κ , input vectors \mathbf{x}, \mathbf{y} and auxiliary input z to \mathcal{A} will be denoted by $\text{REAL}_{\Pi, \mathcal{A}}(\kappa, \mathbf{x}, \mathbf{y}, z)$.

Definition 1 (Security under Concurrent Self-Composition). *A protocol Π is said to securely realize a functionality \mathcal{F} under concurrent composition if for every real model non-uniform probabilistic polynomial-time adversary \mathcal{A} , there exists an ideal-model non-uniform probabilistic expected polynomial-time adversary \mathcal{S} , such that for all polynomials $m = m(\kappa)$, every $z \in \{0, 1\}^*$,*

⁷ All functionalities considered in this paper are (unfair) secure function evaluation (SFE) functionalities. For simplicity we shall not explicitly qualify them as SFE or non-reactive.

every pair of input vectors $\mathbf{x} \in X^m$, $\mathbf{y} \in Y^m$, $\{\text{IDEAL}_{\mathcal{F}, \mathcal{S}}(\kappa, \mathbf{x}, \mathbf{y}, z)\}_{\kappa \in \mathbb{N}}$ and $\{\text{REAL}_{\Pi, \mathcal{A}}(\kappa, \mathbf{x}, \mathbf{y}, z)\}_{\kappa \in \mathbb{N}}$ are computationally indistinguishable.

We shall also consider a few *stronger* security definitions, that we achieve in our positive results (used in proving the main negative result). Firstly, we can require the above to hold even with probabilistic *expected* polynomial time adversaries in the real-model. Secondly, we could in fact allow the real-model adversary to be computationally unbounded, and allow the ideal-model adversary to be unbounded too.⁸ Also, we shall consider Universally Composable (UC) security [4,5]. We shall not present the details of the UC security definition (which has several slight variants, with essentially the same properties), but remark that it implies concurrent self-composition and more.

Finally, sometimes we permit the real model protocols to invoke ideal functionalities as sub-protocols (denoted like $\Pi^{\mathcal{F}}$, where \mathcal{F} is the ideal functionality invoked by the parties in Π). In all the instances we do this, we can actually consider UC-security of the protocol; however this definition can be easily generalized to security under concurrent self-composition too, and would be useful in stating a composition theorem (without involving UC-security).

Non-interactive Protocols and Asynchronous Non-interactive Protocols. A two-party protocol $\Pi^{\mathcal{F}}$ (i.e., in the \mathcal{F} -hybrid model) is called a *non-interactive protocol* if the protocol has the following structure: the two parties carry out local computation; then together they invoke one or more *parallel, synchronized* sessions of \mathcal{F} ; then they carry out more local computation and produce outputs. By synchronized sessions we mean that even corrupt players can invoke these sessions only in parallel.⁹ We shall also require that all copies of \mathcal{F} are invoked with the same fixed roles for the two parties.

A two-party protocol $\Pi^{\mathcal{F}}$ is called an *asynchronous non-interactive protocol* if the protocol has the above structure, but the parallel sessions of \mathcal{F} are parallel but asynchronous. By this we mean that a corrupt player can invoke these sessions in arbitrary order, choosing the inputs for each session based on the outputs from prior sessions; the honest players have to choose their inputs *a priori* and remain oblivious to the order in which the sessions are invoked.

One Time Programs. A one-time program (OTP) [15] for a function f allows a party to evaluate f on a single input x chosen by the party dynamically. As introduced in [15], an OTP is implemented as a package consisting of some software and hardware tokens (specifically *one time memory* tokens), that essentially provided the party with *asynchronous* access to several oblivious transfer invocations. We shall treat OTPs as an *asynchronous non-interactive protocol* in the OT-hybrid model (as was done in [17]), that securely realizes an

⁸ This is not a strengthening of the security definition, as the ideal-model is more relaxed now; however, the protocols we shall consider will satisfy this definition in addition to the other definitions.

⁹ A more accurate notation for such a protocol that invokes at most t sessions of \mathcal{F} , would be $\Pi^{\mathcal{F}^t}$, where \mathcal{F}^t stands for a non-reactive functionality implementing t independent copies of f . For simplicity we do not use this notation.

asymmetric “one-time program functionality” \mathcal{F}_{OTP} against corruption of the receiver alone. \mathcal{F}_{OTP} accepts a circuit f from the party P_1 and an input x from the party P_2 , and returns $f(x)$ to P_2 (and notifies P_1). We refer the reader to [17] or the full version for details.

Definition 2 (One-Time Program). (*Adapted from [15,17]*) A one-time program (OTP) scheme is an asynchronous two-party non-interactive protocol $\Pi^{\mathcal{F}_{\text{OT}}}$ (in the \mathcal{F}_{OT} -hybrid model) that UC-securely realizes \mathcal{F}_{OTP} when the adversary is allowed to corrupt only P_2 .

In the text, we shall refer to the collection of inputs P_1 provides to the \mathcal{F}_{OT} instances in an execution of $\Pi^{\mathcal{F}_{\text{OT}}}$ when its input is a circuit f , as an *OTP for f* . We note that OTPs exist if a (standalone secure) OT protocol exists [17], which in turn exists if a concurrent secure OT protocol exists. Thus, for proving the impossibility of concurrent secure OT protocols, we can assume the existence of OTPs “for free.”

Our use of OTP parallels the use of garbled circuits in the impossibility result in [2]. Following [16], we use OTPs instead of garbled circuits, since they have stronger security properties (namely, security against an actively corrupt receiver), and allows one to simplify the construction in [2]. We refer the reader to the full version for further discussion.

3 A Non-interactive Protocol for OT from Any Non-trivial Asymmetric SFE

The goal of this section is to obtain an asynchronous, non-interactive protocol for \mathcal{F}_{OT} in the \mathcal{F} -hybrid model, for *any* finite deterministic “non-trivial” asymmetric functionality \mathcal{F} . For our purposes, we define a *trivial* asymmetric functionality as follows.

Definition 3 (Dominating input). In an asymmetric functionality defined by a function $f : X \times Y \rightarrow Z$, an input $y \in Y$ for the receiver is said to dominate another input $y' \in Y$ if $\forall x_1, x_2 \in X$, $f(x_1, y) = f(x_2, y) \implies f(x_1, y') = f(x_2, y')$.

Definition 4 (Trivial functionality). An asymmetric functionality is called trivial if there exists an input for Bob that dominates all of its other inputs.

We now state the main result in this section.

Theorem 1. For any non-trivial asymmetric functionality \mathcal{F} , there exists an asynchronous, non-interactive protocol $\Pi^{\mathcal{F}}$ that UC-securely realizes \mathcal{F}_{OT} , even against computationally unbounded adversaries.

Background. In [22] Kilian presented an elegant protocol to show that any non-trivial asymmetric functionality is complete for security against active adversaries. The protocol, which constructs an erasure channel from a non-trivial asymmetric functionality \mathcal{F} , achieves security against active adversaries

by using an input distribution to \mathcal{F} derived from the Nash equilibrium of a zero-sum game defined using \mathcal{F} . Kilian shows that an adversary deviating from the prescribed distribution cannot change the erasure probability in the protocol. This rather enigmatic protocol does invoke \mathcal{F} with fixed roles, but is not useful for showing impossibility of concurrent secure protocol for \mathcal{F} , because it is modestly interactive (two steps which should occur one after the other) and more importantly, because it yields only an erasure channel and not a $(^2_1)$ -OT.¹⁰ The only known substitute for this protocol, by [24], is much more interactive, involving several rounds of communication in both directions, apart from the invocation of \mathcal{F} . Our challenge is to devise an *asynchronous non-interactive* protocol which uses \mathcal{F} with fixed-roles and directly yields $(^2_1)$ -OT. Being non-interactive and asynchronous requires that all the sessions are invoked together by an honest party, but the adversary is allowed to schedule them adaptively, and base its inputs for later sessions based on the outputs from earlier sessions (rushing adversary). As mentioned in Section 1.2, this rules out some standard techniques like privacy amplification (using extractors), since the adversary can learn the seed used for extraction *before* it extracts partial information about a string to which the extraction will be applied.

We present a new and simple non-interactive OT protocol which uses a simple non-linear “extraction” strategy that does not require a seed, but is sufficient to amplify the uncertainty about certain values into almost zero information about at least one of two extracted values. Our protocol is in fact UC-secure (in the PPT as well as information theoretic settings) and is asynchronous.

3.1 The New Protocol

Our asynchronous non-interactive protocol UC-securely realizes the $(^2_1)$ -OT functionality \mathcal{F}_{OT} in the \mathcal{F} -hybrid model, where \mathcal{F} is a finite¹¹ asymmetric functionality defined by a function $f : X \times Y \rightarrow Z$, and \mathcal{F} is *not trivial*. Note that (since domination is transitive) this means that there are at least *two inputs* in Y — denoted by \hat{y}^0 and \hat{y}^1 — which are not dominated by any other input $y \in Y$.

To define the protocol, first we pick a prime number $p \geq \min\{|X|, |Z|\}$. Then we can define two maps to relabel the columns corresponding to \hat{y}^0 and \hat{y}^1 in the function table of f using elements in \mathbb{Z}_p — i.e., two injective functions $N_0 : Z_0 \rightarrow \mathbb{Z}_p$, $N_1 : Z_1 \rightarrow \mathbb{Z}_p$, where $Z_b = \{f(x, \hat{y}^b) | x \in X\}$ — such that there exist $\hat{x}^0, \hat{x}^1 \in X$ satisfying the following.

¹⁰ Our impossibility result in Section 4 holds only for $(^2_1)$ -OT, and not for erasure channels. Indeed, using techniques in [16,18], it would be possible to construct concurrent secure protocols for an asymmetric functionality like erasure channels, in which Bob does not have any input.

¹¹ For simplicity, following [22], we require $|X|, |Y|$ to be constant. But the security of the protocol presented here only requires $|X|$ to be $\text{poly}(\kappa)$ where κ is the security parameter. Alternatively, if $|Y|$ is $\text{poly}(\kappa)$, we can have the protocol use a uniform distribution over a subset of X of size at most $2|Y|$, restricted to which the functionality is still non-trivial.

$$\begin{bmatrix} N_0(f(\hat{x}^0, \hat{y}^0)) & N_1(f(\hat{x}^0, \hat{y}^1)) \\ N_0(f(\hat{x}^1, \hat{y}^0)) & N_1(f(\hat{x}^1, \hat{y}^1)) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

To illustrate this for the case of boolean functions ($p = |Z| = 2$), we note that for a non-trivial boolean functionality, the function table of f , restricted to the two columns corresponding to \hat{y}^0 and \hat{y}^1 must have one of the following minors (possibly with the columns reordered, in the last case): $\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$. In the first two cases N_0, N_1 can be the identity map and in the last case exactly one of N_0, N_1 would be the identity. The formal proof is given in the full version.

The protocol is now defined as follows, in terms of the inputs $\hat{y}^0, \hat{y}^1 \in Y$, $\hat{x}^0, \hat{x}^1 \in X$ and the functions N_0, N_1 as identified above.

Alice's Program: Alice's input is two bits s_0, s_1 .

1. Alice carries out the following computations:
 - For $i = 1$ to 2κ , pick $x_i \leftarrow X$.
 - For each i , let $R_i^0 = N_0(f(x_i, \hat{y}^0))$ and $R_i^1 = N_1(f(x_i, \hat{y}^1))$.
 - Let $r_0 = \sum_{i=1}^{\kappa} R_i^0 R_{\kappa+i}^0$, and $r_1 = \sum_{i=1}^{\kappa} R_i^1 R_{\kappa+i}^1$.
 - Let $m_0 = s_0 + r_0$ and $m_1 = s_1 + r_1$ (interpreting bits s_0, s_1 as elements in $\{0, 1\} \subseteq \mathbb{Z}_p$).
2. Alice invokes, in parallel, several copies of \mathcal{F} with the following inputs:
 - Sessions $i = 1$ to 2κ with inputs x_i .
 - $2\lceil \log p \rceil$ more sessions to “communicate” the bits of (m_0, m_1) : in a session to send a bit 0, use input \hat{x}^0 , and in a session to send a bit 1, use input \hat{x}^1 .
3. If all \mathcal{F} sessions are completed, then Alice completes the protocol (i.e., outputs an acknowledgment).

Bob's Program: Bob's input is a choice bit b .

1. Bob invokes the same copies of \mathcal{F} as Alice with the following inputs:
 - In each of the 2κ sessions numbered $i = 1$ to 2κ , use input \hat{y}^b , and obtain R_i^b .
 - In each of the sessions used for communication, use input \hat{y}^0 ; obtain all bits of (m_0, m_1) .
2. If all sessions of \mathcal{F} are completed, compute $r_b = \sum_{i=1}^{\kappa} R_i^b R_{\kappa+i}^b$, and $s_b = m_b - r_b$. Then, if $s_b = 0$ output 0, otherwise output 1.

Below we sketch the intuition behind the proof of security. The formal simulation and proof is presented in the full version.

The protocol is easily seen to be correct. Also, security when Alice is corrupt is easy to argue as \mathcal{F} does not give any output to Alice. (The simulator can extract Alice's inputs by considering what Bob would output when his input is $b = 0$ and $b = 1$.) The interesting case is when Bob is corrupt.

Note that in the protocol, all the instances of \mathcal{F} are invoked in parallel by the honest parties, but a rushing adversary that corrupts Bob can dynamically schedule the sessions and also choose its inputs for these sessions adaptively,

based on the outputs received thus far. The main idea behind showing security against Bob is that one of r_0 and r_1 appears completely random to Bob (even given the other one), no matter how he chooses his inputs y_i . For this we define a pair of \mathcal{F} sessions $(i, \kappa + i)$ to be a “0-undetermined pair” if $R_i^0 R_{\kappa+i}^0$ is not completely determined by the view of the adversary in those sessions, combined with $R_i^1 R_{\kappa+i}^1$; similarly we define the pair to be a “1-undetermined pair” if $R_i^1 R_{\kappa+i}^1$ is not completely determined by the view of the adversary in those sessions, combined with $R_i^0 R_{\kappa+i}^0$. Then, it can be shown that there will be a constant probability that any pair will be either 0-undetermined or 1-undetermined.

Note that for any input y that the adversary chooses in the first session out of a pair, it does not dominate at least one of \hat{y}^0 and \hat{y}^1 . With constant probability the adversary will be left with some uncertainty about either $f(x, \hat{y}^0)$ or $f(x, \hat{y}^1)$, where x stands for Alice’s input in this session. Suppose $f(x, \hat{y}^0)$ is not fully determined. Now, with a further constant probability Alice’s input in the other session in the pair would be $x' = \hat{x}^1$. Then, even if Bob learns x' exactly, he remains uncertain of $f(x, \hat{y}^0) \cdot f(x', \hat{y}^0) = f(x, \hat{y}^0) \cdot 1$. This uncertainty remains even if Bob learns $f(x, \hat{y}^1) \cdot f(x', \hat{y}^1) = f(x, \hat{y}^1) \cdot 0$, as it is independent of x .

This slight uncertainty about a term in r_0 or r_1 gets amplified by addition in \mathbb{Z}_p , as summarized in the following lemma.

Lemma 1. *Let p be a fixed prime number. Let D be a distribution over \mathbb{Z}_p , such that for some constant $\epsilon > 0$, for all $z \in \mathbb{Z}_p$, $\Pr_{a \leftarrow D}[a = z] < 1 - \epsilon$. For any positive integer N , let a_1, \dots, a_N be i.i.d random variables sampled according to D . Then the statistical distance between the distribution of $\sum_{i=1}^N a_i$ (summation in \mathbb{Z}_p) and the uniform distribution over \mathbb{Z}_p is negligible in N .*

This follows from an elementary argument. The proof is given in the full version.

To complete the intuitive argument of security, we need to also consider how the simulator for Bob can extract his input bit b . The simulator would let Bob schedule several sessions, until a constant fraction of the pairs $(i, \kappa + i)$ have had both sessions completed. At this point Bob would have already accumulated sufficient uncertainty about one of r_0 and r_1 , say $r_{\bar{b}}$, that will remain no matter what he learns from the remaining sessions. Further, not having invoked the remaining sessions will ensure that at this point he still has no information about the other element r_b . So, at this point, the simulator will send $b = 1 - \bar{b}$ to the ideal \mathcal{F}_{OT} and learn what r_b should be, and can henceforth “pretend” that it was always using that value of r_b . Pretending thus (i.e., sampling Alice’s inputs for the remaining sessions according to the right conditional distribution) can be efficiently done by rejection sampling (since p is a constant).

4 Impossibility of Concurrent Oblivious Transfer

Theorem 2. *There does not exist a protocol that securely realizes \mathcal{F}_{OT} under concurrent self-composition even in the static-input, fixed-role setting.*

Suppose towards contradiction, we are given a protocol Π_{OT} that securely realizes the OT functionality under static-input, fixed-roles concurrent self-composition. We will exhibit a set of inputs (for sender and receiver) and a real-world adversarial receiver that is able to perform a concurrent attack and manages to learn a secret value, called `secret` with probability 1. We will then prove that if there exists an adversarial receiver in the ideal world that is able to learn `secret` with high enough probability, then we can break the stand-alone security of Π_{OT} against a cheating sender, thus arriving at a contradiction. Our formal proof follows the high-level structure as discussed in Section 1.2. We now proceed to give details of each of the steps.

Chosen Protocol Attack. Let Π_{OT} be a protocol that securely realizes the OT functionality. To fix notation, let us consider two parties Alice and Bob that are executing an instance of Π_{OT} . Say that Alice's input bits are denoted by s_0 and s_1 and Bob's input bit is b . Upon successful completion of the protocol, Bob obtains s_b . Next, let $\widehat{\Pi}_{\text{OT}}$ be a slightly modified version of Π_{OT} where the receiver Bob also has both of Alice's inputs, s_0 and s_1 in addition to his bit b . In $\widehat{\Pi}_{\text{OT}}$, Bob and Alice run an execution of Π_{OT} with inputs b and s_0, s_1 respectively. Upon receiving an output s^* , Bob checks whether $s^* = s_b$. If so, he sends `secret` = s_b to Alice.

Now, consider the following scenario involving three parties Alice, Eve and David. Alice holds input bits s_0, s_1 , while David holds s_0, s_1 , as well as a random input bit b . Alice plays the sender with receiver Eve in an execution of Π_{OT} , and Eve plays sender with receiver David in an execution of $\widehat{\Pi}_{\text{OT}}$. It is clear that a malicious Eve can launch “man-in-the-middle” attack, where she simply forwards Alice's message to David and David's back to Alice, in order to learn the value `secret` = s_b . However, note that if the execution of Π_{OT} is replaced with an ideal call to the OT functionality, then the attack does not carry through.

Converting $\widehat{\Pi}_{\text{OT}}$ to Π_{OT} . Note that the above attack is valid in the setting of concurrent general composition. However, we are interested in the setting of concurrent self composition, where only Π_{OT} is executed concurrently. Towards this end, in this section, we will convert the protocol $\widehat{\Pi}_{\text{OT}}$ into a series of OT calls which can be implemented by Π_{OT} . Since Π_{OT} executed concurrently with $\widehat{\Pi}_{\text{OT}}$ is insecure, this will allow us to show that Π_{OT} is insecure under concurrent self composition.

To begin, we transform the protocol $\widehat{\Pi}_{\text{OT}}$ run by Eve (as sender) and David into a sequence of calls made by Eve to an ideal reactive functionality (with inputs fixed in advance). As in [2], it is natural to instantiate this ideal functionality by the *next message function* $\mathcal{F}^{\text{David}}$ of David's strategy in protocol $\widehat{\Pi}_{\text{OT}}$. Then Eve can simulate the interaction with David by invoking $\mathcal{F}^{\text{David}}$ each time she expects a message from David. More precisely, the inputs to $\mathcal{F}^{\text{David}}$ will be the bits s_0 and s_1 , a random bit b , a message from Eve denoted by e_i , and a state state_{i-1} (since $\mathcal{F}^{\text{David}}$ is a reactive functionality, it needs to know state_{i-1} in order to compute state_i), and the output of $\mathcal{F}^{\text{David}}$ will be David's i^{th} message

in $\widehat{\Pi}_{\text{OT}}$ and state_i . Thus, Eve can, as before, play the receiver in an execution of Π_{OT} with Alice as the sender and carry out the man in the middle attack by invoking $\mathcal{F}^{\text{David}}$. We will denote the real world attacker played by Eve as $\hat{\mathcal{E}}^{\text{real}}$.

As the next step, we will replace the ideal calls to $\mathcal{F}^{\text{David}}$ made by $\hat{\mathcal{E}}^{\text{real}}$ by a series of OTs executed between Alice and $\hat{\mathcal{E}}^{\text{real}}$. Let n denote the number of messages that David sends in protocol $\widehat{\Pi}_{\text{OT}}$. Then, consider the one time programs $\text{OTP-msg}_1, \dots, \text{OTP-msg}_n$ where OTP-msg_i computes the i^{th} next message function of David. We will provide Alice with all the keys $\{\text{keys}_i\}_{i=1}^n$ to the OTPs. Then, to evaluate the i^{th} OTP, $\hat{\mathcal{E}}^{\text{real}}$ executes (multiple sessions of) Π_{OT} with Alice to obtain the keys corresponding to her input.

Also note that OTPs are *stateless* by definition, but David's next message functionality is *stateful*. We handle this by allowing each OTP-msg_i to pass its private state to OTP-msg_{i+1} using standard techniques. Specifically, we will add to the (fixed) inputs of OTP-msg_i a key K for an authenticated encryption scheme. Now, OTP-msg_i outputs not only David's next message d_i , but also an authenticated encryption of its resultant state, denoted by $\tau_i = \text{Enc}_K(\text{state}_i)$. As input, OTP-msg_i requests not only message e_i , but also a valid authenticated encryption τ_{i-1} such that $\text{Dec}_K(\tau_{i-1}) = \text{state}_{i-1}$. This forces the functionality $\mathcal{F}^{\text{David}}$ implemented by OTPs, to be invoked in proper order. For the rest of the article, we will assume that any OTPs we use are made “stateful” in this way.

Note that there are two kinds of executions of Π_{OT} being carried out between Alice and $\hat{\mathcal{E}}^{\text{real}}$: the “main” OT execution where Alice uses inputs s_0, s_1 , and the additional OT executions that allow Eve to obtain keys for the one time programs. For clarity of exposition, we will refer to the “main” execution as $\Pi_{\text{OT}}^{\text{main}}$ and each of the remaining ones as $\Pi_{\text{OT}}^{\text{David}}$.

Attack in the Real World. Now, we will describe the explicit execution of OTs between Alice and $\hat{\mathcal{E}}^{\text{real}}$, where $\hat{\mathcal{E}}^{\text{real}}$ recovers $\text{secret} = s_{\bar{b}}$. The protocol is as follows:

Alice's program: Alice is given input bits s_0, s_1 for $\Pi_{\text{OT}}^{\text{main}}$ and all the one time program keys $\{\text{keys}_i\}_{i=1}^n$ for $\text{OTP-msg}_1, \dots, \text{OTP-msg}_n$. She behaves honestly according to the protocol Π_{OT} and responds honestly to all OT invocations made by $\hat{\mathcal{E}}^{\text{real}}$.

$\hat{\mathcal{E}}^{\text{real}}$'s Program: $\hat{\mathcal{E}}^{\text{real}}$ is given input bit \hat{b} for $\Pi_{\text{OT}}^{\text{main}}$ and the one time programs $\{\text{OTP-msg}_i\}_{i=1}^n$ where OTP-msg_i computes the i^{th} next message function of David. Let s_0, s_1 and b denote the fixed inputs (hardwired) in the OTPs such that $\text{secret} = s_{\bar{b}}$. For $i = 1, \dots, n$, do:

1. Upon receiving i^{th} message from Alice in $\Pi_{\text{OT}}^{\text{main}}$, say a_i , suspend (temporarily) the ongoing $\Pi_{\text{OT}}^{\text{main}}$ session and start a new $\Pi_{\text{OT}}^{\text{David}}$ session with Alice to compute the i^{th} message that David would have sent in response had he received a_i from Eve. Depending on a_i , retrieve the corresponding keys keys_i from Alice to input to OTP-msg_i . End the $\Pi_{\text{OT}}^{\text{David}}$ protocol.
2. Run OTP-msg_i with keys keys_i and obtain output d_i .
3. If $i \leq n - 1$, resume the suspended $\Pi_{\text{OT}}^{\text{main}}$ protocol and send d_i back to Alice as response. Otherwise, output the value secret received from OTP-msg_n .

Thus, using a sequence of OT executions, a dishonest $\hat{\mathcal{E}}^{\text{real}}$ is able to recover $\text{secret} = s_{\bar{b}}$ with probability 1.

Infeasibility of Ideal World Attacker. Suppose for contradiction that there exists an ideal world attacker $\hat{\mathcal{E}}^{\text{ideal}}$ that succeeds in outputting secret with probability $1 - \text{negl}$. Then, we can construct a stand-alone cheating sender that executes Π_{OT} with an honest receiver \mathcal{R} and uses $\hat{\mathcal{E}}^{\text{ideal}}$ to learn \mathcal{R} 's secret input bit b with non-negligible probability. Please see the full version for details.

5 Putting It All Together

By combining the results from the previous sections, we now state our final results. All functionalities referred to below are 2-party finite deterministic non-reactive functionalities. For brevity and clarity we drop these qualifiers.

In the full version, we prove the following composition theorem for security under concurrent self-composition in the static-input, fixed-role setting.

Theorem 3. *Suppose $\Pi^{\mathcal{F}}$ is an asynchronous, non-interactive protocol that securely realizes \mathcal{G} under concurrent self-composition in the static-input, fixed-role setting, and is secure against expected PPT adversaries. Also, suppose ρ is a protocol (in the plain model) that securely realizes \mathcal{F} under concurrent self-composition in the static-input, fixed-role setting. Then Π^{ρ} securely realizes \mathcal{G} under concurrent self-composition in the static-input, fixed-role setting.*

Since UC-security implies security under concurrent self-composition in the static-input, fixed-role setting, by composing the OT protocol in Theorem 1 with a hypothetical protocol for any non-trivial asymmetric functionality \mathcal{F} (using Theorem 3), we will obtain a protocol for \mathcal{F}_{OT} , contradicting Theorem 2. This gives our main impossibility result:

Theorem 4. *For any non-trivial asymmetric functionality \mathcal{F} , there does not exist a protocol (in the plain model) that securely realizes \mathcal{F} under self-composition even in the static-input, fixed-role setting. (On the other hand, every trivial asymmetric functionality has a UC-secure protocol.)*

Another consequence of the protocol in Theorem 1 is to give a characterization of functionalities that are *non-interactively complete* against active adversaries. This is because \mathcal{F}_{OT} itself has this property, as was shown by the following non-interactive (but not asynchronous) protocol from [19].

Theorem 5. *[19, Full version] For any asymmetric functionality \mathcal{G} , there exists a non-interactive protocol $\Phi^{\mathcal{F}_{\text{OT}}}$ that UC-securely realizes \mathcal{G} , even against computationally unbounded adversaries.*

Since the protocols in our positive results above are UC-secure, by the UC theorem their composition is secure. Further, composing a non-interactive protocol in \mathcal{F}_{OT} -hybrid with a non-interactive protocol for \mathcal{F}_{OT} in \mathcal{F} -hybrid gives a non-interactive protocol in \mathcal{F} -hybrid. This gives us the following characterization:

Theorem 6. *In the class of asymmetric functionalities, every non-trivial functionality is non-interactively complete with respect to UC security (against active adversaries). That is, for any two asymmetric functionalities \mathcal{F}, \mathcal{G} , if \mathcal{F} is non-trivial, then there exists a non-interactive protocol $\Psi^{\mathcal{F}}$ that UC-securely realizes \mathcal{G} , even against computationally unbounded adversaries.*

Acknowledgements. Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1136174, 1118096, 1065276, 0916574, 0830803 and 0747027, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, NSF or the U.S. Government.

References

1. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS (2004)
2. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS (2006)
3. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS (2005)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS (2001)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols (2005), <http://eprint.iacr.org/2000/067>
6. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
7. Canetti, R., Kushilevitz, E., Lindell, Y.: On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)
8. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: FOCS (2010)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC (2002)
10. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: FOCS (2007)
11. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: STOC (1998)
12. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently Secure Computation in Constant Rounds. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (2012)
13. Garg, S., Kumarasubramanian, A., Ostrovsky, R., Visconti, I.: Impossibility Results for Static Input Secure Computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 419–436. Springer, Heidelberg (2012)
14. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: STOC (1987)

15. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-Time Programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
16. Goyal, V.: Positive results for concurrently secure computation in the plain model. IACR Cryptology ePrint Archive 2011, 602 (2011)
17. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding Cryptography on Tamper-Proof Hardware Tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010)
18. Goyal, V., Jain, A., Ostrovsky, R.: Password-Authenticated Session-Key Generation on the Internet in the Plain Model. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 277–294. Springer, Heidelberg (2010)
19. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008), full version on <http://www.cs.uiuc.edu/~mmp/>
20. Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
21. Kidron, D., Lindell, Y.: Impossibility results for universal composable in public-key models and with fixed inputs. *J. Cryptology* 24(3) (2011)
22. Kilian, J.: More general completeness theorems for secure two-party computation. In: STOC (2000)
23. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-logarithm rounds. In: STOC (2001)
24. Kraschewski, D., Müller-Quade, J.: Completeness Theorems with Constructive Proofs for Finite Deterministic 2-Party Functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 364–381. Springer, Heidelberg (2011)
25. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composable from stand-alone non-malleability. In: STOC (2009)
26. Lindell, Y.: General composition and universal composable in secure multi-party computation. In: FOCS (2003)
27. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
28. Lindell, Y.: Lower bounds and impossibility results for concurrent self composition. *J. Cryptology* 21(2) (2008)
29. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: FOCS (2006)
30. Pass, R.: Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
31. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS (2002)
32. Prabhakaran, M., Rosulek, M.: Cryptographic Complexity of Multi-Party Computation Problems: Classifications and Separations. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 262–279. Springer, Heidelberg (2008)
33. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composable without trusted setup. In: STOC (2004)
34. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–431. Springer, Heidelberg (1999)
35. Yao, A.C.: How to generate and exchange secrets. In: FOCS (1986)

Black-Box Constructions of Composable Protocols without Set-Up

Huijia Lin^{1,*} and Rafael Pass^{2,**}

¹ MIT and Boston University

huijia@csail.mit.edu

² Cornell University

rafael@cs.cornell.edu

Abstract. We present the first *black-box* construction of a secure multi-party computation protocol that satisfies a meaningful notion of *concurrent security* in the plain model (without any set-up, and without assuming an honest majority). Moreover, our protocol relies on the minimal assumption of the existence of a semi-honest OT protocol, and our security notion “UC with super-polynomial helpers” (Canetti et al, STOC’10) is closed under universal composition, and implies super-polynomial-time simulation security.

1 Introduction

The notion of *secure multi-party computation* allows m mutually distrustful parties to securely compute (or, *realize*) a functionality $f(\bar{x})$ of their corresponding private inputs $\bar{x} = x_1, \dots, x_m$, such that party P_i receives the i^{th} component of $f(\bar{x})$. Loosely speaking, the security requirements are that the output of each party is distributed according to the prescribed functionality—this is called *correctness*—and that even malicious parties learn nothing more from the protocol than their prescribed output—this is called *privacy*. These properties should hold even in case that an arbitrary subset of the parties maliciously deviates from the protocol.

Soon after the concept was proposed [47], general constructions were developed that appeared to satisfy the intuitive correctness and secrecy for practically any multi-party functionality [47,19]. These constructions require only authenticated communication and can use any enhanced trapdoor permutation. However, definitions that capture the security properties of secure multi-party computation protocols (and, in fact, of secure cryptographic protocols in general) took more time to develop. Here, the *simulation paradigm* emerged as a natural approach: Originally developed for capturing the security of encryption and then extended to Zero-Knowledge [21,22]. The idea is to say that a protocol π securely realizes f if running π “emulates” an idealized process where all parties secretly provide inputs to an imaginary trusted party that computes f and returns the outputs to the parties; more precisely, any “harm” done by

* Supported in part by a DARPA Grant FA8750-11-2-0225, NSF Grant CCF-1018064.

** Supported in part by a Microsoft New Faculty Fellowship, NSF CAREER Award CCF-0746990, AFOSR Award FA9550-08-1-0197 and BSF Grant 2006317.

a polynomial-time adversary in the real execution of π , could have been done even by a polynomial-time adversary (called a *simulator*) in the ideal process. The simulation paradigm provides strong security guarantees: It ensures that running the protocols is “as good as” having a trusted third party computing the functionality for the players, and an adversary participating in the real execution of the protocols does not gain any “computational advantage” over the simulator in the ideal process (except from polynomial time advantage). We call this definition *basic security*.

The original setting in which secure multi-party protocols were investigated, however, only allowed the execution of a single instance of the protocol at a time; this is the so called stand-alone setting. A more realistic setting, is one which allows the concurrent execution of protocols. In the concurrent setting, many protocols are executed at the same time. This setting presents a new risk of a “coordinated attack” in which an adversary interleaves many different executions of a protocol and chooses its messages in each instance based on other partial executions of the protocol. To prevent coordinated attacks, we require the following basic security guarantee:

Concurrent Security: The security properties, correctness and privacy, of the *analyzed protocol* should remain valid even when multiple instances of the protocol are concurrently executed in a potentially unknown environment.

Another natural desideratum is the capability of supporting modular design of secure protocols.

Modular Analysis: The notion of security should support designing composite protocols in a modular way, while preserving security. That is, there should be a way to deduce security properties of the overall protocol from security properties of its components. This is essential for asserting security of complex protocols.

Unfortunately, these properties are not implied by the basic security. In the literature, the strongest and also the most realistic formalization of concurrent security is the notion of Universal Composability (UC) [5]: It considers the concurrent execution of an unbounded number of instances of the analyzed protocol, in an arbitrary, and adversarially controlled, network environment. It also supports modular analysis of protocols. But, these strong properties come at a price: Many natural functionalities cannot be realized with UC security in the *plain model*, where players only have access to authenticated communication channels; some additional trusted set-up is necessary [7,8]; furthermore, the need for additional trusted set up extends to any protocol that only guarantees a concurrent extension of basic security [35]. A large body of works (e.g. [10,1,28,11,24,29,6]) have shown that indeed, with the appropriate trusted set-ups, UC-security becomes feasible. However, in many situations, trusted set-up is hard to come by (or at least expensive). It is thus important to have a notion of concurrent security that can be achieved in the plain model. Several notions of concurrent security have since been proposed.

Concurrent Security in the Plain Model. *Security with super-polynomial simulators (SPS)* [39] is a relaxation of UC security that allows the adversary in the ideal execution to run in super-polynomial time. Informally, this corresponds to guaranteeing that “any polytime attack that can be mounted against the protocol can also be mounted in the ideal execution—albeit with super-polynomial resources.” Although SPS security is sometimes weaker than basic security, it often provides an adequate level of security. In contrast to basic security, however, SPS directly considers security in the concurrent setting. Protocols that realize practically any functionality with SPS security in the plain model were shown based on sub-exponential hardness assumptions [39,2,33]. Very recently, improved constructions are presented [9,17,34] that are based on only standard polynomial-time hardness assumptions. Another notion of security that is closely related to SPS security is input indistinguishability. It is shown in [38] that input indistinguishable protocols for general functionalities can be constructed from standard polynomial time hardness assumptions.

One drawback of SPS security is that it is not closed under composition; thus it is not a convenient basis for modular analysis of protocols. *Angel-based UC security* [43] is a framework for notions of security that provides similar security guarantees as SPS and at the same time supports modular analysis. Specifically, angel-based security considers a model where both the adversary and the simulator have access to an oracle (an “angel”) that allows some judicious use of super-polynomial resources. Since the angels can be implemented in super-polynomial time, for any angel, angel-based security implies SPS security. Furthermore, akin to UC security, angel-based UC security, with any angel, can be used as a basis for modular analysis. Prabhakaran and Sahai [43] exhibited an angle with respect to which practically all functionalities can be securely realized; later another angle is given by [37]; both constructions, however, rely on some non-standard hardness assumptions.

Recently, Canetti, Lin and Pass [9] proposed a new notion of security, called *UC with super-polynomial time helpers*. This notion is very similar to the angel-based security where both the adversary and the simulator have access to a helper that provides some super-polynomial time help through a limited interface. Like angel-based security, UC security with super-polynomial time helpers implies SPS security. But, unlike angel-based security where angels are non-interactive and stateless, the helpers are *highly interactive and stateful*. Canetti, Lin and Pass [9] then constructed protocols that realize practically all functionalities with respect to a particular super-polynomial-time interactive helper, based on the existence of enhanced trapdoor permutations.

Summarizing the state-of-the-art, there are constructions [9,17,34] of protocols satisfying a meaningful notion of concurrent security—SPS security—in the plain model based on standard polynomial time hardness assumptions. Furthermore, the construction of [9] also supports modular analysis. (The constructions of [17,34] are better in terms of round-complexity—they only require a constant number of communication rounds—but they only achieve “non-composable” SPS security).

However, all these constructions are *non-black-box*, that is, the constructed protocols make non-black-box use of the underlying primitives. In fact, these constructions all follow the “Feige-Shamir” paradigm [16]: The protocols contain “trapdoors” embedded into the messages of the protocol, allowing a super-polynomial time simulator to extract the trapdoor and simulate messages in the protocol by proving that “it knows the trapdoor”. In general, protocols following this approach seem hard to turn into a “practical” protocol for secure computations; as such, there results should only be viewed as “feasibility results” regarding concurrent secure computation without set-ups, but not candidates for practical purposes.

In contrast, black-box constructions that only use the underlying primitives through their input/output interfaces, are often much more efficient and are more suitable for implementation. Therefore, a series of recent works [14, 26, 27, 36, 46, 23] have focused on constructing *black-box construction of secure computation protocols*, as an important step towards bringing secure multi-party computation closer to the practice. However, their constructions are all in either the stand-alone setting or rely on strong trusted set-ups (e.g., trusted hardware). This leaves open the following basic questions:

Can we obtain a black-box construction of concurrently secure protocols in the plain model (preferably based only standard polynomial-time assumptions)?

Can we have such a black-box construction that also satisfies a notion of security supporting composability?

1.1 Our Results

We present a black-box construction of protocols that satisfy UC security with super-polynomial time helper for a specific helper, based on the existence of a stand-alone semi-honest oblivious transfer (OT) protocols. The framework of UC with super-polynomial time helper of [9] is formalized through the extended UC (EUC) framework of [6]; it is identical to the standard UC model [4] except that the corrupted parties (and the environment) have access to an super-polynomial time entity \mathcal{H} , called a helper functionality.

Main Theorem (Informally Stated): *Assume the existence of stand-alone semi-honest oblivious transfer protocols. Then there exists a sub-exponential-time computable interactive machine \mathcal{H} such that for any “well-formed”¹ polynomial-time functionality \mathcal{F} , there exists a protocol that realizes \mathcal{F} with \mathcal{H} -EUC security, in the plain model. Furthermore, the protocol makes only black-box calls to the underlying oblivious transfer protocol.*

As far as we know, this is the first black-box construction of secure multi-party computation protocols that achieve any non-trivial notion of concurrent security in the plain model (without any trusted-set up, and without assuming an honest majority).

¹ See [10] for a definition of well-formed functionalities.

The main technical tool used in our construction is a new notion of a commitment that is secure against adaptive Chosen Commitment Attack (CCA security). The notion of CCA secure commitments was previously introduced in [9]. Roughly speaking, a tag-based commitment scheme (i.e., commitment scheme that take an identifier—called the tag—as an additional input) is said to be *CCA-secure* if the value committed to using the tag id remains hidden even if the receiver has access to a (super-polynomial time) oracle that “breaks” commitments using any tag $\text{id}' \neq \text{id}$, where by breaking, it means the oracle returns a decommitment of the commitment. Thus the oracle is called a decommitment oracle. In [9], a commitment scheme that is CCA-secure w.r.t. a decommitment oracle is constructed based on the minimal assumption of one-way functions. However, their construction is non-black-box. In this work, to obtain black-box secure computation protocols, we need a new *black-box* construction of a CCA-secure commitment scheme. Towards this, we weaken the notion of CCA security w.r.t. decommitment oracle to instead consider an oracle that “breaks” commitments by returning only the unique committed value² (instead of the decommitment information); we call this the committed-value oracle. We then provide a black-box construction of a commitment scheme that is CCA-secure w.r.t. the committed-value oracle.

Theorem (Informally Stated): *Assume the existence of one-way functions. Then, for every $\epsilon > 0$, there exists an $O(n^\epsilon)$ -round commitment scheme that is CCA-secure w.r.t. the committed-value oracle and only relies on black-box access to one-way functions (where n is the security parameter).*

1.2 Outline

In Section 2, we define the notion of CCA-security w.r.t. the committed-value oracle. In Section 3, we first reduce the task of achieving UC security with super-polynomial time helpers to the task of constructing a UC-OT (with super-polynomial time helpers); we then sketch our construction of the UC-OT protocol, using CCA-secure commitments. Finally, in Section 4, we present our black-box robust CCA-secure commitment scheme.

2 Definition of CCA-Secure Commitments

We assume familiarity with the definition of commitment schemes and the statistically/computational binding and statistically/computational hiding properties. Unless specified otherwise, by a commitment scheme, we mean one that is statistically binding and computationally hiding. A *tag-based commitment schemes* with $l(n)$ -bit identities [40,15] is a commitment scheme where, in addition to the security parameter 1^n , the committer and the receiver also receive a “tag”—a.k.a. the identity— id of length $l(n)$ as common input.

² The oracle returns \perp if there is no unique committed value.

2.1 CCA-Security w.r.t. Committed Value Oracle

Let $\langle C, R \rangle$ be a tag-based commitment scheme with $l(n)$ -bit identities. A committed-value oracle \mathcal{O} of $\langle C, R \rangle$ acts as follows in interaction with an adversary A : It participates with A in many sessions of the commit phase of $\langle C, R \rangle$ as an honest receiver, using identities of length $l(n)$, chosen adaptively by A . At the end of each session, if the session is *valid*, it reveals the unique committed value of that session to A ; otherwise, it sends \perp . (If a session has multiple committed values, the committed-value oracle also returns \perp . The statistically binding property guarantees that this happens with only negligible probability.) Loosely speaking, a tag-based commitment scheme $\langle C, R \rangle$ is said to be CCA-secure w.r.t. the committed-value oracle, if the hiding property of the commitment holds even with respect to adversaries with access to the committed-value oracle \mathcal{O} . More precisely, denote by $A^{\mathcal{O}}$ the adversary A with access to the committed-value oracle \mathcal{O} . Let $\text{IND}_b(\langle C, R \rangle, A, n, z)$, where $b \in \{0, 1\}$, denote the output of the following probabilistic experiment: on common input 1^n and auxiliary input z , $A^{\mathcal{O}}$ (adaptively) chooses a pair of challenge values $(v_0, v_1) \in \{0, 1\}^n$ —the values to be committed to—and an identity $\text{id} \in \{0, 1\}^{l(n)}$, and receives a commitment to v_b using identity id . Finally, the experiment outputs the output y of $A^{\mathcal{O}}$; the output y is replaced by \perp if during the execution A sends \mathcal{O} any commitment using identity id (that is, any execution where the adversary queries the decommitment oracle on a commitment using the same identity as the commitment it receives, is considered invalid).

Definition 1 (CCA-secure Commitments). Let $\langle C, R \rangle$ be a tag-based commitment scheme with $l(n)$ -bit identities. We say that $\langle C, R \rangle$ is CCA-secure w.r.t. the committed-value oracle, if for every PPT ITM A , the following ensembles are computationally indistinguishable:

- $\{\text{IND}_0(\langle C, R \rangle, A, n, z)\}_{n \in N, z \in \{0, 1\}^*}$
- $\{\text{IND}_1(\langle C, R \rangle, A, n, z)\}_{n \in N, z \in \{0, 1\}^*}$

2.2 k -Robustness w.r.t. Committed-Value Oracle

Consider a man-in-the-middle adversary that participates in an *arbitrary* left interaction with a *limited number of rounds*, while having access to a committed-value oracle. Roughly speaking, $\langle C, R \rangle$ is k -robust if the (joint) output of every k -round interaction with an adversary having access to the oracle \mathcal{O} , can be simulated without the oracle. In other words, having access to the oracle does not help the adversary in participating in any k -round protocols much.

Definition 2. Let $\langle C, R \rangle$ be a tag-based commitment scheme with $l(n)$ -bit identities. We say that $\langle C, R \rangle$ is k -robust w.r.t. the committed-value oracle, if there exists a simulator S , such that, for every PPT adversary A , the following two conditions hold.

Simulation: For every PPT k -round ITM B , the following two ensembles are computationally indistinguishable.

- $\{\text{output}_{B,A^O}[(B(y), A^O(z))(1^n, x)]\}_{n \in N, x, y, z \in (\{0,1\}^*)^3}$
- $\{\text{output}_{B,S^A}[(B(y), S^{A(z)})(1^n, x)]\}_{n \in N, x, y, z \in (\{0,1\}^*)^3}$

where $\text{output}_{A,B}[(B(y), A(z))(x)]$ denote the joint output of A and B in an interaction between them, on common input x and private inputs z to A and y to B respectively, with uniformly and independently chosen random inputs to each machine.

Efficiency: There exists a polynomial t and a negligible function μ , such that, for every $n \in N$, $z \in \{0,1\}^*$ and $x \in \{0,1\}^*$, and every polynomial T , the probability that S with oracle access to $A(z)$ and on input $1^n, x$, runs for more than $T(n)$ steps is smaller than $\frac{t(n)}{T(n)} + \mu(n)$.

The following proposition shows that to construct a k -robust CCA-secure commitment scheme for identities of length n , it suffices to construct one for identities of length $\ell(n) = n^\varepsilon$. The same proposition is established in [9] for robust CCA-security w.r.t. decommitment oracles, and the proof there also applies to CCA-security w.r.t. committed-value oracles; we omit the proof here.

Proposition 1. Let ε be any constant such that $0 < \varepsilon < 1$, ℓ a polynomial such that $\ell(n) = n^\varepsilon$, and $\langle C, R \rangle$ a γ -round k -robust CCA-secure commitment scheme (w.r.t. the committed-value oracle) with ℓ -bit identities. Then assuming the existence of one-way functions, there exists a $\gamma + 1$ -round k -robust CCA-secure commitment scheme $\langle \hat{C}, \hat{R} \rangle$ (w.r.t. the committed-value oracle) with n -bit identities.

3 BB UC-Secure Protocols with Super-Poly Helpers

We consider the model of UC with super-polynomial helper introduced in [9]. At a very high-level, this model is essentially the same as the UC-model introduced by [4], except that both the adversary and the environment in the real and ideal worlds have access to a super-polynomial time functionality that acts as a helper. See [9] for a formal definition of the model. In this section, we show:

Theorem 1. Let δ be any positive constant. Assume the existence of a T'_{OT} -round stand-alone semi-honest oblivious transfer protocol. Then there exists a super-polynomial time helper functionality \mathcal{H} , such that, for every well-formed functionality \mathcal{F} , there exists a $O(\max(n^\delta, T'_{OT}))$ -round protocol Π that \mathcal{H} -EUC-emulates \mathcal{F} . Furthermore, the protocol Π only uses the underlying oblivious transfer protocol in a black-box way.

3.1 Overview of Our Construction

Towards Theorem 1, we need to first exhibit a super-polynomial time helper functionality \mathcal{H} . Roughly speaking, \mathcal{H} simply acts as the committed-value oracle of a CCA secure commitment scheme. More precisely, consider the following two building blocks: First, given any $T'_{OT}(n)$ -round stand-alone semi-honest OT

protocol, it follows from previous works [26,25] that there exists an $T_{OT}(n)$ -round OT protocol $\langle S, R \rangle$ that is secure against a malicious sender and a semi-honest receiver—called mS-OT protocol for short—that only relies on black-box access to the semi-honest OT protocol; furthermore $T_{OT} = O(T'_{OT}(n))$. Second, we need a $T_{OT}(n)$ -robust CCA-secure commitment scheme $\langle C, R \rangle$, whose committed-value oracle \mathcal{O} can be computed in sub-exponential time.³ As we will show in the next section such a protocol exists with $O(\max(T_{OT}, n^\delta)) = O(\max(T'_{OT}, n^\delta))$ rounds, relying on the underlying OWF in a black-box way. Since OWFs can be constructed from a semi-honest OT protocol in a black-box way. Therefore, we have that the second building block can also be based on the semi-honest OT protocols in a black-box way.

Consider a helper functionality \mathcal{H} that “breaks” commitments of $\langle C, R \rangle$ in the same way as its committed-value oracle \mathcal{O} does, subject to the condition that player P_i can only query the functionality on commitments that uses identity P_i . More precisely, every party P_i in a secure computation can simultaneously engage with \mathcal{H} in multiple sessions of the commit phase of $\langle C, R \rangle$ as a committer using identity P_i , where the functionality simply forwards all the messages internally to the committed-value oracle \mathcal{O} , and forwards P_i the committed value returned from \mathcal{O} at the end of each session. Since the committed-value oracle \mathcal{O} can be computed in sub-exponential time, this functionality can also be implemented in sub-exponential time.

We show that Theorem 1 holds w.r.t. the helper functionality defined above in two steps. First, note that to realize any well-formed functionality in a black-box way, it suffices to realize the *ideal oblivious transfer functionality* \mathcal{F}_{OT} . This is because it follows from previous works [31,3,20,27] that every functionality can be UC securely implemented in the \mathcal{F}_{OT} -hybrid model, even w.r.t. super-polynomial time environments. Based on previous works, [9] further shows that by considering only dummy adversaries and treating environments with access to a super-polynomial functionality \mathcal{H} as sub-exponential time machines, we have that every functionality can be \mathcal{H} -EUC securely implemented in the \mathcal{F}_{OT} model. Formally, we have the following lemma from [9].

Lemma 2. *Fix any super-polynomial time functionality \mathcal{H} . For every well-formed functionality \mathcal{F} , there exists a constant-round \mathcal{F}_{OT} -hybrid protocol that \mathcal{H} -EUC-emulates \mathcal{F} .*

Next we show how to implement the \mathcal{F}_{OT} functionality in the \mathcal{H} -EUC model. Then combining with Lemma 2, we conclude Theorem 1.

Lemma 3. *Let δ be any positive constant. Assume the existence of a T'_{OT} -round semi-honest oblivious transfer protocol. Then there exists a $O(\max(n^\delta, T'_{OT}))$ -round protocol Π_{OT} that \mathcal{H} -EUC-emulates \mathcal{F}_{OT} . Furthermore, the protocol Π_{OT} only uses the underlying oblivious transfer protocol in a black-box way.*

³ This can be instantiated by simply using a normal T_{OT} -robust CCA secure commitment that has an exponential time committed value \mathcal{O} , with a “scaled-down” security parameter.

3.2 Overview of the OT Protocol Π_{OT}

In this section we provide an overview of our black-box construction of \mathcal{H} -EUC secure OT protocol Π_{OT} . Our construction is based on the black-box construction of an OT protocol secure against malicious players from a mS-OT protocol of [26,25]. Roughly speaking, the protocol of [26,25], relying on a stand-alone mS-OT protocol $\langle S, R \rangle$, proceeds in the following four stages:

Stage 1 (Receiver’s Random Tape Generation). The sender and the receiver jointly decide the receiver’s inputs and random tapes in Stage 2 using $2n$ parallel “coin tossing in the well” executions.

Stage 2 (OT with Random Inputs). The sender and the receiver perform $2n$ parallel OT executions of $\langle S, R \rangle$ using *random* inputs (s_j^0, s_j^1) and r_j respectively, where the receiver’s inputs r_j ’s (and its random tapes) are decided in Stage 1.

Stage 3 (Cut-and-Choose). A random subset $Q \subset [2n]$ of n locations is chosen using a 3-round coin-tossing protocol where the sender commits to a random value first. (Thus the receiver knowing that random value can bias the coin-tossing output.) The receiver is then required to reveal its randomness in Stage 1 and 2 at these locations, which allows the sender to check whether the receiver behaved honestly in the corresponding OT executions. The randomness of the receiver at the rest of locations remains hidden.

Stage 4 (OT Combiner). Finally, for these locations $j \notin Q$ that are not open, the receiver sends $\alpha_j = u \oplus c_j$ where u is the receiver’s true input. The sender replies with $\beta_0 = v_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $\beta_1 = v_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$. The honest receiver obtains $s_j^{\alpha_j}$ ’s through the OT execution, and thus can always recover v_u .

At a very high-level, the protocol of [26,25] augments security of the mS-OT protocol $\langle S, R \rangle$ to handle malicious receivers, by adding the cut-and-choose (as well as the random tape generation) stage to enforce the adversary behaving honestly in *most* (Stage 2) OT executions. (This is in a similar spirit as the non-black-box approach of requiring the receiver to prove that it has behaved honestly.) Then the security against malicious receivers can be based on that against semi-honest receivers of $\langle S, R \rangle$.

Wee [46] further augmented the stand-alone security of the protocol of [26,25] to achieve parallel security, that is, obtaining a protocol that is secure against man-in-the-middle adversaries that simultaneously acts as sender and receiver in many parallel executions. Towards this, Wee instantiated the commitments used in coin-tossing in Stage 3 of the above protocol, with ones that satisfy a notion of “non-malleability w.r.t. extraction”. Roughly speaking, non-malleability w.r.t. extraction [46] is a weaker notion than non-malleability of [15,32]; it guarantees that no matter what values the adversary is receiving commitments to, the committed values extracted out of the commitments from the adversary (with over-extraction) are indistinguishable. This guarantees that a simulator can bias the coin-tossing output by extracting the committed values from the adversary while the adversary cannot, as otherwise, by non-malleability w.r.t. extraction, it

could do so even if the honest player sends a commitment to 0 instead of its true random challenge q . However, this is impossible as in this case no information of q is revealed. In other words, the coin-tossing protocol when instantiated with a non-malleable w.r.t. extraction commitment becomes parallel secure; Wee then relies on the parallel security of the coin-tossing protocol to show the parallel security of the OT protocol.

Towards \mathcal{H} -EUC-Secure OT Protocols, we need to further overcome two problems. First, we need to go from parallel security to concurrent security. In other words, we need a coin-tossing protocol that is concurrently secure. Informally speaking, non-malleability w.r.t. extraction guarantees that the simulator can extract the committed values of commitments from the adversary (to bias the output of the coin-tossing) while keeping the commitment to the adversary hiding amid rewinds (to ensure that the adversary cannot bias the output). However, this only holds in the parallel setting, as non-malleability only guarantees hiding of a commitment when values of the commitments from the adversary are extracted in parallel at the end of the execution. But, in the concurrent setting, the simulator needs to extract the committed values from the adversary in an on-line manner, that is, whenever the adversary successfully completes a commitment the committed value needs to be extracted. To resolve this problem, we resort to CCA-secure commitments, which guarantees hiding of a commitment even when the committed values are extracted (via the committed-value oracle) concurrently and immediately after each commitment. Now, instantiating the commitment scheme in the coin-tossing protocols with a CCA-secure commitment yields a coin-tossing protocol that is concurrently secure.

The second problem is that to achieve \mathcal{H} -EUC-security (similar to UC-security), we need to design a protocol that admits straight-line simulation. The simulator of a OT protocol has three tasks: It needs to simulate the messages of the honest senders and receivers, extract a choice from the adversary when it is acting as a receiver, and extract two inputs when it is acting as a sender. To achieve the first two tasks, the original simulation strategy in [26,25,46] relies on the capability of breaking the non-malleable commitments from the adversary using rewinds. When using CCA-secure commitments, the simulator can extract the committed values in a straight-line, by forwarding the commitment from the adversary to the helper functionality \mathcal{H} that breaks the CCA commitments using brute force. For the last task, the original simulation strategy uses the simulator of the mS-OT protocol $\langle S, R \rangle$ against malicious senders to extract the adversary's inputs s_j^b 's in all the Stage 3 OT executions, which then allows extraction of the real inputs v_0 and v_1 from the last message. However, the simulator of the mS-OT protocol may use rewinds. To solve this, one way is to simply assume a mS-OT protocol that has a straight-line simulator. We here however, present a different solution.

In our protocol, the sender and the receiver participate in parallel “coin tossing in the well” executions to decide the sender's random inputs s_j^b (and random tapes) in the parallel OT executions (besides the receiver's inputs and random tapes). Since the simulator can bias the coin-tossing in a straight line, it can

determine the sender’s inputs s_j^b ’s, which allows extraction of the sender’s true inputs. For this to work, we need to make sure that a malicious sender would indeed use the outputs of coin-tossing as inputs in the OT executions. Towards this, we again use the cut-and-choose technique: After the OT execution, the sender is required to reveal its randomness in the coin-tossing and OT execution at a randomly chosen subset of locations. The cut-and-choose technique guarantees that a malicious sender will behave consistently in most OT executions. Therefore the simulator extracts the inputs s_j^b ’s correctly at *most* locations. However, in the protocol of [26,25,46], to recover the real inputs v_0 and v_1 , the simulator needs to obtain *all* s_j^b ’s correctly. To bridge the gap, we modify the protocol to have the sender compute a random secret-sharing $\{a_j^b\}$ of each input v_b and hide each share using the appropriate s_j^b , that is, it sends $a_j^b \oplus s_j^{b \oplus \alpha}$ for every j (that is not open in the cut-and-choose procedures). Then, the simulator, able to extract most s_j^b ’s correctly, can recover enough shares to decode to the real inputs correctly. In contrast, a malicious receiver that is enforced to behave honestly in most OT executions by the cut-and-choose procedure, cannot obtain enough shares for both inputs and thus can only recover one of them. Finally, we remark that as in [46], to avoid over-extraction from the secret shares, we use the technique used in [12,13], which adds another cut-and-choose procedure. We defer the formal description of our OT protocol and its security proof (i.e., proof of Lemma 3) to the full version.

4 Black-Box Robust CCA-Secure Commitments

In this section, we present a *black-box* construction of a robust CCA-secure commitment scheme w.r.t. committed-value oracle based on one-way functions. For simplicity of exposition, the presentation below relies on a non-interactive statistically binding commitment scheme com ; this can be replaced with a standard 2-round statistically binding commitment scheme using standard techniques⁴.

4.1 Building Blocks

Our construction makes use of previous black-box constructions of extractable commitments and trapdoor commitment scheme. So let’s start by reviewing them.

Extractable Commitments. Intuitively, an extractable commitment is one such that for any machine C^* sending a commitment, a committed value can be extracted from C^* if the commitment it sends is valid; otherwise, if the commitment is invalid, then no guarantee is provided, that is, an arbitrary garbage value may be extracted. This is known as the “over-extraction” problem. As shown

⁴ This can be done by sending the first message of a 2-round commitment scheme at the beginning of the protocol, and using the second message of the 2-round commitment scheme w.r.t. that first message as a non-interactive commitment in the rest of the protocol.

in [41], the following protocol used in the works of [15,42,45] (also [30]) yields a black-box extractable commitment scheme **ExtCom**: To commit to a value $v \in \{0, 1\}^m$, the committer and receiver on common input a security parameter 1^n , proceed as follows:

Commit: The committer finds n pairs of random shares $\{v_0^i, v_1^i\}_{i \in [n]}$ that sum up to v , (i.e., $v_0^i \oplus v_1^i = v$ for all $i \in [n]$) and commits to them in parallel using the non-interactive statistically binding commitment scheme **com**. Let c_b^i be the commitment to v_b^i .

Challenge: The receiver sends a n -bit string $ch \in \{0, 1\}^n$ sampled at random.

Reply: The committer opens commitments $c_{ch_i}^i$ for every $i \in [n]$.

To decommit, the sender sends v and opens the commitments to all n pairs of strings. The receiver checks whether all the openings are valid and also $v = v_0^i \oplus v_1^i$ for all i .

It is proved in [41] that **ExtCom** is extractable. Furthermore, the commitment scheme has the property that from any two accepting transcripts of the commit stage that has the same commit message but different challenge messages, the committed value can be extracted. This property is similar to the notion of special-soundness for interactive proof/argument systems; here we overload this notion, and refer to this special extractability property of **ExtCom** as special-soundness.

In our construction, we will actually need an extractable commitment scheme to a string $\sigma \in \{0, 1\}^m$ for which we can open any subset of the bits in σ without compromising the security (i.e. hiding) of the remaining bits. As shown in [41], we may obtain such a scheme **PExtCom** by running **ExtCom** to commit to each bit of σ in parallel. It is easy to see that **PExtCom** is also special-sound in the sense that, given two accepting transcripts of **PExtCom** that have the same commit message and two challenge messages that contain a pair of different challenges for every **ExtCom** commitment, the committed string σ can be extracted. We call such two transcripts a pair of admissible transcripts for **PExtCom**.

Trapdoor Commitments. Roughly speaking, a trapdoor commitment scheme is a computationally biding and computationally hiding commitment scheme, such that, there exists a simulator that can generate a simulated commitment, and later open it to any value. (See [41] for a formal definition.) Pass and Wee [41] presented a black-box trapdoor bit commitment scheme **TrapCom**. To commit to a bit σ , the committer and the receiver on common input 1^n do:

Stage 1: The receiver picks a random string challenge $e = (e_1, \dots, e_n)$ and commits to e using the non-interactive statistically binding commitment scheme **com**.

Stage 2: The committer prepares v_1, \dots, v_n . Each v_i is a 2×2 0,1-matrix given by $v_i = \begin{bmatrix} v_i^{0,b} & v_i^{1,b} \\ v_i^{0,\beta} & v_i^{1,\beta} \end{bmatrix}_{2 \times 2}$ where $v_i^{0,b} = \eta_i$, $v_i^{1,b} = \sigma \oplus \eta_i$, with η_i is a random bit. The sender commits to v_1, \dots, v_n using **PExtCom**. In addition, the sender prepares $(a_1^0, a_1^1), \dots, (a_n^0, a_n^1)$ where a_i^β is the opening to $v_i^{\beta 0}, v_i^{\beta 1}$ (i.e., either the top or bottom row of v_i).

Stage 3: The receiver opens to the challenge $e = (e_1, \dots, e_n)$; the sender responds with $a_1^{e_1}, \dots, a_n^{e_n}$.

To decommit, the sender sends σ . In addition, it chooses a random $\gamma \in \{0, 1\}$, and sends the openings to values $v_i^{0\gamma}, v_i^{1\gamma}$ for $i = 1, 2, \dots, n$ (i.e., either the left columns or the right columns of all the matrices). The receiver checks that all the openings are valid, and also that $\sigma = v_1^{0\gamma} \oplus v_1^{1\gamma} = \dots = v_n^{0\gamma} \oplus v_n^{1\gamma}$.

As shown in [41], the protocol **TrapCom** is trapdoor, following a Goldreich-Kahan [18] style proof; moreover, by running **TrapCom** in parallel, we obtain a trapdoor commitment scheme **PTrapCom** for multiple bits. Furthermore, since Stage 2 of the protocol **TrapCom** is simply an execution of **PExtCom**, given any two *admissible transcripts* of Stage 2, the matrices v_1, \dots, v_n prepared in Stage 2 can be extracted; it is easy to see that from these matrices, the actual bit committed in the **TrapCom** commitment can be extracted, provided that the commitment is valid and has a unique committed value. We call this, again, the special-soundness of **TrapCom**. Again, the notion of special soundness (and admissible transcripts) can be easily extended for **PTrapCom**.

4.2 Overview of Our Construction

Towards a black-box construction of robust CCA secure commitment scheme, we start with the non-black-box construction of [9] (CLP), and tries to replace the non-black-box components in the CLP construction with “equivalent” black-box ones.

The CLP Construction: At a very high level, the CLP construction proceeds by having the committer first commit to the value v using a normal statistically binding commitment **com**, followed by a sequence of $\text{poly}(n)$ **WISSP** proofs of the committed value. The **WISSP** proofs are the non-black-box component of the CLP construction, but are crucial for achieving CCA-security. Recall that proving CCA-security w.r.t. \mathcal{O} amounts to showing that the views of A in experiments **IND**₀ and **IND**₁ are indistinguishable (when A has oracle access to \mathcal{O}). Let us refer to the adversary’s interaction with C as the *left interaction*, and its interactions with \mathcal{O} as the *right interactions*. The main hurdle in showing the indistinguishability of **IND**₀ and **IND**₁ is that the oracle \mathcal{O} is not efficiently computable; if it were, indistinguishability would directly follow from the hiding property of the left interaction. The main idea of the security proof of [9] is then to implement the oracle \mathcal{O} by extracting the committed values from the adversary, via “rewinding” the special-sound proofs in the right interactions. The following tow main technical challenges arise in simulating oracle \mathcal{O} .

First, once the simulation starts rewinding the right interactions, A might send new messages also in the left interaction. So, if done naively, this would rewind the left interaction, which could violate its hiding property. To solve this problem, the CLP protocol schedules messages in the special-sound proofs using a special message scheduling (according to the identity of the commitment), called the CLP scheduling, which is a variant of the message scheduling technique of [15,32]. The special message scheduling ensures that for every accepting right

interaction with an identity that is different from the left interaction, there exists many points—called **safe-points**—in the interaction, from which one can rewind the right interaction without requesting any *new* message in the left interaction.

Second, in the experiment IND_b , the adversary A expects to receive the committed value at the very moment it completes a commitment to its oracle. If the adversary “nests” its oracle calls, these rewinds become recursive and the running-time of the extraction quickly becomes exponential. To avoid the extraction time from exploding, the simulation strategy in CLP rewinds from **safe-points** using a concurrent extraction strategy that is similar to that used in the context of concurrent \mathcal{ZK} by Richardson and Killian [44].

New Approach: To obtain a black-box construction, our main goal is to replace the $WISSP$ proofs with an “equivalent” black-box component. The key property that the CLP proof relies on is that the protocol contains many *3-round* constructs satisfying that rewinding the last two messages reveals the committed value, but rewinding three messages reveals nothing. It seems that the 3-round commitment scheme PExtCom is a good replacement of $WISSP$ proofs as one such 3-round construct: The special-soundness property of PExtCom ensures that rewinding the last two messages reveals the committed value, and the hiding property ensures that rewinding three messages reveals nothings. It is thus tempting to consider a commitment scheme in which the committer commits to value v using $\text{poly}(n)$ invocations of PExtCom , arranged according to the CLP scheduling; the CLP extraction strategy guarantees that for every accepting right interaction, (the last two messages of) one PExtCom commitment is rewound and a committed value is extracted. Indeed, if a commitment of this scheme is valid, meaning that all the PExtCom commitments contained in it are valid commitments to the same value, the CLP extraction strategy returns the unique committed value. However, if the commitment is invalid, there arises the over-extraction problem: The CLP extraction strategy may extract a garbage value from an invalid PExtCom commitment or from a valid commitment that is inconsistent with the other commitments.

To solve the over-extraction problem, we use the cut-and-choose technique to enforce the committer to give valid and consistent PExtCom commitments. Instead of having the committer commit to v directly, let it commit to a $(n+1)$ -*out-of-10n Shamir’s secret sharing* s_1, \dots, s_{10n} of v using many PExtCom invocations, still arranged according to the CLP scheduling; we refer to all the commitments to the j^{th} share s_j the j^{th} column. After all the PExtCom commitments, the receiver requests the committer to open all the commitments in n randomly chosen columns; the receiver accepts only if each column contains valid commitments to the same value. It follows from the cut-and-choose technique that except with negligible probability, at most n columns may contain invalid or inconsistent commitments. Therefore, when applying the CLP extraction strategy on a commitment of this scheme, it guarantees to extract out a secret-sharing that is .9-close to all the secret-sharing committed to in this commitment. Then by relying on the error-correcting property of the secret sharing,

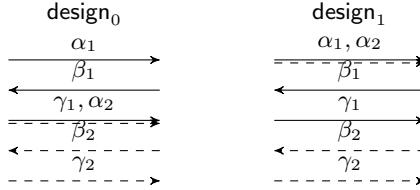


Fig. 1. Description of the schedules used in Stage 2 of the protocol. $(\alpha_1, \beta_1, \gamma_1)$ and $(\alpha_2, \beta_2, \gamma_2)$ are respectively the transcripts of a pair of rows in Stage 2 of $\langle C, R \rangle$.

The robust CCA-secure protocol $\langle C, R \rangle$

Let κ be an arbitrary polynomial, ℓ, η two polynomials such that $\ell(n) = n^\nu$ and $\eta(n) = n^\varepsilon$ for $\nu, \varepsilon > 0$, and L a polynomial such that $L(n) = \max(\kappa(n) + \eta(n), 4\ell(n)\eta(n))$. To commit to a value v , the committer C and the receiver R , on common input 1^n and the identity $\text{id} \in \{0, 1\}^{\ell(n)}$ of the committer C do:

Stage 1: The receiver sends the Stage 1 message of a commitment of PTrapCom. That is, a commitment of `com` to a randomly chosen string challenge $e = (e_1, \dots, e_n)$.

Stage 2: The committer C prepares a $(n+1)$ -out-of- $10n$ Shamir's secret sharing s_1, \dots, s_{10n} of the value v , and commits to these shares using Stage 2 of the protocol PTrapCom in parallel, *for $L(n)$ times*; we call the i^{th} parallel commitment the i^{th} *row*, and all the commitments to the i^{th} share s_i the i^{th} *column*.

Messages in the first $4\ell(n)\eta(n)$ rows are scheduled based on the identity id and relies on scheduling pairs of rows according to schedules design_0 and design_1 depicted in Figure 1. More precisely, Stage 2 consist of $\ell(n)$ phases. In phase i , the committer provides $\eta(n)$ sequential $\text{design}_{\text{id}_i}$ pairs of rows, followed by $\eta(n)$ sequential $\text{design}_{1-\text{id}_i}$ pairs of rows. Messages in the rest of the rows are simply arranged sequentially.

Stage 3: The receiver opens the Stage 1 commitment to the challenge e . The committer completes the $10nL(n)$ executions of PTrapCom w.r.t. challenge e in parallel.

Stage 4 (cut-and-choose): The receiver sends a randomly chosen subset $\Gamma \subseteq [10n]$ of size n . For every $j \in \Gamma$, the committer opens all the commitments in the j^{th} column of Stage 3. The receiver checks that all the openings are valid, and reveal the same committed values s_j .

Decommitment Message: To decommit, the committer sends v , and opens all the commitments in the first row of Stage 2 to s_1, \dots, s_{10n} . The receiver checks all the openings to s_1, \dots, s_{10n} are valid; furthermore, it checks that s_1, \dots, s_{10n} is 0.9-close to a valid codeword $w = (w_1, \dots, w_{10n})$, and for every $j \in \Gamma$, w_j equals to the share s_j revealed in Stage 4.

In other words, a commitment of $\langle C, R \rangle$ is valid if and only if the first row in Stage 2 of the commitment contains valid commitments to shares s_1, \dots, s_{10n} , such that, s_1, \dots, s_{10n} is 0.9 close to a valid codeword w , and w agrees with all the shares revealed in Stage 4 (i.e., for every $j \in \Gamma$, $w_j = s_j$).

Fig. 2. The formal description of the $\kappa(n)$ -robust CCA-secure protocol $\langle C, R \rangle$

a valid committed value can be reconstructed. The formal analysis is actually more subtle; to avoid over-extraction, we employ the technique used in [12,13,46], which involves setting the validity condition of the commitment scheme carefully so that invalid commitment can be identified.

Unfortunately, our use of the cut-and-choose technique brings another problem: The above commitment scheme may not be hiding. This is because, in the last stage, the receiver may request the committer to open an adaptively chosen subset of commitments of PExtCom , and thus the remaining unopened commitments may not be hiding, unless PExtCom were secure against selective opening attack. To resolve this problem, we use the trapdoor commitment scheme PTrapCom to replace PExtCom . Since PTrapCom is trapdoor, it is secure against selective opening attack, and thus the hiding property holds. Furthermore, since Stage 2 of PTrapCom is simply a commitment of PExtCom , we can use Stage 2 of PTrapCom as an implementation of the 3-round construct needed for the CLP scheduling and extraction strategy. More precisely, the commitment scheme proceeds as follow: The committer commits to a $(n+1)$ -out-of- $10n$ secret sharing of the value v using many invocations of PTrapCom , where all the invocations share the same Stage 1 message sent at the beginning, followed by all the 3-round Stage 2 executions arranged according to the CLP scheduling, and then all the Stage 3 executions performed in parallel; finally, the committer and the receiver conducts a cut-and-choose consistency check as described above. A formal description of our CCA secure protocol $\langle C, R \rangle$ in Figure 2.

It seems that the security proof of our CCA-secure commitment should follow from that of the non-black-box construction of [9]. Unfortunately, due to the fact that the “rewinding slots” of our protocol, that is the commitment of ExtCom , may have over-extraction, whereas the \mathcal{WISSP} proofs in the CLP protocol never has this problem, the technical proof of [9] does not go through. In the full version, we rely on a different analysis to show the security of our protocol.

References

1. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)
2. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552 (2005)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
4. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology, 143–202 (2000)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
6. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Security with Global Setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
7. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

8. Canetti, R., Kushilevitz, E., Lindell, Y.: On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In: Biham, E. (ed.) *EUROCRYPT 2003*. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)
9. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: *FOCS*, pp. 541–550 (2010)
10. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: *STOC*, pp. 494–503 (2002)
11. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: *FOCS*, pp. 249–259 (2007)
12. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-Box Construction of a Non-malleable Encryption Scheme from Any Semantically Secure One. In: Canetti, R. (ed.) *TCC 2008*. LNCS, vol. 4948, pp. 427–444. Springer, Heidelberg (2008)
13. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols. In: Reingold, O. (ed.) *TCC 2009*. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
14. Damgård, I., Ishai, Y.: Constant-Round Multiparty Computation Using a Black-Box Pseudorandom Generator. In: Shoup, V. (ed.) *CRYPTO 2005*. LNCS, vol. 3621, pp. 378–394. Springer, Heidelberg (2005)
15. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. *SIAM Journal on Computing* 30(2), 391–437 (2000)
16. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: *STOC*, pp. 416–426 (1990)
17. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently Secure Computation in Constant Rounds. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (2012)
18. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology* 9(3), 167–190 (1996)
19. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: *STOC*, pp. 218–229 (1987)
20. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM* 38(3), 690–728 (1991)
21. Goldwasser, S., Micali, S.: Probabilistic encryption. *J. Comput. Syst. Sci.* 28(2), 270–299 (1984)
22. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM Journal on Computing* 18(1), 186–208 (1989)
23. Goyal, V.: Constant round non-malleable protocols using one way functions. In: *STOC*, pp. 695–704 (2011)
24. Groth, J., Ostrovsky, R.: Cryptography in the Multi-string Model. In: Menezes, A. (ed.) *CRYPTO 2007*. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
25. Haitner, I.: Semi-honest to Malicious Oblivious Transfer—The Black-Box Way. In: Canetti, R. (ed.) *TCC 2008*. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (2008)
26. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: *STOC*, pp. 99–108 (2006)
27. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
28. Kalai, Y.T., Lindell, Y., Prabhakaran, M.: Concurrent general composition of secure protocols in the timing model. In: *STOC*, pp. 644–653 (2005)

29. Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) *EUROCRYPT 2007*. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
30. Kilian, J.: Founding cryptography on oblivious transfer. In: *STOC*, pp. 20–31 (1988)
31. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: *STOC*, pp. 723–732 (1992)
32. Lin, H., Pass, R., Venkatasubramaniam, M.: Concurrent Non-malleable Commitments from Any One-Way Function. In: Canetti, R. (ed.) *TCC 2008*. LNCS, vol. 4948, pp. 571–588. Springer, Heidelberg (2008)
33. Lin, H., Pass, R., Venkatasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: *STOC*, pp. 179–188 (2009)
34. Lin, H., Pass, R., Venkatasubramaniam, M.: UC from semi-honest OT (2012) (manuscript)
35. Lindell, Y.: Lower Bounds for Concurrent Self Composition. In: Naor, M. (ed.) *TCC 2004*. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
36. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) *EUROCRYPT 2007*. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
37. Malkin, T., Moriarty, R., Yakovenko, N.: Generalized Environmental Security from Number Theoretic Assumptions. In: Halevi, S., Rabin, T. (eds.) *TCC 2006*. LNCS, vol. 3876, pp. 343–359. Springer, Heidelberg (2006)
38. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: *FOCS*, pp. 367–378 (2006)
39. Pass, R.: Simulation in Quasi-Polynomial Time, and its Application to Protocol Composition. In: Biham, E. (ed.) *EUROCRYPT 2003*. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
40. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: *FOCS*, pp. 563–572 (2005)
41. Pass, R., Wee, H.: Black-Box Constructions of Two-Party Protocols from One-Way Functions. In: Reingold, O. (ed.) *TCC 2009*. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009)
42. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: *FOCS*, pp. 366–375 (2002)
43. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: *STOC*, pp. 242–251 (2004)
44. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Stern, J. (ed.) *EUROCRYPT 1999*. LNCS, vol. 1592, pp. 415–432. Springer, Heidelberg (1999)
45. Rosen, A.: A Note on Constant-Round Zero-Knowledge Proofs for NP. In: Naor, M. (ed.) *TCC 2004*. LNCS, vol. 2951, pp. 191–202. Springer, Heidelberg (2004)
46. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: *FOCS*, pp. 531–540 (2010)
47. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: *FOCS*, pp. 162–167 (1986)

Crowd-Blending Privacy

Johannes Gehrke*, Michael Hay**, Edward Lui, and Rafael Pass***

Department of Computer Science, Cornell University
`{johannes,mhay,luied,rafael}@cs.cornell.edu`

Abstract. We introduce a new definition of privacy called *crowd-blending privacy* that strictly relaxes the notion of differential privacy. Roughly speaking, k -crowd blending private sanitization of a database requires that each individual i in the database “blends” with k other individuals j in the database, in the sense that the output of the sanitizer is “indistinguishable” if i ’s data is replaced by j ’s.

We demonstrate crowd-blending private mechanisms for histograms and for releasing synthetic data points, achieving strictly better utility than what is possible using differentially private mechanisms. Additionally, we demonstrate that if a crowd-blending private mechanism is combined with a “pre-sampling” step, where the individuals in the database are randomly drawn from some underlying population (as is often the case during data collection), then the combined mechanism satisfies not only differential privacy, but also the stronger notion of zero-knowledge privacy. This holds even if the pre-sampling is slightly biased and an adversary knows whether certain individuals were sampled or not. Taken together, our results yield a practical approach for collecting and privately releasing data while ensuring higher utility than previous approaches.

1 Introduction

Data privacy is a fundamental problem in today’s information age. Large amounts of data are collected from people by government agencies, hospitals, social networking systems, and other organizations, and are stored in databases. There are huge social benefits in analyzing this data, and in many situations, these

* Gehrke’s work on this material was supported by the NSF under Grant IIS-1012593. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

** Hay’s work was supported by the Computing Innovation Fellows Project (<http://cifellows.org/>), funded by the Computing Research Association/Computing Community Consortium through NSF Grant 1019343.

*** Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF CAREER Award CCF-0746990, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US government.

organizations would like to release the data in some form for people to analyze. However, it is important to protect the privacy of the people that contributed their data; organizations need to make sure that sensitive information about individuals is not leaked to the people analyzing the data.

Many privacy definitions and schemes for releasing data have been proposed in the past (see [1] and [2] for surveys). However, many of them have been shown to be insufficient due to realistic attacks on such schemes (e.g., see [3]). The notion of *differential privacy* [4,5], however, has remained strong and resilient to these attacks. Differential privacy requires that when one person's data is added or removed from the database, the output distribution of the database access mechanism changes very little (by at most an ϵ amount, where a specific notion of closeness of distributions is used). Differential privacy has quickly become the standard definition of privacy, and mechanisms for releasing a variety of functions (including histogram queries, principal component analysis, learning, and many more; see [6,7] for a survey) have been developed.

One way to interpret the notion of differential privacy is that an attacker does not learn more about an individual i than what can be deduced from the data of everyone else in the database (see the appendix of [4]). In the context of e.g., social networks, where the data of an individual may be strongly correlated with the data of his/her friends, such a notion may not always provide sufficient privacy guarantees. To address this issue, an even stronger privacy definition, *zero-knowledge privacy*, was introduced in [8]. Roughly speaking, zero-knowledge privacy requires that whatever an adversary learns about an individual i can be “simulated” given just some “aggregate” information about the *remaining* individuals in the database; for instance, this aggregate information could be k random samples of the remaining individuals in the database. If the aggregate information contains all individuals (excluding i), zero-knowledge privacy collapses down to differential privacy, but for more restrictive classes of aggregate information (such as k random samples, where k is smaller than the number of individual in the database) zero-knowledge privacy is strictly stronger, and provides stronger privacy guarantees in contexts where there is correlation between individuals.

Privacy from Random Sampling of Data. Both differential privacy and zero-knowledge privacy provide strong privacy guarantees. However, for certain tasks, mechanisms satisfying these privacy definitions have to add a lot of “noise”, thus lowering the utility of the released data. Also, many of these mechanisms run in exponential time (e.g., [9,10]), so efficiency is also an issue. This leaves open the question of whether there exists a practical approach to sanitizing data, without harming utility too much.

One approach for circumventing the above-mentioned issues is to rely on the fact that in many cases of interest, the data to be sanitized has been collected via *random sampling* from some underlying population. Intuitively, this initial random sampling already provides some basic privacy guarantees, and may thus help us in decreasing the amount of noise added during sanitization. Indeed, there are several results in the literature indicating that random sampling helps

in providing privacy: In [11] the authors quantify the level of the privacy that may be obtained from just random sampling of data (without any further sanitization); in [12] the authors consider a certain type of “sample-and-aggregate” mechanism for achieving differential privacy (but the sampling technique here is more elaborate than just random sampling from a population); a result in [13] shows that random pre-sampling can be used to amplify the privacy level of a differentially private mechanism; finally, in a manuscript [14], the authors demonstrate that a random pre-sampling step applied to a particular mechanism leads to a differentially private mechanism.

In this paper, we continue the investigation of using random sampling as a means to achieve privacy. In particular, our goal is to provide a *general* definition of privacy that allows us to achieve both differential and zero-knowledge privacy in situations where the data is collected using random sampling from some population. In order to be realistic, we allow the random sampling during data collection to be *biased*, and an adversary may even know whether certain individuals were sampled or not. (Although the mechanisms in the earlier papers rely on random sampling, the random sampling is usually thought of as being part of the sanitization procedure and thus the mechanisms are only analyzed under the assumption that the sampling has been done “ideally”.) Additionally, we will require that the privacy notion is meaningful in its own right, also without any pre-sampling; we believe this requirement is crucial for guaranteeing a strong fall-back guarantee even in case the result of the pre-sampling is leaked (and thus the attacker knows exactly who was sampled).

1.1 Towards a Weaker Notion of Privacy

We aim to develop a new privacy definition that allows us to design mechanisms that have greater utility or efficiency than differentially private mechanisms, but still provide a meaningful notion of privacy; furthermore, we want mechanisms satisfying the new definition to achieve differential and zero-knowledge privacy when the underlying data was collected via biased random sampling from some population. To this end, we begin by reconsidering some older notions of privacy.

k-Anonymity and Blending in a Crowd. *k-anonymity* [15] is a privacy definition specifically for releasing data tables, where a data table is simply a table of records (rows), each of which has values for the attributes (columns) of the table. Roughly speaking, a released data table satisfies *k-anonymity* if every record in the table is the same as $k - 1$ other records in the table with respect to certain “identifying” attributes (chosen beforehand). *k-anonymity* imposes constraints on the syntax of the released data table, but does not consider the way the released data table was computed from the underlying database; this issue has led to several practical attacks against the notion of *k-anonymity* (e.g., see [16,17]). *k-anonymity* can be viewed as being based on the intuition of “*blending in a crowd*”, since the records in the released output are required to “blend” with other records. Intuitively, in many cases, if an individual blends in a crowd of many people in the database, then the individual’s privacy is sufficiently

protected. However, as demonstrated by known attacks, k -anonymity does not properly capture this intuition as it does not impose any restrictions on the algorithm/mechanism used to generate the released output. Indeed, one of the key insights behind the notion of differential privacy was that privacy should be a property of the sanitization mechanism and not just the output of it.

Relying on this insight, we aim to develop a privacy notion that captures what it means for a mechanism to guarantee that individuals “blend in a crowd”. (Another definition partly based on the intuition of blending in a crowd is (c, t) -isolation [18], which requires adversaries to be unable to isolate an individual, represented by a data point in \mathbb{R}^d , by roughly determining the individual’s location in \mathbb{R}^d ; we formalize the intuition of blending in a crowd in a very different way.)

Crowd-Blending Privacy – A New Privacy Definition. Let us now turn to describing our new privacy definition, which we call *crowd-blending privacy*. We say that an individual *blends* with another individual with respect to a mechanism *San* if the two individuals are *indistinguishable by the mechanism San*, i.e., whenever we have a database containing either one or both of the individuals, we can replace one of the individual’s data with the other individual’s data, and the mechanism’s output distribution remains essentially the same. We say that an individual t *blends in a crowd of k people in the database D with respect to the mechanism San* if there exist at least $k - 1$ other individuals in the database D that blend with individual t with respect to *San*. The intuition behind this notion is that if an individual t blends in a crowd of k people in the database, then the mechanism essentially does not release any information about individual t beyond the general characteristics of the crowd of k people; in particular, the mechanism does not release any personal information that is specific to individual t and no one else.

Roughly speaking, we say that a mechanism *San* is *crowd-blending private* if the following property holds: For every database and every individual in the database, either the individual *blends in a crowd of k people in the database with respect to San*, or the mechanism *San* *essentially ignores the individual’s data*.

We do not claim that crowd-blending privacy provides sufficiently strong privacy protection in *all* scenarios: the key weakening with respect to differential privacy is that an attacker who knows the data of everyone in an individual i ’s crowd (except i) may learn information about individual i , as long as this information is “general” in the sense that it applies to the entire crowd. For instance, if the attacker knows everyone in the crowd of individual i , it may deduce that i has, say, three children, as long as everyone in i ’s crowd has three children. Although to some extent, this may be viewed as a privacy violation (that would not be allowed by the notion of differential privacy), we would argue that the attribute leaked about individual i is “non-sensitive” as it is shared by a sufficiently large crowd. Thus, we view this weakening as desirable in many contexts as it allows us to trade privacy of “non-sensitive information” for improved utility.

A potentially more serious deficiency of the definition is that (in contrast to differential and zero-knowledge privacy) crowd-blending privacy is not closed

under composition: $\mathcal{S}an_1$ and $\mathcal{S}an_2$ may both be crowd-blending private, but the crowds for an individual with respect to $\mathcal{S}an_1$ and $\mathcal{S}an_2$ could be essentially disjoint, making the individual's crowd for the combination of $\mathcal{S}an_1$ and $\mathcal{S}an_2$ very small. Although we view composition as an important property of a privacy definition, our goal here is to study the weakest possible “meaningful” definition of “stand-alone” privacy that when combined with pre-sampling leads to strong privacy notions (such as differential and zero-knowledge privacy) that themselves are closed under composition.

1.2 New Database Mechanisms

As it turns out, achieving crowd-blending privacy is significantly easier than achieving differential privacy, and crowd-blending private mechanisms may yield significantly higher utility than differentially private ones.

Privately Releasing Histograms with No Noise for Sufficiently Large Counts. We show that we can release histograms with crowd-blending privacy where no noise is added to bins with a sufficiently large count (and only a small amount of noise is added to bins with a small count). Intuitively, individuals in the same bin blend with each other; thus, the individuals that belong to a bin with a sufficiently large count already blend in a crowd, so no noise needs to be added to the bin. It is easy to see that it is impossible to release the exact count of a bin in a histogram while satisfying differential privacy or zero-knowledge privacy. Using crowd-blending privacy, we can overcome this limitation (for bins with a sufficiently large count) and achieve better utility. These results can be found in Section 3.1.

Privately Releasing Synthetic Data Points in \mathbb{R}^d for Computing Smooth Functions. Given a class \mathcal{C} of counting queries whose size is not too large, it is shown in [10] how to release a synthetic database for approximating all the queries in \mathcal{C} simultaneously while satisfying differential privacy; however, the mechanism is not necessarily efficient. It is known that it is impossible (assuming the existence of one-way functions) to *efficiently* and privately release a synthetic database for approximating certain classes of counting queries, such as the class of all 2-way marginals (see [19,20]). However, these query functions are non-smooth in the sense that even slightly changing one row of the input database can affect the output of the query functions quite a lot. Here, we focus on efficiently and privately releasing synthetic data for approximating *all* “smooth” functions $g : (\mathbb{R}^d)^* \rightarrow \mathbb{R}^m$.

Roughly speaking, a function $g : (\mathbb{R}^d)^* \rightarrow \mathbb{R}^m$ is *smooth* if the value of g does not change much when we perturb the data points of the input slightly. We show that we can *efficiently* release synthetic data points in \mathbb{R}^d for approximating *all* smooth functions simultaneously while satisfying crowd-blending privacy. On the other hand, we show that there are smooth functions that cannot even be approximated with non-trivial utility from any synthetic data that has been released with differential privacy (even if the differentially private mechanism is *inefficient*). These results can be found in the full version of this paper.

1.3 From Crowd-Blending Privacy to Zero-Knowledge Privacy

Our main technical result shows that if we combine a crowd-blending private mechanism with a natural pre-sampling step, then the combined algorithm satisfies zero-knowledge privacy (and thus differential privacy as well). We envision the pre-sampling step as being part of the data collection process, where individuals in some population are sampled and asked for their data. Thus, if data is collected using random sampling of individuals from some population, and next sanitized using a crowd-blending private mechanism, then the resulting process ensures zero-knowledge privacy.

We first prove our main theorem for the case where the pre-sampling step samples each individual in the population with probability p independently. In reality, the sampling performed during data collection may be slightly biased or done slightly incorrectly, and an adversary may know whether certain individuals were sampled or not. Thus, we next extend our main theorem to also handle the case where the sampling probability is not necessarily the same for everybody, but the sampling is still “robust” in the sense that most individuals are sampled independently with probability in between p and p' (this probability can even depend on the individual’s data), where p and p' are relatively close to one another, while the remaining individuals are sampled independently with arbitrary probability. As a result, we have that in scenarios where data has been collected using any robust sampling, we may release data which both ensures strong utility guarantees and satisfies very strong notions of privacy (i.e., zero-knowledge privacy and differential privacy). In particular, this methodology can allow us to achieve zero-knowledge privacy and differential privacy while guaranteeing utility that is better than that of previous methods (such as for releasing histograms or synthetic data points as described above). Our main theorems can be found in Section 4.

It is worthwhile to note that the particular mechanism considered in [14] (which in fact is a particular mechanism for achieving k -anonymity) can easily be shown to satisfy crowd-blending privacy; as a result, their main result can be derived (and significantly strengthened) as a corollary of our main theorem.¹ (See Section 3.1 and 4 for more details.)

2 Preliminaries and Existing Privacy Definitions

A *database* is a finite *multiset* of data values, where a data value is simply an element of some fixed set X , which we refer to as the *data universe*. Each data value in a database belongs to an individual, so we also refer to a data value in a database as an *individual* in the database. For convenience, we will sometimes

¹ As mentioned, none of the earlier work using random pre-sampling focus on the case when the sampling is biased; furthermore, even for the case of perfect random sampling, the authors of [14] were not able to provide a closed form expression of the level of differential privacy achieved by their mechanism, whereas a closed form expression can be directly obtained by applying our main theorem.

order the individuals in a database in an arbitrary way and think of the database as an element of X^* , i.e., a vector with components in X (the components are referred to as the *rows* of the database). Given a database D and a data value $v \in X$, let (D, v) denote the database $D \cup \{v\}$. A (database) *mechanism* is simply an algorithm that operates on databases.

Given $\epsilon, \delta \geq 0$ and two distributions Z and Z' over $\{0, 1\}^*$, we shall write $Z \approx_{\epsilon, \delta} Z'$ to mean that for every $Y \subseteq \{0, 1\}^*$ we have

$$\Pr[Z \in Y] \leq e^\epsilon \Pr[Z' \in Y] + \delta$$

and

$$\Pr[Z' \in Y] \leq e^\epsilon \Pr[Z \in Y] + \delta.$$

We shall also write $Z \approx_\epsilon Z'$ to mean $Z \approx_{\epsilon, 0} Z'$. Differential privacy (see [4,5]) can now be defined in the following manner:

Definition 1 ([4,5]). *A mechanism San is said to be ϵ -differentially private if for every pair of databases D and D' differing in only one data value, we have $\text{San}(D) \approx_\epsilon \text{San}(D')$.*

There are two definitions in the literature for “a pair of databases D and D' differing in only one data value”, leading to two slightly different definitions of differential privacy. In one definition, it is required that D contains D' and has exactly one more data value than D' . In the other definition, it is required that $|D| = |D'|$, $|D \setminus D'| = 1$, and $|D' \setminus D| = 1$. Intuitively, differential privacy protects the privacy of an individual t by requiring the output distribution of the mechanism to be essentially the same regardless of whether individual t 's data is included in the database or not (or regardless of what data value individual t has).

We now begin describing zero-knowledge privacy, which is a privacy definition introduced in [8] that is strictly stronger than differential privacy. In the definition of zero-knowledge privacy, *adversaries* and *simulators* are simply randomized algorithms that play certain roles in the definition. Let San be any mechanism. For any database D , any adversary A , and any auxiliary information $z \in \{0, 1\}^*$, let $\text{Out}_A(A(z) \leftrightarrow \text{San}(D))$ denote the output of A on input z after interacting with the mechanism San operating on the database D . San can be interactive or non-interactive. If San is non-interactive, then $\text{San}(D)$ simply sends its output (e.g., sanitized data) to A and then halts immediately.

Let agg be any class of randomized algorithms. agg is normally a class of randomized aggregation functions that provide aggregate information to simulators, as described in the introduction.

Definition 2 ([8]). *A mechanism San is said to be (ϵ, δ) -zero-knowledge private with respect to agg if there exists a $T \in \text{agg}$ such that for every adversary A , there exists a simulator S such that for every database D , every individual $t \in D$, and every auxiliary information $z \in \{0, 1\}^*$, we have*

$$\text{Out}_A(A(z) \leftrightarrow \text{San}(D)) \approx_{\epsilon, \delta} S(z, T(D \setminus \{t\}), |D|).$$

Intuitively, zero-knowledge privacy requires that whatever an adversary can compute about individual t by accessing (i.e., interacting with) the mechanism can also be essentially computed without accessing the mechanism but with certain aggregate information about the remaining individuals; this aggregate information is provided by an algorithm in agg . The adversary in the latter scenario is represented by the simulator S . This ensures that the adversary essentially does not learn any additional information about individual t beyond the aggregate information provided by an algorithm in agg on the remaining individuals.

agg is normally some class of randomized aggregation functions, such as the class of all functions T that draws r random samples from the input database and performs any computation (e.g., computes the average or simply outputs the samples) on the r random samples (note that in the definition, T is applied to $D \setminus \{t\}$ instead of D so that the aggregate information from T does not depend directly on individual t 's data). Zero-knowledge privacy with respect to this class of aggregation functions ensures that an adversary essentially does not learn anything more about an individual beyond some “ r random sample aggregate information” of the other individuals. One can also consider zero-knowledge privacy with respect to other classes of aggregation functions, such as the class of (randomized) functions that first sample each row of the input database with probability p (or in between p and p') independently and then performs any computation on the samples. We will actually use such classes of aggregation functions when we prove our main theorems later. It can be easily shown that zero-knowledge privacy (with respect to any class agg) implies differential privacy (see [8]).

3 Crowd-Blending Privacy – A New Privacy Definition

We now begin to formally define our new privacy definition. Given $t, t' \in X$, $\epsilon \geq 0$, and a mechanism San , we say that t and t' are ϵ -indistinguishable by San , denoted $t \approx_{\epsilon, San} t'$, if $San(D, t) \approx_{\epsilon} San(D, t')$ for every database D . Intuitively, t and t' are indistinguishable by San if for any database containing t , we can replace the t by t' and the output distribution of San remains essentially the same. Usually, t and t' are the data values of two individuals, and if t and t' are indistinguishable by San , then this roughly means that San cannot distinguish these two individuals regardless of who else is in the database. If t and t' are ϵ -indistinguishable by San , we also loosely say that t blends with t' (with respect to San). We now describe what it means for an individual to blend in a crowd of people in the database (with respect to a mechanism).

Definition 3. Let D be any database. An individual $t \in D$ **ϵ -blends in a crowd of k people in D with respect to the mechanism San** if $|\{t' \in D : t' \approx_{\epsilon, San} t\}| \geq k$.

In the above definition, $\{t' \in D : t' \approx_{\epsilon, San} t\}$ should be regarded as a multiset. When the mechanism San is clear from context, we shall simply omit the “with respect to the mechanism San ”. Intuitively, an individual $t \in D$ blends in a

crowd of k people in D if t is indistinguishable by San from at least $k - 1$ other individuals in D . Note that by the definition of two individuals being indistinguishable by San , $t \in D$ must be indistinguishable by San from each of these $k - 1$ other individuals *regardless of what the database is*, as opposed to only when the database is D . (A weaker requirement would be that for each of these $k - 1$ other individuals t' , t and t' only need to be “indistinguishable by San with respect to D ”, i.e., if we take D and replace t by t' or vice versa, the output distributions of San on D and the modified D are essentially the same; we leave investigating this and other possible weaker requirements for future work.) We are now ready to state our new privacy definition.

Definition 4 (Crowd-blending privacy). *A mechanism San is (k, ϵ) -crowd-blending private if for every database D and every individual $t \in D$, either t ϵ -blends in a crowd of k people in D , or $\text{San}(D) \approx_\epsilon \text{San}(D \setminus \{t\})$ (or both).*

Crowd-blending privacy requires that for every individual t in the database, either t blends in a crowd of k people in the database, or the mechanism essentially ignores individual t 's data (the latter case is captured by $\text{San}(D) \approx_\epsilon \text{San}(D \setminus \{t\})$ in the definition). When an individual t blends in a crowd of k people in the database, the mechanism essentially does not release any information about individual t beyond the general characteristics of the crowd of k people. This is because the mechanism cannot distinguish individual t from the people in the crowd of k people, i.e., individual t 's data can be changed to the data of another person in the crowd of k people and the output distribution of the mechanism remains essentially the same. A consequence is that the mechanism does not release any personally identifying information about individual t .

As mentioned in the introduction, crowd-blending privacy is not closed under composition (see the full version of this paper for an example); however, we note that the privacy guarantee of blending in a crowd of k people in the database (described above) holds regardless of the amount of auxiliary information the adversary has (i.e., the definition is agnostic to the adversary's auxiliary information). Additionally, as mentioned previously, we show in Section 4 that when crowd-blending privacy is combined with “robust pre-sampling”, we get zero-knowledge privacy and thus differential privacy as well, both of which satisfy composition in a natural way. Thus, as long as robust sampling is used during data collection before running a crowd-blending private mechanism on the collected data, independent releases from crowd-blending private mechanisms do compose and satisfy zero-knowledge privacy and differential privacy. (We also mention that one can compose a crowd-blending private mechanism with a differentially private mechanism to obtain a crowd-blending private mechanism; see the full version of this paper for details.)

Relationship with Differential Privacy. Differential privacy implies crowd-blending privacy.

Proposition 1 (Differential privacy \implies Crowd-blending privacy). *Let San be any ϵ -differentially private mechanism. Then, San is (k, ϵ) -crowd-blending private for every integer $k \geq 1$.*

Proof. This immediately follows from the two privacy definitions.

(k, ϵ) -crowd-blending privacy for some integer k does not imply differential privacy in general; this will be clear from the examples of crowd-blending private mechanisms that we give later. Crowd-blending privacy requires that for every database D and every individual $t \in D$, at least one of two conditions hold. The second condition $\text{San}(D) \approx_\epsilon \text{San}(D \setminus \{t\})$ is similar to the condition required in differential privacy. Thus, we can view crowd-blending privacy as a relaxation of differential privacy. If we remove the first condition “ t ϵ -blends in a crowd of k people in D ” from crowd-blending privacy, we clearly get the same definition as differential privacy. If we remove the second condition instead, it turns out that we also get differential privacy. (When we remove the second condition $\text{San}(D) \approx_\epsilon \text{San}(D \setminus \{t\})$, we also change the definition to only consider databases of size at least k , since otherwise it would be impossible for individual t to blend in a crowd of k people in the database.)

Proposition 2 (Removing the condition $\text{San}(D) \approx_\epsilon \text{San}(D \setminus \{t\})$ in crowd-blending privacy results in differential privacy). *Let San be any mechanism, let $\epsilon \geq 0$, and let k be any integer ≥ 2 . Then, San is ϵ -differentially private² if and only if San satisfies the property that for every database D of size at least k and every individual $t \in D$, t ϵ -blends in a crowd of k people in D with respect to San .*

See the full version of this paper for the proof of Proposition 2.

3.1 Examples of Crowd-Blending Private Mechanisms

Given a partition P of the data universe X , and given a database D , one can compute the histogram with respect to the partition P using the database D ; the histogram specifies for each block of the partition (which we refer to as a “bin”) the number of individuals in D that belong to the block (which we refer to as the “count” of the bin). We first give an example of a crowd-blending private mechanism that computes a histogram and suppresses (i.e., sets to 0) bin counts that are considered too small.

Example 1 (Histogram with suppression of small counts). Let P be any partition of X . Fix $k \in \mathbb{Z}_{\geq 0}$. Let San be a mechanism that, on input a database D , computes the histogram with respect to the partition P using the database D , suppresses each bin count that is $< k$ (by setting the count to 0), and then releases the resulting histogram.

Then, San is $(k, 0)$ -crowd-blending private. To see this, we note that an individual t in a database D is 0-indistinguishable by San from all the individuals in D that belong to the same bin as t . If there are at least k such people, then individual t blends with k people in D ; otherwise, we have $\text{San}(D) \approx_0 \text{San}(D \setminus \{t\})$ since San suppresses each bin count that is $< k$.

² Here, we are using the version of differential privacy that considers a pair of databases of equal size.

It is easy to see that it is impossible to release the exact count of a bin while satisfying differential privacy. Thus, crowd-blending privacy is indeed weaker than differential privacy. For crowd-blending privacy, we can actually get better utility by adding a bit of noise to bins with low counts instead of completely suppressing them.

Example 2 (Histogram with noise for small counts and no noise for large counts). Let P be any partition of X . Fix $\epsilon > 0$ and $k \in \mathbb{Z}_{\geq 0}$. Let San be a mechanism that, on input a database D , computes the histogram with respect to the partition P using the database D . Then, San replaces each bin count $i < k$ with $A(i)$, where A is any (randomized) algorithm that satisfies $A(j) \approx_{\epsilon} A(j - 1)$ for every $0 < j < k$ ($A(i)$ is normally a noisy version of i). San then releases the noisy histogram.

Then, San is (k, ϵ) -crowd-blending private. To see this, we note that an individual t in a database D is ϵ -indistinguishable (in fact, 0-indistinguishable) by San from all the individuals in D that belong to the same bin as t . If there are at least k such people, then individual t blends with k people in D , as required. If not, then we have $\text{San}(D) \approx_{\epsilon} \text{San}(D \setminus \{t\})$, since the histogram when using the database D is the same as the histogram when using the database $D \setminus \{t\}$ except for individual t 's bin, which differs by one; however, San replaces the count i for individual t 's bin with $A(i)$, and the algorithm A satisfies $A(i) \approx_{\epsilon} A(i - 1)$, so $\text{San}(D) \approx_{\epsilon} \text{San}(D \setminus \{t\})$, as required.

We can choose the algorithm A to be $A(j) = j + \text{Lap}(\frac{1}{\epsilon})$, where $\text{Lap}(\lambda)$ is (a random variable with) the Laplace distribution with probability density function $f_{\lambda}(x) = \frac{1}{2\lambda} e^{|x|/\lambda}$. The proof that $A(j) \approx_{\epsilon} A(j - 1)$ for every $0 < j < k$ is simple and can be implicitly found in [4].

The differentially private mechanism in [4] for computing histograms has to add noise to every bin, while our mechanism here only adds noise to the bins that have a count that is $< k$.

Example 3 (Sanitizing a database by generalizing records safely). Many mechanisms for achieving k -anonymity involve “generalizing” the records in the input table by replacing specific values with more general values, such as replacing a specific age with an age range. If this is not done carefully, the privacy of individuals can be breached, as shown by many attacks in the past (e.g., see [16,17]). Most of these mechanisms do not satisfy crowd-blending privacy. However, if the generalization of records is done carefully, achieving crowd-blending privacy may be possible.

One example is the mechanism of [14]: Let Y be any set, and let $f : X \rightarrow Y$ be any function. We think of Y as a set of possible “generalized records”, and f is a function that maps a record to its generalized version. Let San be a mechanism that, on input a database D , applies the function f to each individual in D ; let $f(D)$ be the multi-set of images in Y . San then removes each record in $f(D)$ that appears fewer than k times in $f(D)$, and then outputs the result. It is easy to see that San is $(k, 0)$ -crowd-blending private. To see this, we note that an individual t in a database D is 0-indistinguishable by San from all the individuals in D

that also get mapped to $f(t)$. If there are at least k such people, then individual t blends with k people in D ; otherwise, we have $\text{San}(D) \approx_0 \text{San}(D \setminus \{t\})$ since San removes each record in $f(D)$ that appears fewer than k times in $f(D)$.

Example 4 (Privately Releasing Synthetic Data Points in \mathbb{R}^d for Computing Smooth Functions). Roughly speaking, a function $g : (\mathbb{R}^d)^* \rightarrow \mathbb{R}^m$ is *smooth* if the value of g does not change much when we perturb the data points of the input slightly. In the full version of this paper, we show that we can *efficiently* release synthetic data points in \mathbb{R}^d for approximating *all* smooth functions simultaneously while satisfying crowd-blending privacy. On the other hand, we show that there are smooth functions that cannot even be approximated with non-trivial utility from any synthetic data that has been released with differential privacy (even if the differentially private mechanism is *inefficient*). See the full version of this paper for details.

4 Our Main Theorem

In this section, we prove our main theorem that says that when we combine a crowd-blending private mechanism with a natural *pre-sampling* step, the combined algorithm is zero-knowledge private (and thus differentially private as well). The pre-sampling step should be thought of as being part of the data collection process, where individuals in some population are sampled and asked for their data. A crowd-blending private mechanism is then run on the samples to release useful information while preserving privacy.

We first prove our main theorem for the case where the pre-sampling step samples each individual in the population with probability p independently. In reality, the sampling performed during data collection may be slightly *biased* or done slightly incorrectly, and an adversary may know whether certain individuals were sampled or not. Thus, we later extend our main theorem to the case where the sampling probability is not necessarily the same for everybody, but the sampling is still *robust* in the sense that most individuals are sampled independently with probability in between p and p' (this probability can even depend on the individual's data), where p and p' are relatively close to one another, while the remaining individuals are sampled independently with arbitrary probability.

We begin with some necessary terminology and notation. A *population* is a collection of *individuals*, where an individual is simply represented by a data value in the data universe X . Thus, a population is actually a multiset of data values, which is the same as a database. (If we want individuals to have unique data values, we can easily modify X to include personal/unique identifiers.) Given a population \mathcal{P} and a real number $p \in [0, 1]$, let $\text{Sam}(\mathcal{P}, p)$ be the outcome of sampling each individual in \mathcal{P} with probability p independently.

Although zero-knowledge privacy was originally defined for mechanisms operating on *databases*, one can also consider mechanisms operating on *populations*, since there is essentially no difference between the way we model populations and databases. (In the definition of zero-knowledge privacy, we simply change

“database” to “population” and D to \mathcal{P} .) We now describe a class of (randomized) aggregation functions that we will use in the definition of zero-knowledge privacy.

- $iidRS(p)$ = i.i.d. random sampling with probability p : the class of algorithms T such that on input a population \mathcal{P} , T chooses each individual in \mathcal{P} with probability p independently, and then performs any computation on the data of the chosen individuals.³

We now state and prove the basic version of our main theorem.

Theorem 1 (Sampling + Crowd-Blending Privacy \Rightarrow Zero-Knowledge Privacy). *Let San be any (k, ϵ) -crowd-blending private mechanism with $k \geq 2$, and let $p \in (0, 1)$. Then, the algorithm San_{zk} defined by $San_{zk}(\mathcal{P}) = San(San(\mathcal{P}, p))$ for any population \mathcal{P} is $(\epsilon_{zk}, \delta_{zk})$ -zero-knowledge private⁴ with respect to $iidRS(p)$, where*

$$\epsilon_{zk} = \ln \left(p \cdot \left(\frac{2-p}{1-p} e^\epsilon \right) + (1-p) \right) \quad \text{and} \quad \delta_{zk} = e^{-\Omega(k \cdot (1-p)^2)}.$$

The proof of Theorem 1 can be found in the full version of this paper, but the main ideas of the proof will be explained here. To prove Theorem 1, we will first prove two supporting lemmas. The first lemma essentially says that if an individual t blends with (i.e., is indistinguishable by San from) many people in the population, then t 's privacy is protected when we sample from the population and run San on the samples:

Lemma 1 (Protection of individuals that blend with many people in the population). *Let San be any mechanism, \mathcal{P} be any population, $p \in (0, 1)$, and $\epsilon \geq 0$. Let t be any individual in \mathcal{P} , and let A be any non-empty subset of $\mathcal{P} \setminus \{t\}$ such that $t' \approx_{\epsilon, San} t$ for every individual $t' \in A$. Let $n = |A|$. Then, we have*

$$San(San(\mathcal{P}, p)) \approx_{\epsilon_{final}, \delta_{final}} San(San(\mathcal{P} \setminus \{t\}, p)),$$

where $\epsilon_{final} = \ln(p \cdot (\frac{2-p}{1-p} e^\epsilon) + (1-p))$ and $\delta_{final} = e^{-\Omega((n+1)p(1-p)^2)}$.

In the lemma, A is any non-empty set of individuals in $\mathcal{P} \setminus \{t\}$ that blend with individual t . (We could set A to be the set of *all* individuals in $\mathcal{P} \setminus \{t\}$ that blend with individual t , but leaving A more general allows us to more easily extend the lemma to the case of “robust” sampling later.) We note that δ_{final} is

³ To make zero-knowledge privacy compose naturally for this type of aggregate information, we can extend $iidRS(p)$ to $iidRS(p, r)$, where T is now allowed to perform r rounds of sampling before performing any computation on the sampled data. It is not hard to see that zero-knowledge privacy with respect to $iidRS(p, r)$ composes in a natural way.

⁴ The constant hidden by the $\Omega(\cdot)$ in δ_{zk} can be easily computed; however, we did not try to optimize the constant in any way.

smaller when $n = |A|$ is larger, i.e., when t blends with more people. Intuitively, if an individual t is indistinguishable by San from many other people in the population, then t 's presence or absence in the population does not affect the output of $\text{San}(\text{Sam}(\cdot, p))$ much, since the people indistinguishable from t can essentially take the place of t in almost any situation (and the output of San would essentially be the same). Since it does not matter much whether individual t is in the population or not, it follows that t 's privacy is protected.

The proof of the lemma *roughly* works as follows: Consider two scenarios, one where individual t is in the population (i.e., $\text{San}(\text{Sam}(\mathcal{P}, p))$ in the lemma), and one where individual t has been removed from the population (i.e., $\text{San}(\text{Sam}(\mathcal{P} \setminus \{t\}, p))$ in the lemma). Our goal is to show that the output of San is essentially the same in the two scenarios, i.e., $\text{San}(\text{Sam}(\mathcal{P}, p)) \approx_{\epsilon_{final}, \delta_{final}} \text{San}(\text{Sam}(\mathcal{P} \setminus \{t\}, p))$. Conditional on individual t not being sampled in the first scenario, the two scenarios are exactly the same, as desired. Thus, we now always condition on individual t being sampled in the first scenario. In the lemma, A is a set of individuals in the population (excluding t) that are indistinguishable from t by San . Let \tilde{m} denote the number of people in A that are sampled. The proof involves showing the following two properties:

1. \tilde{m} is relatively smooth near its expectation: For every integer m near the expectation of \tilde{m} , $\Pr[\tilde{m} = m]$ is relatively close to $\Pr[\tilde{m} = m + 1]$.
2. For every integer $m \in \{0, \dots, n - 1\}$, the output of San in the first scenario conditioned on $\tilde{m} = m$ (and t being sampled) is essentially the same as the output of San in the second scenario conditioned on $\tilde{m} = m + 1$.

For the first property, we note that \tilde{m} follows a binomial distribution, which can be shown to be relatively smooth near its expectation. To show the second property, we note that when we condition on $\tilde{m} = m$ (and t being sampled) in the first scenario, m random samples are drawn uniformly from A (one at a time) without replacement, and also $t \notin A$ is sampled for sure (and the remaining individuals are sampled independently with probability p). This is very similar to the second scenario conditioned on $\tilde{m} = m + 1$, where $m + 1$ random samples are drawn uniformly from A without replacement, since if we replace the $(m + 1)^{th}$ sample by t , we get back the first scenario conditioned on $\tilde{m} = m$ (and t being sampled). Since the $(m + 1)^{th}$ sample is indistinguishable from t by San , the output of San is essentially the same in both scenarios.

Using the two properties above, one can show that when \tilde{m} is close to its expectation, the output of San is essentially the same in both scenarios. δ_{final} in the lemma captures the probability of the bad event where \tilde{m} is not close to its expectation, which we bound by essentially using a Chernoff bound. See the full version of this paper for the proof of Lemma 1.

We now show how pre-sampling combined with a crowd-blending private mechanism can protect the privacy of individuals who blend with (i.e., are indistinguishable by San from) few people in the population.

Lemma 2 (Protection of individuals that blend with few people in the population). *Let San be any (k, ϵ) -crowd-blending private mechanism with*

$k \geq 2$, let \mathcal{P} be any population, and let $p \in (0, 1)$. Let t be any individual in \mathcal{P} , and let $n = |\{t' \in \mathcal{P} \setminus \{t\} : t' \approx_{\epsilon, San} t\}|$. Then, if $n \leq \frac{k-1}{p(2-p)}$, we have

$$San(Sam(\mathcal{P}, p)) \approx_{\epsilon_{final}, \delta_{final}} San(Sam(\mathcal{P} \setminus \{t\}, p)),$$

where $\epsilon_{final} = \ln(pe^\epsilon + (1-p))$ and $\delta_{final} = pe^{-\Omega(k \cdot (1-p)^2)}$.

The proof of the lemma *roughly* works as follows: In the lemma, n is the number of people in the population that individual t blends with, and is assumed to be small. We will show that when we remove individual t from the population, the output of *San* does not change much.

Consider two scenarios, one where individual t is in the population, and one where individual t has been removed from the population. Conditional on individual t not being sampled in the first scenario, the two scenarios are exactly the same, as desired. Thus, we now always condition on individual t being sampled in the first scenario. Since individual t blends with few people in the population, we have that with very high probability, the database obtained from sampling from the population would contain fewer than k people that blend with individual t ; since *San* is (k, ϵ) -crowd-blending private and individual t does not blend in a crowd of k people in the database, *San* must essentially ignore individual t 's data; thus, the first scenario is essentially the same as the second scenario, since individual t 's data is essentially ignored anyway. δ_{final} in the lemma captures the probability of the bad event where the database obtained from sampling actually contains k people that blend with individual t . See the full version of this paper for the proof of Lemma 2.

We are now ready to prove Theorem 1. The proof of the theorem roughly works as follows: By definition of *iidRS*(p), a simulator in the definition of zero-knowledge privacy is able to obtain the aggregate information $Sam(\mathcal{P} \setminus \{t\}, p)$. With $Sam(\mathcal{P} \setminus \{t\}, p)$, the simulator can easily compute $San(Sam(\mathcal{P} \setminus \{t\}, p))$, which it can then use to simulate the computation of the given adversary. It is not hard to see that the simulation works if $San(Sam(\mathcal{P}, p)) \approx_{\epsilon_{zk}, \delta_{zk}} San(Sam(\mathcal{P} \setminus \{t\}, p))$ holds. Thus, consider any population \mathcal{P} and any individual $t \in \mathcal{P}$. Recall that Lemma 1 protects the privacy of individuals that blend with many people in \mathcal{P} , while Lemma 2 protects the privacy of individuals that blend with few people in \mathcal{P} . Thus, if individual t blends with many people in \mathcal{P} , we use Lemma 1; otherwise, we use Lemma 2. It then follows that $San(Sam(\mathcal{P}, p)) \approx_{\epsilon_{zk}, \delta_{zk}} San(Sam(\mathcal{P} \setminus \{t\}, p))$, as required. See the full version of this paper for the proof of Theorem 1.

4.1 Our Main Theorem Extended to Robust Sampling

We now extend our main theorem to the case where the sampling probability is not necessarily the same for everybody, but the sampling is still “robust” in the sense that most individuals are sampled independently with probability in between p and p' (this probability can even depend on the individual's data), where p and p' are relatively close to one another (i.e., $\frac{p'}{p}$ is not too large), while the remaining individuals are sampled independently with arbitrary probability.

We begin with some more notation. Given a population \mathcal{P} and a function $\pi : X \rightarrow [0, 1]$, let $\text{Sam}(\mathcal{P}, \pi)$ be the outcome of sampling each individual t in \mathcal{P} with probability $\pi(t)$ independently. We note that for $\text{Sam}(\mathcal{P}, \pi)$, two individuals in \mathcal{P} with the same data value in X will have the same probability of being sampled. However, we can easily modify the data universe X to include personal/unique identifiers so that we can represent an individual by a unique data value in X . Thus, for convenience, we now define a population to be a subset of the data universe X instead of being a multiset of data values in X . Then, each individual in a population would have a unique data value in X , so π does not have to assign the same sampling probability to two different individuals. We now describe a class of aggregation functions that we will use in the definition of zero-knowledge privacy.

- $iRS(p, p', \ell)$ = independent random sampling with probability in between p and p' except for ℓ individuals: the class of algorithms T such that on input a population \mathcal{P} , T independently chooses each individual $t \in \mathcal{P}$ with some probability $p_t \in [0, 1]$ (possibly dependent on t 's data), but all except for at most ℓ individuals in \mathcal{P} must be chosen with probability in $\{0\} \cup [p, p']$; T then performs any computation on the chosen individuals' data.

We now state the extended version of our main theorem.

Theorem 2 (Robust Sampling + Crowd-Blending Privacy \Rightarrow Zero-Knowledge Privacy). *Let San be any (k, ϵ) -crowd-blending private mechanism with $k \geq 2$, let $0 < p \leq p' < 1$, let $\pi : X \rightarrow [0, 1]$ be any function, let $\ell = |\{x \in X : \pi(x) \notin \{0\} \cup [p, p']\}|$, and let $p_{\max} = \sup_{x \in X} \pi(x)$. Suppose $\ell < k - 1$.*

Then, the algorithm San_{zk} defined by $\text{San}_{zk}(\mathcal{P}) = \text{San}(\text{Sam}(\mathcal{P}, \pi))$ for any population \mathcal{P} is $(\epsilon_{zk}, \delta_{zk})$ -zero-knowledge private with respect to $iRS(p, p', \ell)$, where

$$\begin{aligned}\epsilon_{zk} &= \ln \left(p_{\max} \cdot \left(\frac{p'}{p} \frac{(1-p)(2-p)}{(1-p')^2} e^\epsilon \right) + (1 - p_{\max}) \right) \text{ and} \\ \delta_{zk} &= \max \left\{ \frac{p_{\max}}{p}, \frac{p_{\max}}{1-p'} \right\} e^{-\Omega((k-\ell) \cdot (1-p')^2)}.\end{aligned}$$

In the theorem, ℓ represents the number of individuals that are sampled with probability outside of $\{0\} \cup [p, p']$. We prove the theorem by extending Lemmas 1 and 2 to the case of “robust” sampling. We now describe *some* (but not all) of the main changes to the lemmas and their proofs (see the full version of this paper for the proof of Theorem 2 and the extended lemmas).

Let us first consider Lemma 1, which protects the privacy of individuals that blend with many people in the population. Like before, consider two scenarios, one where individual t is in the population, and one where individual t has been removed. Let \tilde{m} denote the number of people in A that are sampled (recall that A is a set of individuals that blend with individual t). Recall that in the proof of Lemma 1, we had to show two properties: (1) \tilde{m} is relatively smooth near

its expectation, and (2) the output of *San* in the first scenario conditioned on $\tilde{m} = m$ (and t being sampled) is essentially the same as the output of *San* in the second scenario conditioned on $\tilde{m} = m + 1$.

For the first property, we used the fact that the binomial distribution is relatively smooth near its expectation. Here, since the sampling is no longer i.i.d. but is still robust, we need the Poisson binomial distribution (the sum of independent Bernoulli trials, where the success probabilities are not necessarily the same) to be relatively smooth near its expectation. This can be shown as long as the success probabilities are all relatively close to one another; this is ensured by changing the lemma so that everyone in the set A is required to have a sampling probability in $[p, p']$.

For the second property, we used the fact that when we condition on $\tilde{m} = m+1$ in the second scenario, we are drawing $m + 1$ random samples from A (one at a time) uniformly without replacement, and if we replace the $(m + 1)^{th}$ sample by t , we get the first scenario conditioned on $\tilde{m} = m$ and t being sampled. This idea still works in the new setting where the sampling probabilities are no longer the same, since there is still a “draw-by-draw” selection procedure for drawing samples from A (one at a time) in a way so that right after drawing the j^{th} sample, the distribution of samples we currently have is the same as if we have conditioned on $\tilde{m} = j$ (e.g., see Section 3 in [21]).

We now consider Lemma 2, which protects the privacy of individuals that blend with few people in the population. The extension of Lemma 2 to robust sampling redefines what is meant by “few people”, since even if an individual blends with few people, many of them could be sampled with probability 1. With this modification, the proof of the extended lemma is similar to the proof of the original lemma.

When we prove the extended theorem using the extended lemmas, when we are trying to show that privacy holds for individual t , we look at how many people blend with t that are sampled with probability in $[p, p']$ (in particular, we exclude the ℓ people that are sampled with probability outside of $\{0\} \cup [p, p']$); similar to before, if this number is large, we use the extended version of Lemma 1; otherwise, we use the extended version of Lemma 2. See the full version of this paper for the proof of Theorem 2.

References

- Chen, B.C., Kifer, D., LeFevre, K., Machanavajjhala, A.: Privacy-preserving data publishing. *Foundations and Trends in Databases* 2(1-2), 1–167 (2009)
- Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.* 42(4), 1–53 (2010)
- Kifer, D.: Attacks on privacy and definetti’s theorem. In: SIGMOD Conference, pp. 127–138 (2009)
- Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) *TCC 2006. LNCS*, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
- Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) *ICALP 2006. LNCS*, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)

6. Dwork, C.: The Differential Privacy Frontier (Extended Abstract). In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 496–502. Springer, Heidelberg (2009)
7. Dwork, C.: Differential Privacy: A Survey of Results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
8. Gehrke, J., Lui, E., Pass, R.: Towards Privacy for Social Networks: A Zero-Knowledge Based Definition of Privacy. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 432–449. Springer, Heidelberg (2011)
9. Dwork, C., Rothblum, G., Vadhan, S.: Boosting and differential privacy. In: Proc. of the 51st Annual IEEE Symposium on Foundations of Computer Science (2010)
10. Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: STOC 2008: Proc. of the 40th Annual ACM Symposium on Theory of Computing, pp. 609–618 (2008)
11. Chaudhuri, K., Mishra, N.: When Random Sampling Preserves Privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 198–213. Springer, Heidelberg (2006)
12. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: STOC 2007, pp. 75–84 (2007)
13. Kasiviswanathan, S., Lee, H., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? In: Foundations of Computer Science 2008, pp. 531–540 (2008)
14. Li, N., Qardaji, W.H., Su, D.: Provably private data anonymization: Or, k-anonymity meets differential privacy (2011) (manuscript)
15. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10, 557–570 (2002)
16. Wong, R.C.W., Fu, A.W.C., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. In: Proceedings of the 33rd International Conference on Very Large Data Bases. VLDB 2007, pp. 543–554. VLDB Endowment (2007)
17. Zhang, L., Jajodia, S., Brodsky, A.: Information disclosure under realistic assumptions: privacy versus optimality. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 573–583. ACM (2007)
18. Chawla, S., Dwork, C., McSherry, F., Smith, A., Wee, H.: Toward Privacy in Public Databases. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 363–385. Springer, Heidelberg (2005)
19. Ullman, J., Vadhan, S.: PCPs and the Hardness of Generating Private Synthetic Data. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 400–416. Springer, Heidelberg (2011)
20. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 381–390 (2009)
21. Chen, X.H., Dempster, A.P., Liu, J.S.: Weighted finite population sampling to maximize entropy. Biometrika 81(3), 457–469 (1994)

Differential Privacy with Imperfect Randomness

Yevgeniy Dodis^{1,*}, Adriana López-Alt¹, Ilya Mironov², and Salil Vadhan^{3,**}

¹ New York University

{dodis,lopez}@cs.nyu.edu

² Microsoft Research Silicon Valley

mironov@microsoft.com

³ Harvard University

salil@seas.harvard.edu

Abstract. In this work we revisit the question of basing cryptography on imperfect randomness. Bosley and Dodis (TCC’07) showed that if a source of randomness \mathcal{R} is “good enough” to generate a secret key capable of encrypting k bits, then one can deterministically extract nearly k almost uniform bits from \mathcal{R} , suggesting that traditional privacy notions (namely, indistinguishability of encryption) requires an “extractable” source of randomness. Other, even stronger impossibility results are known for achieving privacy under specific “non-extractable” sources of randomness, such as the γ -Santha-Vazirani (SV) source, where each next bit has fresh entropy, but is allowed to have a small bias $\gamma < 1$ (possibly depending on prior bits).

We ask whether similar negative results also hold for a more recent notion of privacy called *differential privacy* (Dwork et al., TCC’06), concentrating, in particular, on achieving differential privacy with the Santha-Vazirani source. We show that the answer is *no*. Specifically, we give a differentially private mechanism for approximating arbitrary “low sensitivity” functions that works even with randomness coming from a γ -Santha-Vazirani source, for any $\gamma < 1$. This provides a somewhat surprising “separation” between traditional privacy and differential privacy with respect to imperfect randomness.

Interestingly, the design of our mechanism is quite different from the traditional “additive-noise” mechanisms (e.g., Laplace mechanism) successfully utilized to achieve differential privacy with perfect randomness. Indeed, we show that *any* (non-trivial) “SV-robust” mechanism for our problem requires a demanding property called *consistent sampling*, which is strictly stronger than differential privacy, and cannot be satisfied by any additive-noise mechanism.

1 Introduction

Most cryptographic algorithms require randomness (for example, to generate their keys, probabilistically encrypt messages, etc.). Usually, one assumes that

* Supported by NSF Grants CNS-1065134, CNS-1065288, CNS-1017471, CNS-0831299 and Google Faculty Award.

** Supported by a gift from Google, Inc. Work done in part while on leave as a Visiting Researcher at Microsoft Research SVC and a Visiting Scholar at Stanford University.

perfect randomness is available, but in many situations this assumption is problematic, and one has to deal with more realistic, “imperfect” sources of randomness \mathcal{R} . Of course, if one can (deterministically) extract nearly perfect randomness from \mathcal{R} , then one can easily implement traditional cryptographic schemes with \mathcal{R} . Unfortunately, many natural sources are not extractable [5, 21, 24]. The simplest example of such a source is the Santha-Vazirani (SV) source [21], which produces an infinite sequence of (possibly correlated) bits $\mathbf{x} = x_1, x_2, \dots$, with the property that $\Pr[x_i = 0 | x_1 \dots x_{i-1}] \in [\frac{1}{2}(1-\gamma), \frac{1}{2}(1+\gamma)]$, for any setting of the prior bits $x_1 \dots x_{i-1}$. Namely, each bit has almost one bit of fresh entropy, but can have a small bias $\gamma < 1$ (possibly dependent on the prior bits). Yet, the celebrated result of Santha and Vazirani [21] showed that there exists no deterministic extractor $\text{Ext}: \{0,1\}^n \rightarrow \{0,1\}$ capable of extracting even a *single* bit of bias *strictly* less than γ from the γ -SV source, irrespective of how many SV bits $x_1 \dots x_n$ it is willing to wait for. In particular, outputting the first bit is already optimal in terms of traditional extraction.

Despite this pessimistic result, ruling out the “black-box compiler” from perfect to imperfect (e.g., SV) randomness for *all* applications, one may still hope that specific “non-extractable” sources, such as SV-sources, might be sufficient for *concrete* applications, such as simulating probabilistic algorithms or cryptography. Indeed, a series of celebrated results [1, 5, 21, 22, 24] showed that very “weak” sources (including SV-sources and much more) are sufficient for simulating probabilistic polynomial-time algorithms; namely, for problems which do not inherently need randomness, but which could potentially be sped up using randomization. Moreover, even in the area of cryptography — where randomness is *essential* (e.g., for key generation) — it turns out that many “non-extractable” sources (again, including SV sources and more) are sufficient for *authentication* applications, such as the designs of MACs [7, 16] and even signature schemes [8] (under appropriate hardness assumptions). Intuitively, the reason for the latter “success story” is that authentication applications only require that it is hard for the attacker to completely guess (i.e., “forge”) some long string, so having (min-)entropy in our source \mathcal{R} should be sufficient to achieve this goal.

Privacy with Imperfect Randomness? In contrast, the situation appears to be much less bright when dealing with *privacy* applications, such as encryption, commitment, zero-knowledge, etc. First, McInnes and Pinkas [18] showed that unconditionally secure symmetric encryption cannot be based on SV sources, even if one is restricted to encrypting a single bit. This result was subsequently strengthened by Dodis et al. [8], who showed that SV sources are not sufficient for building even computationally secure encryption (again, even of a single bit), and, if fact, essentially any other cryptographic task involving “privacy” (e.g., commitment, zero-knowledge, secret sharing, etc.). Finally, Bosley and Dodis [3] showed an even more general result: if a source of randomness \mathcal{R} is “good enough” to generate a secret key capable of encrypting k bits, then one

can deterministically extract nearly k almost uniform bits from \mathcal{R} , suggesting that traditional privacy *requires* an “extractable” source of randomness.¹

In this work we ask the question if similar pessimistic conclusions also hold for a more recent, but already very influential variant of privacy called *differential privacy* (DP), introduced by Dwork et al. [10], concentrating in particular on achieving differential privacy with the simple Santha-Vazirani source.

MAIN QUESTION: Is it possible to achieve (non-trivial) differential privacy with SV-sources?

As our main result, we give a *positive* answer to this question, showing a somewhat surprising “separation” between traditional privacy and differential privacy. But, first, let us examine the above question more closely, gradually explaining the path towards our solution.

Differential Privacy. Differential privacy was introduced for the purposes of allowing the owner of a sensitive database D to securely release some “aggregate statistics” $f(D)$ while protecting the privacy of individual users whose data is in D . Unfortunately, revealing $f(D)$ by itself might violate the privacy of some individual records, especially if the attacker has some partial information about D . Instead, we wish to design a *randomized mechanism* $M(D; \mathbf{r})$ which will approximate $f(D)$ with relatively high accuracy, but will use its randomness \mathbf{r} to “add enough noise” to the true answer $f(D)$ to protect the privacy of the *individual* records of D . For simplicity, we will restrict our attention to real-valued queries f , so that we can define the *utility* ρ of M as the expected value (over uniform \mathbf{r} , for now) of $|f(D) - M(D; \mathbf{r})|$, which we want to minimize. For example, f might be a *counting query*, where $f(D)$ is the number of records in D satisfying some predicate π , in which case we seek to achieve utility $o(|D|)$ or even independent of $|D|$. More interestingly, we want M to satisfy the following very strong notion called ε -*differential privacy*: for any *neighboring databases* D_1 and D_2 (i.e. D_1 and D_2 differ on a single record) and for any potential output z , $\Pr_{\mathbf{r}}[M(D_1; \mathbf{r}) = z] / \Pr_{\mathbf{r}}[M(D_2; \mathbf{r}) = z]$ is between $e^{-\varepsilon} \approx 1 - \varepsilon$ and $e^{\varepsilon} \approx 1 + \varepsilon$ (assuming ε is close to 0). This definition shows one difference between standard privacy, which holds between *all* pairs of databases D_1 and D_2 , and differential privacy, which only holds for *neighboring* databases. Related to the above, one cannot achieve any useful utility ρ if ε is required to be negligibly small (as then one can gradually transfer any D_1 to any other D_2 without noticeably changing the answers given by M). Instead, the one typically assumes that ε is a small constant *which can be pushed arbitrarily close to 0*, possibly at the expense of worse utility ρ . Motivated by these considerations, we will say that f admits a class of *non-trivial* mechanisms $\mathcal{M} = \{M_\varepsilon \mid \varepsilon > 0\}$ if there exists some fixed function $g(\cdot)$ s.t., for *all* $\varepsilon > 0$, M_ε is ε -DP and has utility $g(\varepsilon)$, independent of the size of the database D .

¹ On the positive side, [9], [3] showed that extractable sources are not strictly necessary for encrypting a “very small” number of bits. Still, for natural “non-extractable” sources, such as SV sources, it is known that encrypting even a single bit is impossible [8, 21].

Additive-Noise Mechanisms. The simplest class of non-trivial differentially private mechanisms (with perfect randomness) are the so called *additive-noise mechanisms* [10, 12, 13], introduced in the original work of [2, 6, 10, 11]. These mechanisms have the form $M(D; \mathbf{r}) = f(D) + X(\mathbf{r})$, where X is an appropriately chosen “noise” distribution added to guarantee ε -DP. For example, for counting queries (and more general “low-sensitivity” queries where $|f(D_1) - f(D_2)|$ is bounded on all neighboring databases D_1 and D_2), the right distribution is the *Laplace* distribution with standard deviation $\Theta(1/\varepsilon)$ [10], giving the (additive-noise) Laplace mechanism for such functions, which is private and accurate (in fact, essentially optimal for a wide range of loss functions [12]). One perceived advantage of additive-noise mechanisms comes from the fact that the noise is oblivious to the input, and it is natural to ask if it is possible to design additive-noise mechanisms which would be non-trivial even if the noise distribution is generated using the Santha-Vazirani source. For example, perhaps one can generate a “good enough” sample of the Laplace distribution even with SV sources? Unfortunately, we show that this is not the case. In fact, any non-trivial additive-noise mechanism for a source \mathcal{R} implies the existence of a randomness extractor for \mathcal{R} , essentially collapsing the notion of differential privacy to that of traditional privacy, and showing the impossibility of non-trivial additive-noise mechanisms for SV sources.

Need for Consistent Sampling. In fact, the main reason why additive-noise mechanisms failed to handle SV sources comes from the fact that such algorithms use *disjoint sets* of coins to produce the same “noisy answer” on two databases having different “real answers”. More formally, if $f(D_1) \neq f(D_2)$ and $T_i(z)$ is the set of coins \mathbf{r} where $M(D_i; \mathbf{r}) = z$, an additive-noise mechanism must satisfy $T_1(z) \cap T_2(z) = \emptyset$. On the other hand, ε -DP requires that $\Pr[\mathbf{r} \in T_1(z)] / \Pr[\mathbf{r} \in T_2(z)] \leq 1 + \varepsilon$. For the uniform distribution, this simply means that $|T_1| \approx |T_2|$. Since T_1 and T_2 are disjoint, the SV adversary can try to bias the coins \mathbf{r} so as *simultaneously* increase (or, at least maintain) the odds of hitting T_1 , while decreasing the odds of hitting T_2 . Indeed, in Lemma 2 we show that an SV adversary can always succeed in amplifying the ratio $\Pr[\mathbf{r} \in T_1] / \Pr[\mathbf{r} \in T_2]$ (and, hence, violate the differential privacy of our mechanism) whenever T_1 and T_2 have small intersection (e.g., are disjoint).

In fact, in Lemma 6 we prove that any “SV-robust” mechanism should strive to produce *identical* outputs on neighboring databases *for a majority of random tapes*; in particular, for any z , $|T_1(z) \cap T_2(z)| \approx |T_1(z)| \approx |T_2(z)|$ (see Definition 8 for the exact quantitative formulation). This general property, which we call *consistent sampling* (CS), is closely related to the “consistent sampling” methodology that has found applications in web search [4] and parallel repetition theorems [14], among others. Moreover, we show that ε -consistent sampling implies ε -differential privacy, but the converse is false.

Our Main Result. The lower bound above suggests a path forward toward building SV-robust mechanisms, which starts with the design of consistently samplable mechanisms. For example, the classical Laplace mechanism for low

sensitivity functions could be viewed as sampling some noise x of expected magnitude $\rho = O(1/\varepsilon)$, and adding it to the exact solution $y = f(D)$. Being additive-noise, this mechanism is not CS. But, imagine a new mechanism which further rounds the answer $z = y + x$ to the nearest multiple z' of $1/\varepsilon$. Clearly, the expected utility has gone from ρ to at most $\rho' = \rho + 1/\varepsilon = O(\rho)$. Yet, it turns out that the new mechanism is now ε -CS, since, informally, the *rounded* answers on neighboring databases are only distinct on an ε -fraction of coins \mathbf{r} (see Section 5).

Still, designing CS mechanisms was only a *necessary* condition for building SV-robust, differentially private mechanisms. For example, the basic notion of consistency ignores the binary representations of random coins \mathbf{r} defining the needed pre-image sets T_1 and T_2 , which are (intuitively) very important for handling SV sources since their randomness properties are bit-by-bit. Indeed, we show that consistency alone is *not* enough for SV-robustness, and we need an additional (fortunately, simply stated) property of our sampling to guarantee the latter. (As expected, this property asks something quite natural about the binary representations of the coins inside T_1 and T_2 .) We call the resulting notion *SV-consistent sampling* (SVCS; Definition 10). Building a non-trivial mechanism satisfying this condition formed the main technical bulk of our work.

In particular, starting with the “rounded” Laplace mechanism, we show a careful implementation of this CS mechanism, so that the resulting mechanism is actually SVCS (with appropriate parameters guaranteeing ε -DP of the final mechanism against γ -SV sources). The details of this technical step, which uses properties of *arithmetic coding* (see [19, 23]) applied to the specific Laplace distribution, are explained in Section 5. This gives us our main result (Theorem 2) and an affirmative answer to our Main Question: a non-trivial class of SV-robust mechanisms for counting queries and arbitrary low-sensitivity functions.

Due to space constraints, we defer all proofs to the full version.

2 Random Sources and Differential Privacy

Notation. For a positive integer n , we use the notation $[n]$ to denote the set $\{1, 2, \dots, n\}$. We use $\lfloor \cdot \rfloor$ to denote the nearest integer function. For a distribution or random variable R , we write $r \leftarrow R$ to denote the operation of sampling a random r according to R . For a randomized function h , we write $h(x ; r)$ to denote the unique output of f on input x with random coins r . When the distribution of random coins R is understood from context, we write $h(x)$ to denote the random variable $h(x ; r)$ for $r \leftarrow R$. Finally, we denote a sequence of bits using boldface, e.g. $\mathbf{x} = x_1, x_2, \dots$.

We use calligraphic letters to denote families of the corresponding letter. For example, \mathcal{F} denotes a family of functions f , \mathcal{R} denotes a family of distributions R . We see a distribution over $\{0, 1\}^*$ as continuously outputting (possibly correlated) bits. In particular, we let U be the distribution over $\{0, 1\}^*$ that samples each bit independently and uniformly at random. When U is truncated after n bits, the result is the distribution U_n , which is the uniform distribution over $\{0, 1\}^n$, the bit-strings of length n .

2.1 Random Sources

We call a family \mathcal{R} of distributions over $\{0, 1\}^*$ a *source*. In this work, we model perfect randomness with the uniform source and imperfect randomness with the γ -Santha-Vazirani source [21], arguably the simplest type of a “non-extractable” source. The *uniform source* $\mathcal{U} \stackrel{\text{def}}{=} \{\mathbf{U}\}$ is the set containing only the distribution \mathbf{U} on $\{0, 1\}^*$ that samples each bit uniformly at random. We define γ -Santha-Vazirani sources below.

Definition 1 (γ -Santha-Vazirani Source [21]). *Let $\gamma \in [0, 1]$. A probability distribution $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \dots)$ over $\{0, 1\}^*$ is a γ -Santha-Vazirani distribution if for all $i \in \mathbb{Z}^+$ and $x_1 \dots x_{i-1} \in \{0, 1\}^{i-1}$, it holds that*

$$\frac{1}{2}(1 - \gamma) \leq \Pr[\mathbf{X}_i = 0 \mid \mathbf{X}_1 = x_1, \dots, \mathbf{X}_{i-1} = x_{i-1}] \leq \frac{1}{2}(1 + \gamma).$$

We define the γ -Santha-Vazirani source $\mathcal{SV}(\gamma)$ to be the set of all γ -Santha-Vazirani distributions. Finally, for a distribution $\mathbf{SV}(\gamma) \in \mathcal{SV}(\gamma)$, we let $\mathbf{SV}(\gamma, n)$ be the distribution $\mathbf{SV}(\gamma)$ restricted to the first n coins $(\mathbf{X}_1, \dots, \mathbf{X}_n)$. We let $\mathcal{SV}(\gamma, n)$ be the set of all distributions $\mathbf{SV}(\gamma, n)$.

We now define γ -biased semi-flat sources, which were introduced by [20] (see also [8], where they were referred to as γ -biased halfspace sources).

Definition 2 (γ -Biased Semi-Flat Source). *For $S \subset \{0, 1\}^n$ of size $|S| = 2^{n-1}$, and $\gamma \in [0, 1]$, the distribution $\mathbf{H}_S(\gamma, n)$ over $\{0, 1\}^n$ is defined as follows: for all $x \in S$, $\Pr_{x \leftarrow \mathbf{H}_S(\gamma, n)}[x] = (1 + \gamma) \cdot 2^{-n}$, and for all $x \notin S$, $\Pr_{x \leftarrow \mathbf{H}_S(\gamma, n)}[x] = (1 - \gamma) \cdot 2^{-n}$. We define the γ -biased semi-flat source $\mathcal{H}(\gamma, n)$ to be the set of all distributions $\mathbf{H}_S(\gamma, n)$ for all $|S| = 2^{n-1}$.*

Lemma 1 ([8, 20]). *For any $n \in \mathbb{Z}^+$ and $\gamma \in [0, 1]$, $\mathcal{H}(\gamma, n) \subset \mathcal{SV}(\gamma, n)$.*

We prove a general lemma about γ -semi-flat sources, which will be very useful in later sections.

Lemma 2. *Let $G, B \subseteq \{0, 1\}^n$ such that $|G| \geq |B| > 0$, and let $\sigma \stackrel{\text{def}}{=} \frac{|B \setminus G|}{|B|} \in [0, 1]$. Then there exists $S \subseteq \{0, 1\}^n$ of size $|S| = 2^{n-1}$ such that*

$$\frac{\Pr_{\mathbf{r} \leftarrow \mathbf{H}_S(\gamma, n)}[\mathbf{r} \in G]}{\Pr_{\mathbf{r} \leftarrow \mathbf{H}_S(\gamma, n)}[\mathbf{r} \in B]} \geq (1 + \gamma\sigma) \cdot \frac{|G|}{|B|}.$$

2.2 Differential Privacy and Utility

We start by briefly recalling the notion of differential privacy. Given a database containing confidential information, we wish to allow learning of statistical information about the contents of the database without violating the privacy of any of its individual entries. The standard cryptographic notion of privacy where negligible information is revealed, is not appropriate in this setting as it does not allow to learn even one bit of “global” information about the contents of the database. Therefore, a new privacy definition is needed for this setting,

in particular, one that allows a better trade-off between privacy and utility. This is precisely what differential privacy achieves.

The Model. We model a *statistical database* as an array of rows, and say that two databases are *neighboring* if they differ in exactly one row. Throughout the paper, we let \mathcal{D} be the space of all databases. We consider the *interactive* setting, in which interested parties submit queries, modeled as functions $f: \mathcal{D} \rightarrow \mathcal{Z}$, where \mathcal{Z} is a specified range. In this paper, we are only concerned with queries with range $\mathcal{Z} = \mathbb{Z}$, and henceforth only consider this case. A *mechanism* M is a probabilistic algorithm that takes as input a database $D \in \mathcal{D}$ and a query $f: \mathcal{D} \rightarrow \mathbb{Z}$, and outputs a value $z \in \mathbb{Z}$. We assume that M 's random tape is in $\{0, 1\}^*$, that is, that M has at its disposal a possibly infinite number of random bits, but for a fixed outcome $z \in \mathbb{Z}$, M needs only a finite number of coins $n = n(D, f, z)$ to determine whether $M(D, f) = z$. Furthermore, we assume that if $\mathbf{r} \in \{0, 1\}^n$ is a prefix of $\mathbf{r}' \in \{0, 1\}^{n'}$ and $M(D, f ; \mathbf{r}) = z$ is already determined from \mathbf{r} , then $M(D, f ; \mathbf{r}') = z$ also. In other words, providing M with extra coins does not change its output.

Definitions. Informally, we wish $z = M(D, f)$ to approximate the true answer $f(D)$ without revealing too much information. We say a mechanism is *differentially private* for a class of queries \mathcal{F} if for all queries $f \in \mathcal{F}$, replacing a real entry in the database with one containing fake information only changes the outcome of the mechanism by a small amount. In other words, evaluating the mechanism on the same query $f \in \mathcal{F}$, on two neighboring databases, does not change the output by much. On the other hand, we define its utility to be the expected difference between the true answer $f(D)$ and the output of the mechanism. Since the purpose of this work is to analyze mechanisms with respect to their sources of randomness, the following definitions of privacy and utility explicitly take the source of randomness \mathcal{R} into account.

Definition 3 ((ε, \mathcal{R})-Differential Privacy). Let $\varepsilon \geq 0$, \mathcal{R} be a source, and $\mathcal{F} = \{f: \mathcal{D} \rightarrow \mathbb{Z}\}$ be a class of functions. A mechanism M is $(\varepsilon, \mathcal{R})$ -differentially private for \mathcal{F} if for any pair $D_1, D_2 \in \mathcal{D}$ of neighboring databases, all $f \in \mathcal{F}$, all possible outputs $z \in \mathbb{Z}$ of M , and all $\mathbf{R} \in \mathcal{R}$:

$$\frac{\Pr_{\mathbf{r} \leftarrow \mathcal{R}}[M(D_1, f ; \mathbf{r}) = z]}{\Pr_{\mathbf{r} \leftarrow \mathcal{R}}[M(D_2, f ; \mathbf{r}) = z]} \leq 1 + \varepsilon.$$

This is a very strong definition. Not only does it give a *statistical* guarantee, making it independent of the computation power of any adversary, but it is also strictly stronger than the requirement that the statistical distance between $M(D_1, f)$ and $M(D_2, f)$ is at most ε (for example, the latter allows some low-probability outcomes of $M(D_1, f)$ to never occur under $M(D_2, f)$). We also note that, traditionally, differential privacy has been defined by having the ratio of probabilities be bounded by e^ε . We instead bound it by $1 + \varepsilon$, since this formulation makes some of our calculations slightly cleaner. This is fine since we always have $1 + \varepsilon \leq e^\varepsilon$, and, when $\varepsilon \in [0, 1]$ (which is the key range of interest), we anyway have $e^\varepsilon \approx 1 + \varepsilon$.

If a mechanism M is $(\varepsilon, \mathcal{R})$ -differentially private for some randomness source \mathcal{R} , then a mechanism M' that runs M as a black box and then performs some post-processing on the output, is also $(\varepsilon, \mathcal{R})$ -differentially private. Intuitively, this is because given only $z = M(D, f)$, M' cannot reveal more information about D than z itself. In our work we only consider the case where M' evaluates a *deterministic* function h of $z = M(D, f)$, so that M and h do not have to “share” the random source \mathcal{R} .

Lemma 3. *Let M be a $(\varepsilon, \mathcal{R})$ -differentially private mechanism, and let h be any function. Define $M'(D, f) \stackrel{\text{def}}{=} h(M(D, f))$. Then M' is $(\varepsilon, \mathcal{R})$ -differentially private.*

Definition 4 ((ρ, \mathcal{R})-Utility). *Let $\rho > 0$, let \mathcal{R} be a source, and let $\mathcal{F} = \{f: \mathcal{D} \rightarrow \mathbb{Z}\}$ be a class of functions. We say a mechanism M has (ρ, \mathcal{R}) -utility for \mathcal{F} if for all databases $D \in \mathcal{D}$, all queries $f \in \mathcal{F}$, and all distributions $\mathbf{R} \in \mathcal{R}$,*

$$\mathbb{E}_{\mathbf{r} \leftarrow \mathcal{R}}[|f(D) - M(D, f ; \mathbf{r})|] \leq \rho.$$

At the extremes, a mechanism that always outputs 0 is $(0, \mathcal{R})$ -differentially private, while a mechanism that outputs the true answer $f(D)$ has $(0, \mathcal{R})$ -utility. Neither of these mechanisms is very interesting—the first gives no utility, while the second provides no privacy. Instead, we wish to find mechanisms that achieve a good trade-off between privacy and utility. This motivates the following definition.

Definition 5 (Non-Triviality). *We say a function family \mathcal{F} admits non-trivial differentially private mechanisms w.r.t. \mathcal{R} if there exists a function $g(\cdot)$ such that for all $\varepsilon > 0$ there exists a mechanism M_ε that is $(\varepsilon, \mathcal{R})$ -differentially private and has $(g(\varepsilon), \mathcal{R})$ -utility. We call $\mathcal{M} = \{M_\varepsilon\}$ a class of non-trivial mechanism for \mathcal{F} w.r.t. \mathcal{R} .*

We make a few remarks regarding this definition. First, we require that the utility $\rho = g(\varepsilon)$ is independent of $|D|$. Second, we note that non-triviality implies that we can achieve $(\varepsilon, \mathcal{R})$ -differential privacy for *any* $\varepsilon > 0$ (possibly at the expense of utility). E.g., when $\mathcal{R} = \mathcal{SV}(\gamma)$, we should be able to achieve $\varepsilon \ll \gamma$, which is below the “extraction barrier” for SV-sources. Finally, we note that for the purpose of satisfying this definition, we can assume w.l.o.g. that $\varepsilon \leq 1$, which is anyway the case of most interest. Moreover, we can assume that $1/\varepsilon$ is an integer, since otherwise we can simply take a slightly smaller ε for which this is the case.

Infinite-Precision Mechanisms. As we will see shortly, it is sometimes easier to describe mechanisms using samples from some *continuous* random variable X , instead of using a (discrete) random tape in $\{0, 1\}^*$. Moreover, the notions of privacy, utility, and non-triviality definitions can be analogously defined for this case as well (which we omit for brevity). Of course, to actually “implement” such abstract mechanisms in practice, one must specify how to approximate them using a “finite precision” random tape in $\{0, 1\}^*$, without significantly affecting their privacy and/or utility. When perfect randomness \mathcal{U} is available, this is typically quite easy (and usually not spelled out in most differential privacy

papers), by simply approximating a continuous sample from X within some “good enough” finite precision. In contrast, our mechanisms will have to deal with imperfect randomness $\mathcal{SV}(\gamma)$, so rounding a given “continuous” mechanism into a “discrete” mechanism will be non-trivial and require utmost care. In particular, we will have to design quite special “infinite-precision” mechanisms which will be “SV-friendly” toward appropriate “finite-precision rounding”.

Additive Noise Mechanisms. One type of non-trivial mechanisms follow the following blueprint: first, they sample *data-independent* noise x from some (discrete or continuous) distribution X , calculate the true answer $f(D)$, and output $z = f(D) + x$. We call such mechanisms, *additive-noise mechanisms* (examples of additive-noises mechanisms include the Laplacian mechanism [10], the geometric mechanism [12], and the K -norm mechanism for multiple linear queries [13]). If $E[|X|]$ is bounded, then the mechanism has bounded utility. However, to argue that such bounded “noise” X is sufficient to ensure the differential privacy of such mechanisms, we must first restrict our query class \mathcal{F} . In particular, it turns out that additive-noise mechanisms achieve differential privacy for a pretty large class of useful functions, called *bounded sensitivity* functions.

Definition 6 (Sensitivity). For $f: \mathcal{D} \rightarrow \mathbb{Z}$, the sensitivity of f is defined as

$$\Delta f \stackrel{\text{def}}{=} \max_{D_1, D_2} \|f(D_1) - f(D_2)\|$$

for all neighboring databases $D_1, D_2 \in \mathcal{D}$. For $d \in \mathbb{Z}^+$, we define $\mathcal{F}_d = \{f: \mathcal{D} \rightarrow \mathbb{Z} \mid \Delta f \leq d\}$ to be the class of functions with sensitivity at most d .

Intuitively, low sensitivity functions do not change too much on neighboring databases, which suggests that relatively small noise can “mask” the difference between $f(D_1)$ and $f(D_2)$. The particular (continuous) distribution turns out to be the Laplacian distribution, defined below.

Definition 7 (Laplacian Distribution). The Laplacian distribution with mean μ and standard deviation $\sqrt{2}b$, denoted $\text{Lap}_{\mu,b}$, has probability density function $\text{Lap}_{\mu,b}(x) = (1/2b) \cdot e^{-|x-\mu|/b}$. The cumulative distribution function is $\text{CDF}_{\mu,b}^{\text{Lap}}(x) = (1/2b) \cdot (1 + \text{sgn}(x) \cdot (1 - e^{|x-\mu|/b}))$.

We also define the distribution obtained from sampling the Laplacian distribution $\text{Lap}_{\mu,b}$ and rounding to the nearest integer $\lfloor \text{Lap}_{\mu,b} \rfloor$. We call this the “rounded” Laplacian distribution and denote it by $\text{RLap}_{\mu,b}$.

In particular, for any sensitivity bound d , Dwork et al. [10] show the following class of (infinite-precision) additive-noise mechanisms $\mathcal{M}^{\text{Lap}} = \{M_\varepsilon^{\text{Lap}}\}$ is non-trivial for \mathcal{F}_d . Given a database $D \in \mathcal{D}$, a query $f \in \mathcal{F}_d$ and the target value of ε , the mechanism $M_\varepsilon^{\text{Lap}}$ computes $f(D)$ and adds noise from the Laplacian distribution with mean 0 and standard deviation $(\sqrt{2} \cdot d)/\varepsilon$; i.e. $M_\varepsilon^{\text{Lap}}(D, f) \stackrel{\text{def}}{=} f(D) + \text{Lap}_{0, d/\varepsilon}$. Equivalently, we can also view this mechanism as computing $y = f(D)$ and outputting a sample from the distribution $\text{Lap}_{y, d/\varepsilon}$. Moreover, it is easy to see that this infinite-precision mechanism achieves utility $O(d/\varepsilon)$.

In order to ensure that the output of the mechanism of [10] is an integer, the result can be rounded to the nearest integer. Since this is post-processing, by Lemma 3, the result has the same privacy guarantees. Furthermore, since $f(D) \in \mathbb{Z}$, we have $\lfloor f(D) + \text{Lap}_{0,d/\varepsilon} \rfloor = y + \lfloor \text{Lap}_{0,d/\varepsilon} \rfloor$. In particular, for queries of integer range, the mechanism of [10] can be seen as computing $y = f(D)$ and outputting $z = \text{RLap}_{y,d/\varepsilon}$. We denote this (still infinite-precision, but now integer range) variant by $M_\varepsilon^{\text{RLap}}$. Clearly, it still has utility $O(d/\varepsilon)$.

Finally, we must describe how to approximate this mechanism family $\mathcal{M}^{\text{RLap}}$ by a finite precision family $\overline{\mathcal{M}}^{\text{RLap}}$ w.r.t. \mathcal{U} , without significantly affecting privacy or utility. As it turns out, a good enough approximation can be accomplished by sampling each value $z \in \mathbb{Z}$ with precision roughly proportional to $\Pr[z]$ (under $M_\varepsilon^{\text{RLap}}$), which requires $n(z) = O(|z| \log(d/\varepsilon))$ (truly random) coins $\mathbf{U}_{n(z)}$, and increases both ε and ρ by at most a constant factor. Since we will not use the resulting (finite-precision) mechanism in this paper (indeed, we will see in Lemma 5 that no additive-noise mechanism can be non-trivial w.r.t. $\mathcal{SV}(\gamma)$), we state the end result without further justification.

Lemma 4 ([10]). *For any $d \in \mathbb{Z}^+$, there exists a family $\overline{\mathcal{M}}^{\text{RLap}} = \{\overline{M}_\varepsilon^{\text{RLap}}\}$ of non-trivial mechanisms for \mathcal{F}_d w.r.t. the uniform source \mathcal{U} , with utility function $g^{\text{RLap}}(\varepsilon) = O(d/\varepsilon)$.*

Our Question. Lemma 4 shows that for all $d \in \mathbb{Z}^+$ there exists a class of non-trivial mechanisms for \mathcal{F}_d w.r.t. \mathcal{U} . The main goal of this work is to determine if this is also true for other randomness sources, in particular, for the γ -Santha-Vazirani sources.

MAIN QUESTION (RESTATED): Does there exist a class $\mathcal{M} = \{M_\varepsilon\}$ of non-trivial mechanisms for \mathcal{F}_d w.r.t. $\mathcal{SV}(\gamma)$ for all $\gamma \in [0, 1)$? If so, can they be additive-noise mechanisms?

For clarity, from now we will focus on the case $d = 1$; however, all our results extend to any sensitivity bound d . We will prove that non-trivial mechanisms for \mathcal{F}_1 w.r.t. $\mathcal{SV}(\gamma)$ cannot be additive noise, answering the second question in the negative. Despite this, however, we will answer the first question positively by displaying a class $\mathcal{M} = \{M_\varepsilon\}$ of non-trivial (non-additive-noise) mechanisms for \mathcal{F}_1 w.r.t. $\mathcal{SV}(\gamma)$.

3 Naive Approaches and a Lower Bound

We will start by showing a few naive approaches that will explain the intuition behind why non-trivial mechanisms for \mathcal{F}_1 w.r.t. $\mathcal{SV}(\gamma)$ cannot be additive noise. Moreover, we will prove a general lower bound restricting the type of mechanisms “friendly” to SV-sources, which will motivate a very special type of mechanisms that we will introduce in Section 4.

First Attempt. A first approach to answer our main question would be to prove that *any* class of non-trivial mechanisms for \mathcal{F}_1 w.r.t. \mathcal{U} is also non-trivial w.r.t. $\mathcal{SV}(\gamma)$. This turns out to be far too optimistic. To see this, take any mechanism M w.r.t. \mathcal{U} , and assume that with high probability M needs at most

n random coins, where n is odd. Define (artificial) mechanism M' as follows. Whenever M needs a fresh coin b , M' samples n coins $b_1 \dots b_n$ and simply sets $b = \text{MAJ}_n(b_1, \dots, b_n)$, where $\text{MAJ}_n(\cdot)$ is the majority of n bits. Clearly, M' has the same differential privacy and utility guarantees as M w.r.t. \mathcal{U} , since majority of perfectly random bits is perfectly random. On the other hand, by biasing each bit towards 0 (resp. 1), a Santha-Vazirani adversary can fix every n -bit majority function to 0 (resp. 1) with probability at least $(1 - e^{-\gamma^2 n/2})$, which means that he can fix all n coins of M to any desired outcome with probability at least $(1 - ne^{-\gamma^2 n/2}) \approx 1$. Hence, the Santha-Vazirani adversary for M' can effectively fix the random tape of M , making it deterministic (with probability exponentially close to 1). On the other hand, it is easy to see that no deterministic mechanism having non-trivial utility (i.e., giving distinct answers on some two neighboring databases) can be differentially private.

Hence, non-trivial mechanisms w.r.t. the uniform source \mathcal{U} are not necessarily non-trivial w.r.t. γ -Santha-Vazirani sources $\mathcal{SV}(\gamma)$.

Second Attempt. A seemingly less naive idea would be to prove that *any* class of non-trivial mechanisms for \mathcal{F}_1 w.r.t. \mathcal{U} is also non-trivial w.r.t. $\mathcal{SV}(\gamma)$ if we first run some extractor Ext on the randomness. More precisely, suppose $\mathcal{M} = \{M_\varepsilon\}$ is non-trivial w.r.t. \mathcal{U} and suppose M_ε uses n coins. Can we construct a deterministic extractor $\text{Ext}: \{0, 1\}^m \rightarrow \{0, 1\}^n$ (for some sufficiently large $m \gg n$) and let $M'_\varepsilon \stackrel{\text{def}}{=} M_\varepsilon(D, f; \text{Ext}(\mathbf{r}))$, such that $\mathcal{M}' = \{M'_\varepsilon\}$ is non-trivial w.r.t. $\mathcal{SV}(\gamma)$ whenever $\mathcal{M} = \{M_\varepsilon\}$ is non-trivial w.r.t. \mathcal{U} ? More generally, one can define an analogous “extractor conjecture” for any imperfect source \mathcal{R} in place of $\mathcal{SV}(\gamma)$. Unfortunately, we show that this naive approach does not work for any “non-extractable” source \mathcal{R} , such as $\mathcal{SV}(\gamma)$. To show this, we look at the family of *additive-noise* mechanisms for the family \mathcal{F}_1 of sensitivity-1 functions given by Lemma 4, and observe that applying an extractor to any additive-noise mechanism is *still* an additive-noise mechanism. Then, we show a more general statement that *any* non-trivial additive-noise mechanism for \mathcal{F}_1 under \mathcal{R} implies the existence of a bit extractor for \mathcal{R} , which is impossible for non-extractable \mathcal{R} , such as $\mathcal{SV}(\gamma)$.

Lemma 5. *Assume \mathcal{R} is a source and $\mathcal{M} = \{M_\varepsilon\}$ is a family of additive-noise mechanisms for \mathcal{F}_1 , where each M_ε is $(\varepsilon, \mathcal{R})$ -differentially private. Then, for all $\varepsilon > 0$, one can deterministically extract an ε -biased bit from \mathcal{R} . In particular, (a) there does not exist a class $\mathcal{M} = \{M_\varepsilon\}$ of non-trivial additive-noise mechanisms for \mathcal{F}_1 w.r.t. $\mathcal{SV}(\gamma)$; and, by Lemma 4, (b) the “extractor-conjecture” is false for any “non-extractable” \mathcal{R} , such as $\mathcal{SV}(\gamma)$.*

General Lower Bound. The failure of our naive approaches suggests that one cannot take any non-trivial mechanism w.r.t. uniform randomness \mathcal{U} , and apply some simple transformation to its randomness to derive a non-trivial mechanism w.r.t. $\mathcal{SV}(\gamma)$. Indeed, we will show that *any* non-trivial mechanism w.r.t. $\mathcal{SV}(\gamma)$ must in fact satisfy a pretty restrictive condition w.r.t. to the uniform source. In particular, this condition (later called *consistent-sampling*) is never satisfied by additive-noise mechanisms.

First, we need some important notation. Consider a mechanism M with randomness space $\{0, 1\}^*$, and let $D \in \mathcal{D}$. We define the set $T(D, f, z) \stackrel{\text{def}}{=} \{\mathbf{r} \in \{0, 1\}^n \mid z = M(D, f ; \mathbf{r})\}$ to be the set of random coins $\mathbf{r} \in \{0, 1\}^*$ such that M outputs z when run on database D , query f , and random coins \mathbf{r} . We remark that since we assume that only $n = n(f, z, D)$ coins need to be sampled to determine if $M(D, f) = z$, we can assume w.l.o.g. that $T(f, z, D) \subseteq \{0, 1\}^n$. In the interest of clarity, we simply write T when f, D , and z are understood from context.

Without loss of generality, we assume that the function family \mathcal{F} is by itself non-trivial, meaning that there exist two neighboring databases D_1, D_2 and a query f such that $f(D_1) \neq f(D_2)$. We also let $T_1 \stackrel{\text{def}}{=} T(D_1, f, z)$ and $T_2 \stackrel{\text{def}}{=} T(D_2, f, z)$. Fix $z \in \mathbb{Z}, f \in \mathcal{F}, R \in \mathcal{R}$. To show that M is $(\varepsilon, \mathcal{R})$ -differentially private for \mathcal{F} w.r.t. randomness source \mathcal{R} , we are concerned with bounding the following ratio by $1 + \varepsilon$:

$$\frac{\Pr_{\mathbf{r} \leftarrow \mathcal{R}}[M(D_1, f ; \mathbf{r}) = z]}{\Pr_{\mathbf{r} \leftarrow \mathcal{R}}[M(D_2, f ; \mathbf{r}) = z]} = \frac{\Pr_{\mathbf{r} \leftarrow \mathcal{R}}[\mathbf{r} \in T_1]}{\Pr_{\mathbf{r} \leftarrow \mathcal{R}}[\mathbf{r} \in T_2]}$$

As we show below, bounding the above ratio for all Santha-Vazirani sources introduces a non-trivial constraint of M . For illustration, let us first look at any additive-noise mechanism M and re-derive the conclusion of Lemma 5 directly. If $z = M(D_1, f ; \mathbf{r}_1) = M(D_2, f ; \mathbf{r}_2)$ then $z = f(D_1) + x_1 = f(D_2) + x_2$ for $x_1, x_2 \leftarrow \mathbb{X}$. Since we assumed $f(D_1) \neq f(D_2)$ then $x_1 \neq x_2$, which means that $\mathbf{r}_1 \neq \mathbf{r}_2$. Thus, $T_1 \cap T_2 = \emptyset$. Furthermore, we can assume w.l.o.g. that $|T_1| \geq |T_2|$ since otherwise we can switch D_1 and D_2 . Using Lemma 2 with $G = T_1$ and $B = T_2$, and the fact that $\mathcal{H}(\gamma, n) \subset \mathcal{SV}(\gamma, n)$, we have that there exists $\mathsf{SV}(\gamma) \in \mathcal{SV}(\gamma)$ such that

$$\frac{\Pr_{\mathbf{r} \leftarrow \mathsf{SV}(\gamma)}[M(D_1, f ; \mathbf{r}) = z]}{\Pr_{\mathbf{r} \leftarrow \mathsf{SV}(\gamma)}[M(D_2, f ; \mathbf{r}) = z]} \geq (1 + \gamma) \cdot \frac{|T_1|}{|T_2|} \geq 1 + \gamma,$$

which is the same conclusion as the one obtained in the proof of Lemma 5.

More generally, coming back to arbitrary mechanisms, since Lemma 2 works even when $G \cap B \neq \emptyset$, we get the following much stronger result. Suppose $\sigma \stackrel{\text{def}}{=} \frac{|T_2 \setminus T_1|}{|T_2|} \in [0, 1]$. Then there exists $\mathsf{SV}(\gamma) \in \mathcal{SV}(\gamma)$ such that

$$\frac{\Pr_{\mathbf{r} \leftarrow \mathsf{SV}(\gamma)}[M(D_1, f ; \mathbf{r}) = z]}{\Pr_{\mathbf{r} \leftarrow \mathsf{SV}(\gamma)}[M(D_2, f ; \mathbf{r}) = z]} \geq 1 + \gamma\sigma.$$

This shows that a *necessary* condition to achieve $(\varepsilon, \mathcal{SV}(\gamma))$ -differential privacy is that $\sigma \leq \varepsilon/\gamma = O(\varepsilon)$. We summarize this in the following lemma.

Lemma 6. *Assume $\gamma > 0$ and M is $(\varepsilon, \mathcal{SV}(\gamma))$ -differentially private mechanism for some class \mathcal{F} . Fix any $z \in \mathbb{Z}, f \in \mathcal{F}$, and any neighboring databases $D_1, D_2 \in \mathcal{D}$ s.t. $f(D_1) \neq f(D_2)$. Let $T_1 \stackrel{\text{def}}{=} T(D_1, f, z), T_2 \stackrel{\text{def}}{=} T(D_2, f, z)$, and assume that $|T_1| \geq |T_2|$. Then $\sigma \stackrel{\text{def}}{=} \frac{|T_2 \setminus T_1|}{|T_2|} \leq \frac{\varepsilon}{\gamma} = O(\varepsilon)$.*

4 SV-Consistent Sampling

Recall that we defined $T(D, f, z) \stackrel{\text{def}}{=} \{\mathbf{r} \in \{0, 1\}^n \mid z = M(D, f; \mathbf{r})\}$ to be the set of all coins \mathbf{r} such that M outputs z when run on database D , query f and randomness \mathbf{r} . Further recall that for neighboring databases D_1, D_2 , we let $T_1 \stackrel{\text{def}}{=} T(D_1, f, z)$ and $T_2 \stackrel{\text{def}}{=} T(D_2, f, z)$.

By Lemma 6 we know that in order to achieve $(\varepsilon, \mathcal{SV}(\gamma))$ -differential privacy we must have $\frac{|T_2 \setminus T_1|}{|T_2|} = O(\varepsilon)$. This means that for arbitrary $\varepsilon > 0$, it must be that $\frac{|T_2 \setminus T_1|}{|T_2|} \rightarrow 0$ as $\varepsilon \rightarrow 0$. This motivates our definition of $\tilde{\varepsilon}$ -consistent sampling. Later we will define ε in terms of $\tilde{\varepsilon}$ such that $\varepsilon \rightarrow 0$ as $\tilde{\varepsilon} \rightarrow 0$. We remark that our definition of $\tilde{\varepsilon}$ -consistent sampling is similar to the definition of [14, 15], which has already been used in the context of differential privacy [17].

Definition 8. We say M has $\tilde{\varepsilon}$ -consistent sampling ($\tilde{\varepsilon}$ -CS) if for all $z \in \mathbb{Z}, f \in \mathcal{F}$ and neighboring databases $D_1, D_2 \in \mathcal{D}$ such that $T_2 \neq \emptyset$, we have

$$\frac{|T_1 \setminus T_2|}{|T_2|} \leq \tilde{\varepsilon}.$$

We make a few remarks about Definition 8. First, notice that w.l.o.g. we can assume that $|T_1| \geq |T_2|$ since in this case we have $\frac{|T_2 \setminus T_1|}{|T_1|} \leq \frac{|T_1 \setminus T_2|}{|T_2|}$. Second, notice that $\tilde{\varepsilon}$ -consistent sampling also guarantees that $\frac{|T_2 \setminus T_1|}{|T_2|} \leq \frac{|T_1 \setminus T_2|}{|T_2|} \leq \tilde{\varepsilon}$, which Lemma 6 tells us is a necessary condition for non-trivial differential privacy. Finally, it is easy to see that if a mechanism has $\tilde{\varepsilon}$ -consistent sampling, then it is $(\tilde{\varepsilon}, \mathcal{U})$ -differentially private, as

$$\frac{\Pr_{\mathbf{r} \leftarrow \mathcal{U}_n}[\mathbf{r} \in T_1]}{\Pr_{\mathbf{r} \leftarrow \mathcal{U}_n}[\mathbf{r} \in T_2]} = \frac{|T_1|}{|T_2|} = \frac{|T_1 \cap T_2|}{|T_2|} + \frac{|T_1 \setminus T_2|}{|T_2|} \leq 1 + \tilde{\varepsilon}.$$

To summarize, $\tilde{\varepsilon}$ -consistent sampling is *sufficient* to achieve $(\tilde{\varepsilon}, \mathcal{U})$ -differential privacy and is essentially *necessary* to achieve $(\gamma\tilde{\varepsilon}, \mathcal{SV}(\gamma))$ -differential privacy. But is it sufficient to achieve $(p(\tilde{\varepsilon}), \mathcal{SV}(\gamma))$ -differential privacy for some function p such that $p(\tilde{\varepsilon}) \rightarrow 0$ as $\tilde{\varepsilon} \rightarrow 0$? This turns out not to be the case, as it is still possible for a Santha-Vazirani distribution to increase the probability of $T_1 \setminus T_2$ while simultaneously decreasing the probability of T_2 . For instance, consider the example in Figure 1, where pictorially, we view each coin $\mathbf{r} \in \{0, 1\}^*$ as defining a path down a binary tree. In this example, $T_1 \setminus T_2$ and T_2 are positioned precisely to the left and right of $1/2$, respectively. After the first coin, the SV-distribution can focus on either targeting $T_1 \setminus T_2$ or avoiding T_2 . If the height of this tree is big, then the SV distribution can greatly increase our ratio. This suggests that in order to handle γ -Santha Vazirani distributions, we need to make more restrictions on the mechanism.

New Observations. We make two observations that will help us guarantee that the example described in Figure 1 does not arise, but we first define some notation. For $m \in \mathbb{Z}^+$ and a bit sequence $\mathbf{x} = x_1, \dots, x_m \in \{0, 1\}^m$, we define

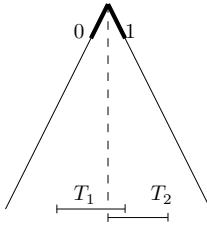


Fig. 1. Example of how a $\text{SV}(\gamma) \in \mathcal{SV}(\gamma)$ distribution can increase the ratio $\frac{\Pr_{\mathbf{r} \leftarrow \text{SV}(\gamma)}[\mathbf{r} \in T_1]}{\Pr_{\mathbf{r} \leftarrow \text{SV}(\gamma)}[\mathbf{r} \in T_2]}$

$\text{SUFFIX}(\mathbf{x}) \stackrel{\text{def}}{=} \{\mathbf{y} = y_1, y_2, \dots \in \{0, 1\}^* \mid x_i = y_i \text{ for all } i \in [m]\}$ to be the set of all bit strings that have \mathbf{x} as a prefix. For $n \in \mathbb{Z}^+$ such that $m \leq n$, we define $\text{SUFFIX}(\mathbf{x}, n) \stackrel{\text{def}}{=} \text{SUFFIX}(\mathbf{x}) \cap \{0, 1\}^n$.

Our first observation is that if we consider the longest prefix \mathbf{u} of all elements in $T_1 \cup T_2$, then the ratio is the same as when the probabilities are conditioned on \mathbf{r} having this prefix. This is because in order for $\mathbf{r} \in T_1 \setminus T_2$ or $\mathbf{r} \in T_2$, it must be the case that $\mathbf{r} \in \text{SUFFIX}(\mathbf{u}, n)$.

Our second observation is that we want to ensure that $\text{SUFFIX}(\mathbf{u}, n)$ is a good approximation of $T_1 \cup T_2$, that is, that $|\text{SUFFIX}(\mathbf{u}, n)| \approx |T_1 \cup T_2|$. This guarantees that we never encounter the problem that arose in the example in Figure 1. For this to be the case, however, we must first ensure that $T_1 \cup T_2$ are “close together”. We therefore make the following definition.

Definition 9. We say \mathcal{M} is an interval mechanism if for all queries $f \in \mathcal{F}$, databases $D \in \mathcal{D}$, and possible outcomes $z \in \mathbb{Z}$, the values in T constitute an interval, that is, $T \neq \emptyset$ and the set $\{\text{INT}(\mathbf{r}) \mid \mathbf{r} \in T\}$ contains consecutive integers, where for $\mathbf{r} = r_1 \dots r_n \in \{0, 1\}^n$, we define $\text{INT}(\mathbf{r}) \stackrel{\text{def}}{=} \sum_{i=1}^n r_i \cdot 2^{n-i}$.

We now formalize the requirement we described above. Let D_1, D_2 be two neighboring databases, let $f \in \mathcal{F}$, let z be a possible outcome, and let $n \stackrel{\text{def}}{=} \max(n(D_1, f, z), n(D_2, f, z))$. We let \mathbf{u} be the longest prefix such that $T_1 \cup T_2 \subseteq \text{SUFFIX}(\mathbf{u}, n)$. Formally,

$$\mathbf{u} \stackrel{\text{def}}{=} \text{argmax}\{|\mathbf{u}'| \mid \mathbf{u}' \in \{0, 1\}^{\leq n} \text{ and } T_1 \cup T_2 \subseteq \text{SUFFIX}(\mathbf{u}', n)\}$$

Definition 10. Let $\tilde{\varepsilon} > 0, c > 1$. We say that an interval mechanism M has $(\tilde{\varepsilon}, c)$ -SV-consistent sampling (($\tilde{\varepsilon}, c$)-SVCS) if it has $\tilde{\varepsilon}$ -consistent sampling and for all queries $f \in \mathcal{F}$, all neighboring databases $D_1, D_2 \in \mathcal{D}$ and all possible outcomes $z \in \mathbb{Z}$, which define \mathbf{u} as above, we have:

$$\frac{|\text{SUFFIX}(\mathbf{u}, n)|}{|T_1 \cup T_2|} \leq c$$

We now show that $(\tilde{\varepsilon}, c)$ -SV-consistent sampling is sufficient to obtain $(\varepsilon, \mathcal{SV}(\gamma))$ -differential privacy for an interesting value of ε , that is, for an ε that can be made arbitrarily small by decreasing $\tilde{\varepsilon}$.

Theorem 1. *If M has $(\tilde{\varepsilon}, c)$ -SV-consistent sampling, then M is $(\varepsilon, \mathcal{SV}(\gamma))$ -differentially private, where*

$$\varepsilon = 2 \cdot (8\tilde{\varepsilon})^{1-\log(1+\gamma)} \left(\frac{1+\gamma}{1-\gamma} \right)^{\log(8c)}$$

In particular, for $\gamma \in [0, 1]$ and $c = O(1)$, we have $\varepsilon \rightarrow 0$ as $\tilde{\varepsilon} \rightarrow 0$.

As part of proving Theorem 1, we make a few additional definitions and prove a lemma. Let D_1, D_2 be two neighboring databases, $f \in \mathcal{F}$, z be a possible outcome, and $n = \max(n(D_1, f, z), n(D_2, f, z))$.

- Define \mathbf{v} to be the longest prefix such that $T_1 \setminus T_2 \subseteq \text{SUFFIX}(\mathbf{v}, n)$. Formally,

$$\mathbf{v} = \operatorname{argmax}\{|\mathbf{v}'| \mid \mathbf{v}' \in \{0, 1\}^{\leq n} \text{ and } T_1 \setminus T_2 \subseteq \text{SUFFIX}(\mathbf{v}', n)\}$$

- Define $I_0 \stackrel{\text{def}}{=} \text{SUFFIX}(\mathbf{v}0, n) \cap T_1 \setminus T_2$ and $I_1 \stackrel{\text{def}}{=} \text{SUFFIX}(\mathbf{v}1, n) \cap T_1 \setminus T_2$. That is, $I_0 \cup I_1 = T_1 \setminus T_2$ and I_b contains all coins in $T_1 \setminus T_2$ that have $\mathbf{v}b$ as prefix.
- Define \mathbf{v}_0 to be the longest prefix such that $I_0 \subseteq \text{SUFFIX}(\mathbf{v}_0, n)$. Formally,

$$\mathbf{v}_0 = \operatorname{argmax}\{|\mathbf{v}'_0| \mid \mathbf{v}'_0 \in \{0, 1\}^{\leq n} \text{ and } I_0 \subseteq \text{SUFFIX}(\mathbf{v}'_0, n)\}$$

- Define \mathbf{v}_1 to be the longest prefix such that $I_1 \subseteq \text{SUFFIX}(\mathbf{v}_1, n)$. Formally,

$$\mathbf{v}_1 = \operatorname{argmax}\{|\mathbf{v}'_1| \mid \mathbf{v}'_1 \in \{0, 1\}^{\leq n} \text{ and } I_1 \subseteq \text{SUFFIX}(\mathbf{v}'_1, n)\}$$

- Define \mathbf{w} to be the shortest prefix such that $\text{SUFFIX}(\mathbf{w}, n) \subseteq T_2$. Formally,

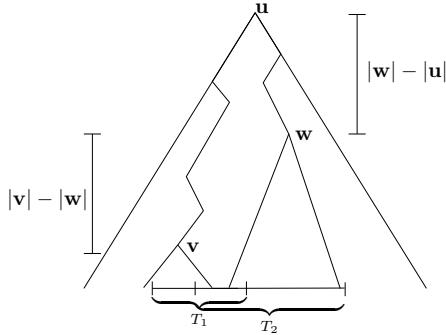
$$\mathbf{w} = \operatorname{argmin}\{|\mathbf{w}'| \mid \mathbf{w}' \in \{0, 1\}^{\leq n} \text{ and } \text{SUFFIX}(\mathbf{w}', n) \subseteq T_2\}$$

We remark that \mathbf{w} may not be unique. In this case, any of the possible values is just as good since we will be concerned with the value $|\mathbf{w}|$ which is the same across all possible values of \mathbf{w} .

See Figure 2 for a pictorial representation of $\mathbf{u}, \mathbf{v}, \mathbf{w}$. Note the asymmetry of the definitions of \mathbf{u}, \mathbf{v} , and \mathbf{w} . Also note that we define \mathbf{v} and \mathbf{w} in such a way that $\text{SUFFIX}(\mathbf{v}) \cap \text{SUFFIX}(\mathbf{w}) = \emptyset$. Informally, $\max(|\mathbf{v}_0|, |\mathbf{v}_1|) - |\mathbf{w}|$ is roughly the number of coins that the Santha-Vazirani distribution needs to use to increase the probability of landing in $T_1 \setminus T_2$ without affecting the probability of landing in T_2 , while $|\mathbf{w}| - |\mathbf{u}|$ is roughly the number of coins that it can use to decrease the probability of landing in T_2 without affecting the probability of landing in $T_1 \setminus T_2$. We first prove a lemma that says that if M has $(\tilde{\varepsilon}, c)$ -SV-consistent sampling then $\max(|\mathbf{v}_0|, |\mathbf{v}_1|) - |\mathbf{w}| = \Omega(\log(1/\tilde{\varepsilon}))$ and $|\mathbf{w}| - |\mathbf{u}| = O(1)$.

Lemma 7. *If M has $(\tilde{\varepsilon}, c)$ -SV-consistent sampling then for all neighboring databases $D_1, D_2 \in \mathcal{D}$ which define $\mathbf{u}, \mathbf{v}_0, \mathbf{v}_1, \mathbf{w}$ as above, we have:*

$$\max(|\mathbf{v}_0|, |\mathbf{v}_1|) - |\mathbf{w}| \geq \log\left(\frac{1}{8\tilde{\varepsilon}}\right) \quad \text{and} \quad |\mathbf{w}| - |\mathbf{u}| \leq \log(8c)$$

**Fig. 2.** Definitions of $\mathbf{u}, \mathbf{v}, \mathbf{w}$

5 Non-trivial SVCS Mechanisms

In this section we show a mechanism, which we call $\overline{M}_{\tilde{\epsilon}}^{\text{SVCS}}$, that achieves $(\tilde{\epsilon}, O(1))$ -SVCS for \mathcal{F}_d – the class of functions with bounded sensitivity $d \in \mathbb{Z}^+$. By Theorem 1 this gives us a $(\epsilon, \mathcal{SV}(\gamma))$ -differentially private mechanism, where $\epsilon \rightarrow 0$ as $\tilde{\epsilon} \rightarrow 0$. Furthermore, by our observation in Section 4, the mechanism is also $(\tilde{\epsilon}, \mathcal{U})$ -differentially private. We highlight that for convenience, we parametrize the mechanism $\overline{M}_{\tilde{\epsilon}}^{\text{SVCS}}$ with the privacy parameter $\tilde{\epsilon}$ w.r.t. \mathcal{U} , and state the privacy and utility guarantees w.r.t. $\mathcal{SV}(\gamma)$ as a function of $\tilde{\epsilon}$ (see Lemma 8). For clarity, we focus on the case $d = 1$.

We start with the $(\tilde{\epsilon}, \mathcal{U})$ -differentially private mechanism of Dwork et.al. [10], $M_{\tilde{\epsilon}}^{\text{RLap}}(D, f) = f(D) + \text{RLap}_{0,1/\tilde{\epsilon}}$. Note that since $M_{\tilde{\epsilon}}^{\text{RLap}}$ is additive-noise, then any finite-precision implementation will also be additive-noise, and by Lemma 5 we know it cannot be non-trivial for \mathcal{F}_1 w.r.t. $\mathcal{SV}(\gamma)$. This is because the set of random coins that make the mechanism output $z \in \mathbb{Z}$ on two neighboring databases is *disjoint*. We will therefore need to make several changes to ensure not only that these sets overlap, but that their intersection is large, thus ensuring $\tilde{\epsilon}$ -consistent sampling. Moreover, we must carefully implement our mechanism with finite precision so that the resulting mechanism is $(\tilde{\epsilon}, O(1))$ -SV-consistent, ensuring that pathological cases, such as the one in Figure 1, do not occur. Finally, in performing all these changes we must also keep in mind that we want a good bound on utility. We first describe a new infinite-precision mechanism, which we call $M_{\tilde{\epsilon}}^{\text{SVCS}}$, and then show how to implement it with finite precision to ensure $(\tilde{\epsilon}, O(1))$ -SV-consistency.

A New Infinite-Precision Mechanism. Recall that $M_{\tilde{\epsilon}}^{\text{RLap}}(D, f) = f(D) + \text{RLap}_{0,1/\tilde{\epsilon}} = \lfloor f(D) + \text{Lap}_{0,1/\tilde{\epsilon}} \rfloor$. For our new mechanism, which we call $M_{\tilde{\epsilon}}^{\text{SVCS}}$, we choose to perform the rounding step differently. $M_{\tilde{\epsilon}}^{\text{SVCS}}(D, f)$ computes $f(D) + \text{Lap}_{0,1/\tilde{\epsilon}}$ as before but then rounds the final outcome to the nearest multiple of $1/\tilde{\epsilon}$. Recall that w.l.o.g. we can assume that $1/\tilde{\epsilon} \in \mathbb{Z}$ since otherwise we can choose a smaller $\tilde{\epsilon}$ so that this is indeed the case. Formally, $M_{\tilde{\epsilon}}^{\text{SVCS}}(D, f)$

computes $y \stackrel{\text{def}}{=} f(D)$ and outputs $z \leftarrow 1/\tilde{\varepsilon} \cdot \lceil \tilde{\varepsilon} \cdot \text{Lap}_{y, 1/\tilde{\varepsilon}} \rceil$. We let Z_y denote the induced distribution of the outcome z . We remark that $M_{\tilde{\varepsilon}}^{\text{SVCS}}$ is not additive-noise, since the rounding ensures that the “noise” introduced is dependent on $y = f(D)$. Further, the output distribution is only defined on multiples of $1/\tilde{\varepsilon}$, i.e. for $k/\tilde{\varepsilon}$ where $k \in \mathbb{Z}$.

Consistent Sampling. We now give some intuition as to why this mechanism already satisfies $\tilde{\varepsilon}$ -consistent sampling. Since we are considering only queries in \mathcal{F}_1 , for any two neighboring databases D_1, D_2 , we can assume w.l.o.g. that $f(D_1) = y$ and $f(D_2) = y - 1$. Then for $k \in \mathbb{Z}$,

$$\frac{\Pr[M_{\tilde{\varepsilon}}^{\text{SVCS}}(f, D_1) = k/\tilde{\varepsilon}]}{\Pr[M_{\tilde{\varepsilon}}^{\text{SVCS}}(f, D_2) = k/\tilde{\varepsilon}]} = \frac{\Pr\left[\frac{k-1/2}{\tilde{\varepsilon}} \leq \text{Lap}_{y, 1/\tilde{\varepsilon}} < \frac{k+1/2}{\tilde{\varepsilon}}\right]}{\Pr\left[\frac{k-1/2}{\tilde{\varepsilon}} + 1 \leq \text{Lap}_{y, 1/\tilde{\varepsilon}} < \frac{k+1/2}{\tilde{\varepsilon}} + 1\right]}$$

Notice that both the intervals defined in the numerator and denominator have size $1/\tilde{\varepsilon}$, and that the interval in the denominator is simply the interval in the numerator, shifted by 1. Therefore, their intersection is roughly a $1 - \tilde{\varepsilon}$ fraction of their size, which is precisely what is required by $\tilde{\varepsilon}$ -consistent sampling. Of course, we now need to implement this $\tilde{\varepsilon}$ -consistent mechanism with finite precision, so as to achieve a stronger form of $(\tilde{\varepsilon}, O(1))$ -SV-consistency. For that, we will use *arithmetic coding* and some specific properties of the Laplace distribution.

From Infinite to Finite Precision via Arithmetic Coding. In what follows, we use the following notation: for a sequence $\mathbf{x} = x_1, x_2, \dots \in \{0, 1\}^*$, we define its *real representation* to be the real number $\text{REAL}(\mathbf{x}) \stackrel{\text{def}}{=} 0.x_1x_2x_3\dots \in [0, 1]$. Arithmetic coding gives us a way to approximate any distribution X on \mathbb{Z} from a bit string $\mathbf{r} \in \{0, 1\}^*$, as follows. Let CDF^X be the cumulative distribution of X , so that $X(x) = \text{CDF}^X(x) - \text{CDF}^X(x-1)$. Let $s(x) \stackrel{\text{def}}{=} \text{CDF}^X(x)$. Then the set of points $\{s(x)\}_{x \in \mathbb{Z}}$ partitions the interval $[0, 1]$ into infinitely many intervals $\{I^X(x) \stackrel{\text{def}}{=} [s(x-1), s(x)]\}_{x \in \mathbb{Z}}$, where $X(x) = |I^X(x)|$. Note that if a value $x \in \mathbb{Z}$ has zero probability, then we can simply ignore it as its corresponding interval will be empty. We can obtain distribution X from U by sampling a sequence of bits $\mathbf{r} = r_1, r_2, r_3, \dots$ and outputting the unique $x \in \mathbb{Z}$ such that $\text{REAL}(\mathbf{r}) \in I^X(x)$. Note that arithmetic coding has the very nice property that intervals $I^X(x)$ and $I^X(x+1)$ are always consecutive for any $x \in \mathbb{Z}$.

Since for some $x \in \mathbb{Z}$ we can have that $s(x)$ has an infinite binary decimal representation, there is no *a priori* bound on the number of coins to decide whether $\text{REAL}(\mathbf{r}) \in I^X(x)$ or $\text{REAL}(\mathbf{r}) \in I^X(x+1)$. To avoid this, we simply round each endpoint $s(x)$ to its most $n = n(x)$ significant figures, for some $n > 1$ which potentially depends on x . We will need to make sure that $n(x)$ is *legal*, in the sense that rounding with respect to $n(x)$ should not cause intervals to “disappear” or for consecutive intervals to “overlap”. We use a bar to denote rounded values: $\bar{s}(x)$ for the rounded endpoint, and $\bar{I}^X(x)$ for the rounded interval.

A New Finite Precision Mechanism. We now show how to sample Z_y , the output distribution of $M_{\tilde{\varepsilon}}^{\text{SVCS}}(D, f)$ using arithmetic coding. This yields a

new finite precision mechanism, which we call $\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}$, and let \overline{Z}_y be its output distribution which will approximate Z_y . The distribution Z_y is the Laplacian distribution $\text{Lap}_{y, 1/\tilde{\varepsilon}}$ where for all $k \in \mathbb{Z}$, the probability mass in the interval $\left[\frac{k-1/2}{\tilde{\varepsilon}}, \frac{k+1/2}{\tilde{\varepsilon}}\right)$ collapses to the point $k/\tilde{\varepsilon}$. Let $s_y(k) \stackrel{\text{def}}{=} \text{CDF}^{Z_y}\left(\frac{k+1/2}{\tilde{\varepsilon}}\right)$, and let $\overline{s}_y(k)$ be $s_y(k)$, rounded to its $n = n(y, k)$ most significant figures. Then the set of points $\{\overline{s}_y(k)\}_{k \in \mathbb{Z}}$ partition the interval $[0, 1]$ into infinitely many intervals $\{\overline{I}_y(k) \stackrel{\text{def}}{=} [\overline{s}_y(k-1), \overline{s}_y(k))\}_{k \in \mathbb{Z}}$, where $\Pr[\overline{Z}_y = k/\tilde{\varepsilon}] = |\overline{I}_y(k)|$. We obtain distribution \overline{Z}_y from U by sampling a sequence of bits $\mathbf{r} \in \{0, 1\}^*$ and outputting $k/\tilde{\varepsilon}$ where $k \in \mathbb{Z}$ is the unique integer such that $\text{REAL}(\mathbf{r}) \in \overline{I}_y(k)$. We have not yet defined what the precision $n = n(y, k)$ is; we will do this below, but first we give some intuition as to why $\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}$ will satisfy $(\tilde{\varepsilon}, O(1))$ -SV-consistent sampling for some “good-enough” precision.

SV-Consistent Sampling. Recall that since we assume $f \in \mathcal{F}_1$, for any two neighboring databases D_1, D_2 we can assume that $f(D_1) = y$ and $f(D_2) = y-1$, so that for any $k \in \mathbb{Z}$

$$\frac{\Pr[\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}(f, D_1) = k/\tilde{\varepsilon}]}{\Pr[\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}(f, D_2) = k/\tilde{\varepsilon}]} = \frac{\Pr[\overline{Z}_y = k/\tilde{\varepsilon}]}{\Pr[\overline{Z}_{y-1} = k/\tilde{\varepsilon}]} = \frac{|\overline{I}_y(k)|}{|\overline{I}_{y-1}(k)|}$$

We thus wish to prove that the mechanism has $(\tilde{\varepsilon}, c)$ -SV-consistent sampling where $T_1 = \overline{I}_y(k) \approx I_y(k)$ and $T_2 = \overline{I}_{y-1}(k) \approx I_{y-1}(k)$ in Definition 10. For now, let us assume that we use arithmetic coding with infinite precision, that is, we do not round the endpoints. We will give intuition as to why our mechanism satisfies an “infinite-precision analogue” of SV-consistent sampling. We can define \mathbf{u} to be the longest prefix of all coins in $I \stackrel{\text{def}}{=} I_y(k) \cup I_{y-1}(k)$, and let $\mathbf{u}_\ell \stackrel{\text{def}}{=} \mathbf{u}, 0, 0, \dots$ and $\mathbf{u}_r \stackrel{\text{def}}{=} \mathbf{u}, 1, 1, \dots$. Informally, \mathbf{u} is the longest prefix such that \mathbf{u}_ℓ is to the left of I and \mathbf{u}_r is to the right of I . Then an “infinite-precision analogue” of $(\cdot, O(1))$ -SV-consistent sampling is the following:

$$\frac{\text{REAL}(\mathbf{u}_r) - \text{REAL}(\mathbf{u}_\ell)}{|I_y(k) \cup I_{y-1}(k)|} = O(1) \quad (1)$$

By construction, we have $\text{REAL}(\mathbf{u}_r) - \text{REAL}(\mathbf{u}_\ell) \approx 2^{-|\mathbf{u}|}$. Furthermore, arithmetic coding ensures that $I_y(k) \cap I_{y-1}(k) \neq \emptyset$; indeed, we can view $I_{y-1}(k)$ as having “shifted” $I_y(k)$ slightly to the right. We can therefore view $I = I_y(k) \cup I_{y-1}(k)$ as one single interval that is slightly bigger. Moreover, arithmetic coding and our use of the Laplacian distribution ensures that smaller intervals are farther from the center than bigger ones, and in fact, the size of the interval that contains I and everything to its right (or left, depending on whether I is to the right or left of $1/2$, respectively) is a constant factor of $|I|$. This means that $|I_y(k) \cup I_{y-1}(k)| = |I| = c \cdot 2^{-|\mathbf{u}|}$ for a constant c , and we thus obtain the ratio required in Equation (1).

Defining the Precision. Now we just need to round all the points $s_y(k)$ with enough precision so that the rounding is “legal” (i.e., preserves the relative sizes

of all intervals $I_y(k)$ and $I_y(k) \setminus I_{y-1}(k)$ to within a constant factor), so that our informal analysis of SV-consistency above still holds after the rounding. Formally, we let $I'_y(k) \stackrel{\text{def}}{=} I_y(k) \setminus I_{y-1}(k)$, be the interval containing the coins that will make the mechanism output $k/\tilde{\varepsilon}$ when it is run on D_1 but output $(k-1)/\tilde{\varepsilon}$ when run on D_2 . We then let

$$n(y, k) = n(D, f, z) \stackrel{\text{def}}{=} \log\left(\frac{1}{|I'_y(k)|}\right) + 3$$

and round $s_y(k)$ to its $\max(n(y+1, k+1), n(y, k+1))$ most significant figures.

The resulting mechanism $\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}$ is shown in Figure 3.

We can now state our main results about SV-consistency, SV-privacy, and SV-utility of our mechanism:

Lemma 8. *Mechanism $\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}$ has $(27\tilde{\varepsilon}, 57)$ -SV-consistent sampling. In particular, $\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}$ is $(27\tilde{\varepsilon}, \mathcal{U})$ -differentially private and $(\varepsilon, \mathcal{SV}(\gamma))$ -differentially private for $\varepsilon = (216\tilde{\varepsilon})^{1-\log(1+\gamma)} \left(\frac{1+\gamma}{1-\gamma}\right)^9$. Mechanism $\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}$ has $(O(1/\tilde{\varepsilon}), \mathcal{U})$ -utility and $(\rho, \mathcal{SV}(\gamma))$ -utility, where $\rho = O\left(\frac{1}{\tilde{\varepsilon}} \cdot \frac{1}{1-\gamma}\right)$.*

$\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}(D, f ; \mathbf{r})$: Compute $y \stackrel{\text{def}}{=} f(D)$ and output a sample from the distribution $\mathbb{Z}_y \stackrel{\text{def}}{=} 1/\tilde{\varepsilon} \cdot [\tilde{\varepsilon} \cdot \text{Lap}_{y, 1/\tilde{\varepsilon}}]$ by using arithmetic coding as explained below.

- Let $n(y, k) = n(D, f, z) \stackrel{\text{def}}{=} \log\left(\frac{1}{|I'_y(k)|}\right) + 3$ and let $\mathbf{r}'_{y,k}$ be the $n(y, k)$ most significant figures of \mathbf{r} .
- Output the unique $z = k/\tilde{\varepsilon}$ such that $\frac{k-1/2}{\tilde{\varepsilon}} \leq \text{REAL}(\mathbf{r}'_{y,k}) < \frac{k+1/2}{\tilde{\varepsilon}}$.

Fig. 3. Finite precision mechanism $\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}$ that has $(27\tilde{\varepsilon}, 57)$ -SV-consistent sampling

Theorem 2. *For all $\gamma < 1$, $\overline{\mathcal{M}}^{\text{SVCS}} = \{\overline{M}_{\tilde{\varepsilon}}^{\text{SVCS}}\}$ is a class of non-trivial mechanisms for \mathcal{F}_1 w.r.t. $\mathcal{SV}(\gamma)$.*

References

1. Andreev, A.E., Clementi, A.E.F., Rolim, J.D.P., Trevisan, L.: Weak random sources, hitting sets, and BPP simulations. *SIAM J. Comput.* 28(6), 2103–2116 (1999)
2. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the sulq framework. In: Li, C. (ed.) *PODS*, pp. 128–138. ACM (2005)
3. Bosley, C., Dodis, Y.: Does Privacy Require True Randomness? In: Vadhan, S.P. (ed.) *TCC 2007. LNCS*, vol. 4392, pp. 1–20. Springer, Heidelberg (2007)
4. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the web. *Computer Networks* 29(8-13), 1157–1166 (1997)

5. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.* 17(2), 230–261 (1988)
6. Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: Neven, F., Beeri, C., Milo, T. (eds.) PODS, pp. 202–210. ACM (2003)
7. Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 232–250. Springer, Heidelberg (2006)
8. Dodis, Y., Ong, S.J., Prabhakaran, M., Sahai, A.: On the (im)possibility of cryptography with imperfect randomness. In: FOCS, pp. 196–205. IEEE Computer Society (2004)
9. Dodis, Y., Spencer, J.: On the (non)universality of the one-time pad. In: FOCS, pp. 376–385. IEEE Computer Society (2002)
10. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
11. Dwork, C., Nissim, K.: Privacy-Preserving Datamining on Vertically Partitioned Databases. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 528–544. Springer, Heidelberg (2004)
12. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: Mitzenmacher, M. (ed.) STOC, pp. 351–360. ACM (2009)
13. Hardt, M., Talwar, K.: On the geometry of differential privacy. In: Schulman, L.J. (ed.) STOC, pp. 705–714. ACM (2010)
14. Holenstein, T.: Parallel repetition: simplifications and the no-signaling case. In: Johnson, D.S., Feige, U. (eds.) STOC, pp. 411–419. ACM (2007)
15. Manber, U.: Finding similar files in a large file system. In: Proceedings of the USENIX Winter 1994 Technical Conference, p. 2. USENIX Association, Berkeley (1994)
16. Maurer, U.M., Wolf, S.: Privacy Amplification Secure against Active Adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 307–321. Springer, Heidelberg (1997)
17. McGregor, A., Mironov, I., Pitassi, T., Reingold, O., Talwar, K., Vadhan, S.P.: The limits of two-party differential privacy. In: FOCS, pp. 81–90. IEEE Computer Society (2010)
18. McInnes, J.L., Pinkas, B.: On the Impossibility of Private Key Cryptography with Weakly Random Keys. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 421–435. Springer, Heidelberg (1991)
19. Moffat, A., Neal, R.M., Witten, I.H.: Arithmetic coding revisited. *ACM Trans. Inf. Syst.* 16(3), 256–294 (1998)
20. Reingold, O., Vadhan, S., Widgerson, A.: No deterministic extraction from santha-vazirani sources – a simple proof (2004),
<http://windowsonttheory.org/2012/02/21/no-deterministic-extraction-from-santha-vazirani-sources-a-simple-proof/>
21. Santha, M., Vazirani, U.V.: Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.* 33(1), 75–87 (1986)
22. Vazirani, U.V., Vazirani, V.V.: Random polynomial time is equal to slightly-random polynomial time. In: FOCS, pp. 417–428. IEEE Computer Society (1985)
23. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. *Commun. ACM* 30(6), 520–540 (1987)
24. Zuckerman, D.: Simulating BPP using a general weak random source. *Algorithmica* 16(4/5), 367–391 (1996)

Tamper and Leakage Resilience in the Split-State Model

Feng-Hao Liu and Anna Lysyanskaya

Brown University
`{fenghao,anna}@cs.brown.edu`

Abstract. It is notoriously difficult to create hardware that is immune from side channel and tampering attacks. A lot of recent literature, therefore, has instead considered *algorithmic* defenses from such attacks. In this paper, we show how to algorithmically secure any cryptographic functionality from continual split-state leakage and tampering attacks. A split-state attack on cryptographic hardware is one that targets separate parts of the hardware separately. Our construction does not require the hardware to have access to randomness. In contrast, prior work on protecting from continual combined leakage and tampering [23] required true randomness for each update. Our construction is in the common reference string (CRS) model; the CRS must be hard-wired into the device. We note that prior negative results show that it is impossible to algorithmically secure a cryptographic functionality against a combination of arbitrary continual leakage and tampering attacks without true randomness; therefore restricting our attention to the split-state model is justified. Our construction is simple and modular, and relies on a new construction, in the CRS model, of non-malleable codes with respect to split-state tampering functions, which may be of independent interest.

1 Introduction

Recently, the cryptographic community has been extensively studying various flavors of the following general problem. Suppose that we have a device that implements some cryptographic functionality (for example, a signature scheme or a cryptosystem). Further, suppose that an adversary can, in addition to input/output access to the device, get some side-channel information about its secret state, potentially on a continual basis; for example, an adversary can measure the power consumption of the device, timing of operations, or even read part of the secret directly [25,18]. Additionally, suppose that the adversary can, also possibly on a continual basis, somehow alter the secret state of the device through an additional physical attack such as microwaving the device or exposing to heat or EM radiation [4,1]. What can be done about protecting the security of the functionality of the device?

Unfortunately, strong negative results exist even for highly restricted versions of this general problem. For example, if the device does not have access to randomness, but is subject to arbitrary continual leakage, and so, in each round i , can leak to the adversary just one bit $b_i(s_i)$ for a predicate b_i of the adversary's

choice, eventually it will leak its entire secret state. Moreover, even in a very restricted leakage model where the adversary can continually learn a physical bit of the secret state s_i , if the adversary is also allowed to tamper with the device and the device does not have access to randomness, Liu and Lysyanskaya [28] showed that the adversary will eventually learn the entire secret state. Further, even with tampering alone, Gennaro et al. [16] show that security from arbitrary tampering cannot be achieved unless the device can overwrite its memory; further, they show that security can only be achieved in the common reference string model.

For the leakage-only case, positive results are known for continual attacks assuming an on-device source or randomness [5,8,27,26]. The one-time leakage case has also been studied [2,30,3,24]. For the tampering-only case, positive results are known as well for different setup and tampering models [16,13,6].

Finally, there are positive results for signature and encryption devices when both continual tampering and leakage are possible, and the device has access to a protected source of true randomness [23]. One may be tempted to infer from this positive result that it can be “derandomized” by replacing true randomness with the continuous output of a pseudorandom generator, but this approach is ruled out by Liu and Lysyanskaya [28]. Yet, how does a device, while under a physical attack, access true randomness? True randomness is a scarce resource even when a device is not under attack; for example, the GPG implementations of public-key cryptography ask the user to supply random keystrokes whenever true randomness is needed, which leads to non-random bits should a device fall into the adversary’s hands.

In this paper, we investigate general techniques for protecting cryptographic devices from continual leakage and tampering attacks without requiring access to true randomness after initialization. Since, as we explained above, this is impossible for general classes of leakage and tampering functions, we can only solve this problem for restricted classes of leakage and tampering functions. Which restrictions are reasonable? Suppose that a device is designed such that its memory M is split into two compartments, M_1 and M_2 , that are physically separated. For example, a laptop may have more than one hard drive. Then it is reasonable to imagine that the adversary’s side channel that leaks information about M_1 does not have access to M_2 , and vice versa. Similarly, the adversary’s tampering function tampers with M_1 without access to M_2 , and vice versa. This is known as the split-state model, and it has been considered before in the context of leakage-only [12,9] and tampering-only [13] attacks.

Our main result. Let $G(\cdot, \cdot)$ be any deterministic cryptographic functionality that, on input some secret state s and user-provided input x , outputs to the user the value y , and possibly updates its secret state to a new value s' ; formally, $(y, s') = G(s, x)$. For example, G can be a stateful pseudorandom generator that, on input an integer m and a seed s , generates $m + |s|$ pseudorandom bits, and lets y be the first m of these bits, and updates its state to be the next $|s|$ bits. A signature scheme and a decryption functionality can also be modeled this way. A participant in an interactive protocol, such as a zero-knowledge proof, or an MPC protocol, can also be modeled as a stateful cryptographic functionality;

the initial state s would represent its input and random tape; while the supplied input x would represent a message received by this participant. A construction that secures such a general stateful functionality G against tampering and leakage is therefore the most general possible result. This is what we achieve: our construction works for any efficient deterministic cryptographic functionality G and secures it against tampering and leakage attacks in the split-state model, without access to any randomness after initialization, but with access to a trusted common reference string (CRS). Any randomized functionality G can be securely derandomized using a pseudorandom generator whose seed is chosen in the initialization phase; our construction also applies to such a derandomized version of G . Quantitatively, our construction tolerates continual leakage of as many as $(1 - o(1))n$ bits of the secret memory, where n is the size of the secret memory.

Our construction assumes the existence of a one-time leakage resilient public-key cryptosystem that allows leakage of any poly-time computable $g(\text{sk})$ of length $c|\text{sk}|$ for some constant c , for example one due to Naor and Segev [30]. Further, we need robust non-interactive zero-knowledge proof systems [7] for an appropriate NP language. See the full version of this paper for further detailed discussions.

Prior work. Here we give a table summarizing the state of the art in tolerating continual leakage and tampering attacks; specific attacks we consider are split-state attacks (abbreviated as “SS”), attacks on physical bits (abbreviated as “bits”), attacks on small blocks (abbreviated as “blocks”), and attacks by any polynomial-sized circuits (abbreviated as “any”).

Type of leakage	Type of tampering	Local coins	Known results about continual attacks
None	Any	No	Signature and decryption in the CRS model [16]
Any	None	No	Trivially impossible
Bits	Any	No	Impossible [28]
Any	None	Yes	Signature and encryption in the plain model [5,8,27,26]
None	Bits	Yes	All functionalities in the plain model [13]
None	SS	Yes	All functionalities in the RO model [13]
None	Blocks	Yes	All functionalities in the plain model [6]
Any	Any	Yes	Signature and encryption in the CRS model [23]
SS	SS	No	All functionalities in the CRS model [This work]

We remark that all the results referenced above apply to attacks on the memory of the device, rather than its computation (with one exception). The exception [26] is the work that constructed the first encryption and signature schemes that can leak more than logarithmic number of bits during their update procedure (but cannot be tampered with). Thus, all these works assume computation to be somewhat secure. In this work, for simplicity, we also assume that computation is secure, and remark that there is a line of work on protecting computation from leakage or tampering [21,29,20,12,31,10,15,17,22,14]. This is orthogonal to the study of protecting memory leakage and tampering.

In particular, we can combine our work with that of Goldwasser and Rothblum [17], or Juma and Vahlis [22] to obtain a construction where computation is protected as well; however, this comes at a cost of needing fresh local randomness. All known cryptographic constructions that allow an adversary to issue leakage queries while the computation is going on rely on fresh local randomness.

A decryption device produced by our compiler will have stronger leakage resilience properties than most previous work [5,27,26,23] on leakage resilient encryption: it will tolerate *after-the-fact* leakage defined by Halevi and Lin [19]; since this will be guaranteed on a continual basis, our results solve a problem left explicitly open by Halevi and Lin.

Our building block: non-malleable codes. We use non-malleable codes, defined by Dziembowski et al. [13], as our building block.

Let $\mathcal{E}nc$ be an encoding procedure and $\mathcal{D}ec$ be the corresponding decoding procedure. Consider the following tampering experiment [13]: (1) A string s is encoded yielding a codeword $c = \mathcal{E}nc(s)$. (2) The codeword c is mangled by some function f to some $c^* = f(c)$. (3) The resulting codeword is decoded, resulting in $s^* = \mathcal{D}ec(c^*)$. $(\mathcal{E}nc, \mathcal{D}ec)$ constitutes a non-malleable code if tampering with c can produce only two possible outcomes: (1) f leaves c unchanged; (2) the decoded string s^* is unrelated to the original string s . Intuitively, this means that one cannot learn anything about the original string s by tampering with the codeword c .

It is clear [13] that, without any restrictions on f , this notion of security is unattainable. For example, f could, on input c , decode it to s , and then compute $s^* = s + 1$ and then output $\mathcal{E}nc(s^*)$. Such an f demonstrates that no $(\mathcal{E}nc, \mathcal{D}ec)$ can satisfy this definition. However, for restricted classes of functions, this definition can be instantiated.

Dziembowski et al. constructed non-malleable codes with respect to bit-wise tampering functions in the plain model, and with respect to split-state tampering functions in the random oracle model. They also show a compiler that uses non-malleable codes to secure any functionality against tampering attacks. In this paper, we improve their result in four ways: first, we construct a non-malleable code with respect to split-state tampering, in the CRS model (which is a significant improvement over the RO model). Second, our code has an additional property: it is leakage resilient. Third, we prove that plugging in a leakage-resilient non-malleable code in the Dziembowski et al. compiler secures any functionality against *both* tampering and leakage attacks. Fourth, we *de-randomize* the compiled construction such that it no longer requires a trusted source of randomness for updates.

Our continual tampering and leakage model. We consider the same tampering and leakage attacks as those of Liu and Lysyanskaya[28] and Kalai et al. [23], which generalized the model of tampering-only [16,13] and leakage-only [5,8,27,26] attacks. (However, in this attack model we achieve stronger security, as discussed above.)

Let M be the memory of the device under attack. We view time as divided into discrete time periods, or rounds. In each round, the adversary A makes a leakage query g or a tampering query f ; as a result, A obtains $g(M)$ or modifies the memory: $M := f(M)$. In this work, we consider both g, f to be split-state functions. We consider a simulation-based definition of security against such attacks.

Our approach. Let $G(s, x)$ be the functionality we want to secure, where s is some secret state and x is the user input. Our compiler takes the leakage-resilient non-malleable code and G as input, outputs $G'(\mathcal{E}nc(s), x)$, where G' gets an encoded version of the state s , emulates $G(s, x)$ and re-encodes the new state at the end of each round. Then we will argue that even if the adversary can get partial information or tamper with the encoded state in every round, the compiled construction is still secure.

2 Our Model

Definition 1. Define the following three function classes $\mathcal{G}_t, \mathcal{F}^{\text{half}}, \mathcal{G}_{t_1, t_2}^{\text{half}}$:

- Let $t \in \mathbb{N}$. By \mathcal{G}_t we denote the set of poly-sized circuits with output length t .
- Let $\mathcal{F}^{\text{half}}$ denote the set of functions of the following form: $f : \{0, 1\}^{2m} \rightarrow \{0, 1\}^{2m} \in \mathcal{F}^{\text{half}}$ if there exist two poly-sized circuits $f_1, f_2 : \{0, 1\}^m \rightarrow \{0, 1\}^m$, such that for all $x, y \in \{0, 1\}^m$, $f(x, y) = f_1(x) \circ f_2(y)$.
- Let $t_1, t_2 \in \mathbb{N}$, and $\mathcal{G}_{t_1, t_2}^{\text{half}}$ be the set of all poly-sized leakage functions that leak independently on each half of their inputs, t_1 bits on the first half and t_2 bits on the second half.

We further denote $\mathcal{G}_{t_1, \text{all}}^{\text{half}}$ as the case where $g_1(x)$ leaks t_1 bits, and $g_2(y)$ can leak all its input y .

Next, let us define an adversary's access to a functionality under tampering and leakage attacks. In addition to queries to the functionality itself (called `Execute` queries) an attacker has two more operations: he can cause the memory of the device to get tampered according to some function f , or he can learn some function g of the memory. Formally:

Definition 2 (Interactive Functionality Subject to Tampering and Leakage Attacks). Let $\langle G, s \rangle$ be an interactive stateful system consisting of a public (perhaps randomized) functionality $G : \{0, 1\}^u \times \{0, 1\}^k \rightarrow \{0, 1\}^v \times \{0, 1\}^k$ and a secret initial state $s \in \{0, 1\}^k$. We consider the following ways of interacting with the system:

- `Execute`(x): For $x \in \{0, 1\}^u$, the system will compute $(y, s_{\text{new}}) \leftarrow G(s, x)$, privately update state to s_{new} , and output y .
- `Tamper`(f): the state s is replaced by $f(s)$.
- `Leak`(g): the adversary can obtain the information $g(s)$.

Next, we define a compiler that compiles a functionality $\langle G, s \rangle$ into a hardware implementation $\langle G', s' \rangle$ that can withstand leakage and tampering attacks. A compiler will consist of two algorithms, one for compiling the circuit for G into another circuit, G' ; the other algorithm is for compiling the memory, s , into s' . This compiler will be *correct*, that is to say, the resulting circuit and memory will provide input/output functionality identical to the original circuit; it will also be tamper- and leakage-resilient in the following strong sense: there exists a simulator that, with oracle access to the original $\langle G, s \rangle$, will simulate the behavior of $\langle G', s' \rangle$ under tampering and leakage attacks. The following definitions formalize this:

Definition 3. Let CRS be an algorithm that generates a common reference string, on input the security parameter 1^k . The algorithms $(\text{CircuitCompile}, \text{MemCompile})$ constitute a correct and efficiency-preserving compiler in the $\text{CRS}(1^k)$ model if for all $\Sigma \in \text{CRS}(1^k)$, for any Execute query x , $\langle G', s' \rangle$'s answer is distributed identically to $\langle G, s \rangle$'s answer, where $G' = \text{CircuitCompile}(\Sigma, G)$ and $s' \in \text{MemCompile}(\Sigma, s)$; moreover, CircuitCompile and MemCompile run in polynomial time and output G' and s' of size polynomial in the original circuit G and secret s .

Note that this definition of the compiler ensures that the compiled functionality G' inherits all the security properties of the original functionality G . Also the compiler defined here works separately on the functionality G and on the secret s , which means that it can be combined with another compiler that strengthens G' in some other way (for example, it can be combined with the compiler of Goldwasser and Rothblum [17]). This definition allows for both randomized and deterministic G' ; as we discussed in the introduction, in general a deterministic circuit is more desirable.

Remark 1. Recall that G , and therefore G' , are modeled as stateful functionalities. By convention, running $\text{Execute}(\varepsilon)$ will cause them to update their states.

As defined above, in the face of the adversary's Execute queries, the compiled G' behaves identically to the original G . Next, we want to formalize the important property that whatever the adversary can learn from the compiled functionality G' using Execute , Tamper and Leak queries, can be learned just from the Execute queries of the original functionality G .

We want the real experiment where the adversary interacts with the compiled functionality $\langle G', s' \rangle$ and issues Execute , Tamper and Leak queries, to be indistinguishable from an experiment in which a simulator $\mathcal{S}\text{im}$ only has black-box access to the original functionality G with the secret state s (i.e. $\langle G, s \rangle$). More precisely, in every round, $\mathcal{S}\text{im}$ will get some tampering function f or leakage function g from A and then respond to them. In the end, the adversary halts and outputs its view. The simulator then may (potentially) output this view. Whatever view $\mathcal{S}\text{im}$ outputs needs to be indistinguishable from the view A obtained in the real experiment. This captures the fact that the adversary's tampering and leakage attacks in the real experiment can be simulated by only accessing the functionality in a black-box way. Thus, these additional physical attacks do not give the adversary any additional power.

Definition 4 (Security Against \mathcal{F} Tampering and \mathcal{G} Leakage). A compiler (CircuitCompile , MemCompile) yields an \mathcal{F} - \mathcal{G} resilient hardened functionality in the CRS model if there exists a simulator Sim such that for every efficient functionality $G \in \text{PPT}$ with k -bit state, and non-uniform PPT adversary A , and any state $s \in \{0, 1\}^k$, the output of the following real experiment is indistinguishable from that of the following ideal experiment:

Real Experiment $\text{Real}(A, s)$: Let $\Sigma \leftarrow \text{CRS}(1^k)$ be a common reference string given to all parties. Let $G' \leftarrow \text{CircuitCompile}(\Sigma, G)$, $s' \leftarrow \text{MemCompile}(\Sigma, s)$. The adversary $A(\Sigma)$ interacts with the compiled functionality $\langle G', s' \rangle$ for arbitrarily many rounds where in each round:

- A runs $\text{Execute}(x)$ for some $x \in \{0, 1\}^u$, and receives the output y .
- A runs $\text{Tamper}(f)$ for some $f \in \mathcal{F}$, and then the encoded state is replaced with $f(s')$.
- A runs $\text{Leak}(g)$, and receives some $\ell = g(s')$ for some $g \in \mathcal{G}$, where s' is the current state. Then the system updates its memory by running $\text{Execute}(\varepsilon)$, which will update the memory with a re-encoded version of the current state.

Let $\text{view}_A = (\text{state}_A, x_1, y_1, \ell_1, x_2, y_2, \ell_2, \dots)$ denote the adversary's view where x_i 's are the execute input queries, y_i 's are their corresponding outputs, ℓ_i 's are the leakage at each round i . In the end, the experiment outputs (Σ, view_A) .

Ideal Experiment $\text{Ideal}(\text{Sim}, A, s)$: Sim first sets up a common reference string Σ , and $\text{Sim}^{A(\Sigma), \langle G, s \rangle}$ outputs $(\Sigma, \text{view}_{\text{Sim}}) = (\Sigma, (\text{state}_{\text{Sim}}, x_1, y_1, \ell_1, x_2, y_2, \ell_2, \dots))$, where (x_i, y_i, ℓ_i) is the input/output/leakage tuple simulated by Sim with oracle access to $A, \langle G, s \rangle$.

Note that we require that, in the real experiment, after each leakage query the device updates its memory. This is necessary, because otherwise the adversary could just keep issuing Leak query on the same memory content and, over time, could learn the memory bit by bit.

Also, note that, following Dziembowski et al. [13] we require that each experiment faithfully record all the Execute queries. This is a way to capture the idea that the simulator cannot make more queries than the adversary; as a result, an adversary in the real experiment (where he can tamper with the secret and get side information about it) learns the same amount about the secret as the simulator who makes the same queries (but does NOT get the additional tampering and leakage ability) in the ideal experiment.

3 Leakage Resilient Non-malleable Codes

In this section, we present the definition of leakage resilient non-malleable codes (LR-NM codes), and our construction. We also extend the definition of Dziembowski et al. [13] in two directions: we define a coding scheme in the CRS model, and we consider leakage resilience of a scheme. Also, our construction achieves the stronger version of non-malleability, so we present this version. For the normal non-malleability and the comparison, we refer curious readers to the paper [13].

Definition 5 (Coding Scheme in the Common Reference String Model). Let k be the security parameter, and $\text{Init}(1^k)$ be an efficient randomized algorithm that outputs a common reference string (CRS) $\Sigma \in \{0, 1\}^{\text{poly}(k)}$. We say $\mathcal{C} = (\text{Init}, \mathcal{E}nc, \mathcal{D}ec)$ is a coding scheme in the CRS model if for every k , $(\mathcal{E}nc(1^k, \Sigma, \cdot), \mathcal{D}ec(1^k, \Sigma, \cdot))$ is a $(k, n(k))$ coding scheme for some polynomial $n(k)$: i.e. for each $s \in \{0, 1\}^k$, $\Pr[\mathcal{D}ec(\Sigma, \mathcal{E}nc(\Sigma, s)) = s] = 1$. For simplicity, we will omit the security parameter.

Now we define the two properties of coding schemes: non-malleability and leakage resilience. We extend the definition of the strong non-malleability by Dziembowski et al. [13] to the CRS model.

Definition 6 (Strong Non-malleability in the CRS Model). Let \mathcal{F} be some family of functions. For each function $f \in \mathcal{F}$, and $s \in \{0, 1\}^k$, define the tampering experiment in the common reference string model. For any CRS Σ , we define

$$\text{Tamper}_s^{f, \Sigma} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} c \leftarrow \mathcal{E}nc(\Sigma, s), \tilde{c} = f^\Sigma(c), \tilde{s} = \mathcal{D}ec(\Sigma, \tilde{c}) \\ \text{Output : same* if } \tilde{c} = c, \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\},$$

where the randomness of this experiment comes from the randomness of the encoding and decoding algorithms.

We say the coding scheme $(\text{Init}, \mathcal{E}nc, \mathcal{D}ec)$ is strong non-malleable if we have $\{(\Sigma, \text{Tamper}_{s_0}^{f, \Sigma})\}_{k \in \mathbb{N}} \approx \{(\Sigma, \text{Tamper}_{s_1}^{f, \Sigma})\}_{k \in \mathbb{N}}$ where $\Sigma \leftarrow \text{Init}(1^k)$, any $s_0, s_1 \in \{0, 1\}^k$, and $f \in \mathcal{F}$, and \approx can refer to statistical or computational indistinguishability.

Definition 7 (Leakage Resilience). Let \mathcal{G} be some family of functions. A coding scheme $(\text{Init}, \mathcal{E}nc, \mathcal{D}ec)$ is leakage resilient with respect to \mathcal{G} if for every function $g \in \mathcal{G}$, every two states $s_0, s_1 \in \{0, 1\}^k$, and every efficient adversary A , we have $\Pr[A(\Sigma, g(\Sigma, \mathcal{E}nc(\Sigma, s_b)) = b] \leq 1/2 + \text{ngl}(k)$, where b is a random bit, and $\Sigma \leftarrow \text{Init}(1^k)$.

How do we realize this definition? Consider a technique reminiscent of non-malleable encryption [11,32]: set $M_1 = \text{sk}$, $M_2 = (\text{pk}, \hat{s} = \text{Encrypt}_{\text{pk}}(s), \pi)$ where π is a proof of consistency (i.e. it proves that there exists a secret key corresponding to pk and that \hat{s} can be decrypted using this secret key). Does this work? If the underlying proof system is malleable, then it could be possible to modify both parts at the same time, so that the attacker could obtain an encoding of a string that is related to the original s . So we require that the proof system be *non-malleable*; specifically we use the notion of *robust NIZK* given by de Santis et al. [7], in which, informally, the adversary can only output new proofs for which he knows the corresponding witnesses, even when given black-box access to a simulator that produces simulated proofs on demand; there exists an extractor that can extract these witnesses.

Now let us try to give a high-level proof of security. Given a public key pk , and a ciphertext c , it is the reduction's job to determine whether c is an encryption of s_0 or s_1 , with the help of the adversary that distinguishes $\text{Tamper}_{s_0}^f$ and $\text{Tamper}_{s_1}^f$. A natural way for the reduction is to pretend that $M_1 = \text{sk}$, and put

the public key \mathbf{pk} and the ciphertext $\hat{s} = c$ with a simulated proof into M_2 , setting $M_2 = (\mathbf{pk}, \hat{s}, \pi_{Sim})$. Then the reduction simulates Tamper_s^f . Clearly, irrespective of f_1 the reduction can compute $f_2(M_2) = (\mathbf{pk}', \hat{s}', \pi_{Sim})$, and intuitively, the non-malleability of the proof assures that the adversary can only generate valid (\mathbf{pk}', \hat{s}') if he knows \mathbf{sk}' and s' . So at first glance, the outcome of the tampering experiment (i.e. the decoding of the tampered codeword) should be s' , which can be simulated by the reduction. Thus, the reduction can use A to distinguish the two different experiments.

However, there are several subtle missing links in the above argument. The reduction above does not use any property of f_1 , which might cause a problem. Suppose $f_1(\mathbf{sk}) = \mathbf{sk}'$, then the decoding of the tampered codeword is really s' , so the reduction above simulates the tampering experiment faithfully. However, if not, then the decoding should be \perp instead. Thus, the reduction crucially needs one bit of information: $\mathbf{sk}' \stackrel{?}{=} f_1(\mathbf{sk})$. If the reduction could get leakage $f_1(\mathbf{sk})$ directly, then it could compute this bit. However, the length of $f_1(\mathbf{sk})$ is the same as that of \mathbf{sk} itself, and therefore no leakage-resilient cryptosystem can tolerate this much leakage.

Our novel observation here is that actually a small amount of leaked information about the secret key \mathbf{sk} is sufficient for the reduction to tell the two cases apart. Let h be a hash function that maps input strings to strings of length ℓ . Then, to check whether $f_1(\mathbf{sk}) = \mathbf{sk}'$, it is very likely (assuming proper collision-resistance properties of h) sufficient to check if $h(f_1(\mathbf{sk})) = h(\mathbf{sk}')$. So if given a cryptosystem that can tolerate ℓ bits of leakage, we can build a reduction that asks that $h(f_1(\mathbf{sk}))$ be leaked, and this (in addition to a few other technicalities that we do not highlight here) enables us to show that the above construction is non-malleable.

Besides non-malleability, the above code is also leakage-resilient in the sense that getting partial information about a codeword does not reveal any information about the encoded string. Intuitively, this is because the NIZK proof hides the witness, i.e. the message, and partial leakage of the secret key does not reveal anything about the message, either. Thus, this construction achieves non-malleability and leakage resilience at the same time.

The Construction. Let t be a polynomial, $\mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ be an encryption scheme that is semantically secure against one-time leakage \mathcal{G}_t , and $\Pi = (\ell, \mathcal{P}, \mathcal{V}, \mathcal{S})$ be a robust NIZK proof system (we defer the formal definitions to the full version of this paper). The encryption scheme and robust NIZK need to have some additional properties, and we briefly summarize them here: (1) given a secret key \mathbf{sk} , one can efficiently derive its corresponding public key \mathbf{pk} ; (2) given a key pair $(\mathbf{pk}, \mathbf{sk})$, it is infeasible to find another valid $(\mathbf{pk}, \mathbf{sk}')$ where $\mathbf{sk} \neq \mathbf{sk}'$; (3) different statements of the proof system must have different proofs. In the full version of this paper, we give formal definitions of these additional properties and show that simple modifications of leakage-resilient crypto systems and robust NIZK proof systems satisfy them. We define a coding scheme $(\mathcal{I}nit, \mathcal{E}nc, \mathcal{D}ec)$ in Figure 1.

The coding scheme:

- $\mathcal{I}nit(1^k)$: sample a CRS: $\Sigma \leftarrow \{0, 1\}^{\ell(k)}$.
- $\mathcal{E}nc(\Sigma, s)$: on input $s \in \{0, 1\}^k$, sample $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KeyGen}(1^k)$. Let \mathbf{L} be the following language, and \mathbf{W} be its natural witness relation:

$$\mathbf{L} \stackrel{\text{def}}{=} \left\{ (\mathbf{pk}, \hat{m}) : \exists w = (\mathbf{sk}, m) \text{ s.t. } \begin{array}{l} (\mathbf{pk}, \mathbf{sk}) \text{ forms a key pair for } \mathcal{E} \text{ and} \\ m = \text{Decrypt}_{\mathbf{sk}}(\hat{m}) \end{array} \right\}$$

Compute $\pi \leftarrow \mathcal{P}((\mathbf{pk}, \hat{s}), (\mathbf{sk}, s, r), \Sigma)$, an NIZK proof of the statement that $(\mathbf{pk}, \hat{s}) \in \mathbf{L}$. Output the encoding $c = (\mathbf{sk}; \mathbf{pk}, \hat{s} = \text{Encrypt}_{\mathbf{pk}}(s), \pi)$.

- $\mathcal{D}ec(\Sigma, c)$: If (1) $\mathcal{V}((\mathbf{pk}, \hat{s}), \pi, \Sigma)$ accepts and (2) $(\mathbf{pk}, \mathbf{sk})$ form a valid key pair, output $\text{Decrypt}_{\mathbf{sk}}(\hat{s})$. Otherwise, output \perp .

Fig. 1. The coding scheme

Let $n = n(k)$ be the polynomial that is equal to the length of $\mathbf{sk} \circ \mathbf{pk} \circ \hat{s} \circ \pi$. Without loss of generality, we assume that n is even, and $|\mathbf{sk}| = n/2$, and $|\mathbf{pk} \circ \hat{s} \circ \pi| = n/2$ (these properties can be easily guaranteed by padding the shorter side with 0's). Thus, a split-state device where $n(k)$ -bit memory M is partitioned into M_1 and M_2 could store \mathbf{sk} in M_1 and $(\mathbf{pk}, \hat{s}, \pi)$ in M_2 .

Remark 2. Note that the decoding algorithm $\mathcal{D}ec$ is deterministic if the verifier \mathcal{V} and the decryption algorithm Decrypt are both deterministic; as almost all known instantiations are. In the rest of the paper, we will assume that the decoding algorithm is deterministic.

Then we are able to achieve the following theorem:

Theorem 1. *Let $t : \mathbb{N} \rightarrow \mathbb{N}$ be some non-decreasing polynomial, and \mathcal{G}_t , $\mathcal{F}^{\text{half}}$, $\mathcal{G}_{t,\text{all}}^{\text{half}}$ be as defined above. Suppose the encryption scheme \mathcal{E} is semantically secure against one-time leakage \mathcal{G}_t ; the system Π is a robust NIZK as stated above; and $\mathcal{H}_k : \{h_z : \{0, 1\}^{\text{poly}(k)} \rightarrow \{0, 1\}^k\}_{z \in \{0, 1\}^k}$ is a family of universal one-way hash functions.*

Then the coding scheme is strong non-malleable (Def 6) with respect to $\mathcal{F}^{\text{half}}$, and leakage resilient (Def 7) with respect to $\mathcal{G}_{t,\text{all}}^{\text{half}}$.

Proof (Sketch). The proof contains two parts: showing that the code is non-malleable and that it is leakage resilient. The second part is easy so we only give the intuition. First let us look at $M_2 = (\mathbf{pk}, \hat{s}, \pi)$. Since π is a NIZK proof, it reveals no information about the witness (\mathbf{sk}, s) . For the memory $M_1 = \mathbf{sk}$, since the encryption scheme is leakage resilient, getting partial information about \mathbf{sk} does not hurt the semantic security. Thus, for any $g \in \mathcal{G}_{t,\text{all}}^{\text{half}}$, $g(M_1, M_2)$ hides the original input string. We omit the formal details of the reduction, since they are straightforward.

Now we focus on the proof of non-malleability. In particular, we need to argue that for any $s_0, s_1 \in \{0, 1\}^k$, and $f \in \mathcal{F}^{\text{half}}$, we have $(\Sigma, \text{Tamper}_{s_0}^{f, \Sigma}) \approx_c (\Sigma, \text{Tamper}_{s_1}^{f, \Sigma})$ where $\Sigma \leftarrow \mathcal{I}nit(1^k)$. We show this by contradiction: suppose

there exist $f = (f_1, f_2) \in \mathcal{F}^{\text{half}}$, s_0, s_1 , some $\varepsilon = 1/\text{poly}(k)$, and a distinguisher D such that $\Pr[D(\Sigma, \text{Tamper}_{s_0}^{f, \Sigma}) = 1] - \Pr[D(\Sigma, \text{Tamper}_{s_1}^{f, \Sigma}) = 1] > \varepsilon$, then we are going to construct a reduction that breaks the encryption scheme \mathcal{E} .

The reduction will work as discussed in the overview. Before describing it, we first make an observation: D still distinguishes the two cases of the Tamper experiments even if we change all the real proofs to the simulated ones. More formally, let $(\Sigma, \tau) \leftarrow \mathcal{S}_1(1^k)$, and define $\text{Tamper}_s^{f, \Sigma, \tau}$ be the same game as $\text{Tamper}_s^{f, \Sigma}$ except proofs in the encoding algorithm $\mathcal{E}nc(\Sigma, \cdot)$ are computed by the simulator $\mathcal{S}_2(\cdot, \Sigma, \tau)$ instead of the real prover. We denote this distribution as Tamper_s^{f*} . We claim that D also distinguishes $\text{Tamper}_{s_0}^{f*}$ from $\text{Tamper}_{s_1}^{f*}$.

Suppose not, i.e. D , who distinguishes $\text{Tamper}_{s_0}^{f, \Sigma}$ from $\text{Tamper}_{s_1}^{f, \Sigma}$ does not distinguish $\text{Tamper}_{s_0}^{f*}$ from $\text{Tamper}_{s_1}^{f*}$. Then one can use D, f, s_0, s_1 to distinguish real proofs and simulated ones using standard proof techniques. This violates the multi-theorem zero-knowledge property of the NIZK system Π . Thus, we have:

$$\Pr[D(\Sigma, \text{Tamper}_{s_0}^{f*}) = 1] - \Pr[D(\Sigma, \text{Tamper}_{s_1}^{f*}) = 1] > \varepsilon/2.$$

In the following, we are going to define a reduction Red to break the leakage resilient encryption scheme \mathcal{E} . The reduction Red consists of an adversary $A = (A_1, A_2, A_3)$ and a distinguisher D' defined below.

The reduction (with the part A) plays the leakage-resilience game $\text{LE}_b(\mathcal{E}, A, k, \mathcal{F})$ with the challenger, and with the help of the distinguisher D and the tampering function $f = (f_1, f_2)$. Informally speaking of the game, the adversary first sends a leakage function in $g \in \mathcal{F}$ (using A_1), and the challenger replies $g(\text{sk})$. Then A_2 chooses two messages m_0, m_1 , and the challenger encrypts either of them, and sends a challenge ciphertext. Finally, A_3 determines which message the challenge was generated from. We defer the formal definition of the game to the full version of this paper. Now we describe the reduction:

- First A_1 samples $z \in \{0, 1\}^{t-1}$ (this means A_1 samples a universal one-way hash function $h_z \leftarrow \mathcal{H}_{t-1}$), and sets up a simulated CRS with a corresponding trapdoor $(\Sigma, \tau) \leftarrow \mathcal{S}(1^k)$.
 - A_1 sets $g : \{0, 1\}^{n/2} \rightarrow \{0, 1\}^t$ to be the following function, and sends this leakage query to the challenger: $g(\text{sk}) = \begin{cases} 0^t & \text{if } f_1(\text{sk}) = \text{sk}, \\ 1 \circ h_z(f_1(\text{sk})) & \text{otherwise.} \end{cases}$
- This leakage value tells A_1 if the tampering function f_1 alters sk .
- A_2 chooses m_0, m_1 to be s_0, s_1 respectively. Then the challenger samples (pk, sk) and sets $\hat{m} = \text{Encrypt}_{\text{pk}}(m_b)$ to be the ciphertext, and sends $\text{pk}, g(\text{sk}), \hat{m}$ to the adversary.
 - Then A_3 computes the simulated proof $\pi = \mathcal{S}_2(\text{pk}, \hat{m}, \Sigma, \tau)$, and sets $(\text{pk}', \hat{m}', \pi') = f_2(\text{pk}, \hat{m}, \pi)$. Then A_3 computes a bit b using one of the algorithms in figure 2, depending on the outcome of $g(\text{sk})$.
 - Finally, A_3 outputs d , which is the output of the game $\text{LE}_b(\mathcal{E}, A, k, \mathcal{F}^{\text{half}})$.

Define the distinguisher D' on input d outputs $D(\Sigma, d)$. Then we need to show that A, D' break the scheme \mathcal{E} by the following lemma. In particular, we will show that the above A 's strategy simulates the distributions $\text{Tamper}_{s_b}^{f*}$, so that the distinguisher D 's advantage can be used by D' to break \mathcal{E} .

<p>If $g(\mathbf{sk}) = 0^t$:</p> <ol style="list-style-type: none"> 1. $\mathbf{pk}' \neq \mathbf{pk}$, set $d = \perp$. 2. Else ($\mathbf{pk}' = \mathbf{pk}$), <ol style="list-style-type: none"> (a) if $(\hat{m}', \pi') = (\hat{m}, \pi)$, set $d = \text{same}^*$. (b) if $\hat{m}' \neq \hat{m}, \pi' = \pi$, set $d = \perp$. (c) else ($\pi' \neq \pi$), check whether $\mathcal{V}((\mathbf{pk}', \hat{m}'), \pi', \Sigma)$ accepts. <ol style="list-style-type: none"> i. If no, set $d = \perp$. ii. If yes, use the extractor Ext to compute $(\mathbf{sk}'', m'') \leftarrow \text{Ext}(\Sigma, \tau, x' = (\mathbf{pk}', \hat{m}'), \pi')$, where the list $Q = ((\mathbf{pk}, \hat{m}), \pi)$. If the extraction fails, then set $d = \perp$; otherwise $d = m''$. 	<p>Else if $g(\mathbf{sk}) = 1 \circ h_z(f_1(\mathbf{sk})) \stackrel{\text{def}}{=} 1 \circ \text{hint}$:</p> <ol style="list-style-type: none"> 1. if $\pi' = \pi$, then set $d = \perp$. 2. else, check if $\mathcal{V}(\mathbf{pk}', \pi', \text{crs})$ verifies, if not set $d = \perp$. Else, compute $(\mathbf{sk}'', m'') \leftarrow \text{Ext}(\Sigma, \tau, x' = (\mathbf{pk}', \hat{m}'), \pi')$, where the list $Q = ((\mathbf{pk}, \hat{m}), \pi)$. If the extraction fails, then set $d = \perp$; otherwise consider the following two cases: <ol style="list-style-type: none"> (a) If $h_z(\mathbf{sk}'') \neq \text{hint}$, then set $d = \perp$. (b) Else, set $d = m''$.
---	--

Fig. 2. The two cases for the reduction

To analyze the reduction, we are going to establish the following claim. We defer the formal proof to the full version of this paper.

Claim. Given the above A and D' , we have

$$\Pr[D'(\mathsf{LE}_0(\mathcal{E}, A, k, \mathcal{F}^{\mathsf{half}})) = 1] - \Pr[D'(\mathsf{LE}_1(\mathcal{E}, A, k, \mathcal{F}^{\mathsf{half}})) = 1] > \varepsilon/2 - \mathsf{ngl}(k).$$

4 Our Compilers

In this section, we present two compilers that use our LR-NM code to secure any functionality G from split-state tampering and leakage attacks. The first compiler, as an intermediate result, outputs a compiled functionality G' that has access to fresh random coins. The second one outputs a deterministic functionality by derandomizing G' using a pseudorandom generator.

Randomized Implementation. Let $G(s, x)$ be an interactive functionality with a k -bit state s that we want to protect, and let $\mathcal{C} = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ be the LR-NM coding scheme we constructed in the previous section. Our compiler works as follows: first it generates the common parameters $\Sigma \leftarrow \mathsf{Init}(1^k)$. Then $\mathsf{MemCompile}(\Sigma, s)$ outputs an encoding of s , $(M_1, M_2) \leftarrow \mathsf{Enc}(\Sigma, s)$; and $\mathsf{CircuitCompile}(G, \mathcal{C}, \Sigma)$ outputs a randomized functionality G' such that $\langle G', \mathsf{Enc}(\Sigma, s) \rangle$ works in the following way: on user input x , first G' decodes the memory using the decoding algorithm Dec . If the outcome is \perp , then G' will always output \perp (equivalently, self-destruct); otherwise it obtains s . Then G' computes $(s_{\mathsf{new}}, y) \leftarrow G(s, x)$ and outputs y . Finally G' re-encodes its memory: $(M_1, M_2) \leftarrow \mathsf{Enc}(\Sigma, s_{\mathsf{new}})$. There are two places where G' uses fresh randomness: the functionality G itself and the re-encoding step.

We denote this randomized hardware implementation of the compiler as $\text{Hardware}_{\text{rand}}(\mathcal{C}, G) \stackrel{\text{def}}{=} \langle G', \text{Enc}(s) \rangle$. Obviously the compiler is correct, i.e. the implementation's input/output behavior is the same as that of the original functionality. Then we are able to achieve the following theorem:

Theorem 2. *Let $t : \mathbb{N} \rightarrow \mathbb{N}$ be some non-decreasing polynomial, and $\mathcal{G}_t, \mathcal{F}^{\text{half}}$, $\mathcal{G}_{t,\text{all}}^{\text{half}}$ be as defined above.*

Suppose we are given a cryptosystem $\mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ that is semantically secure against one-time leakage \mathcal{G}_t ; a robust NIZK $\Pi = (\ell, \mathcal{P}, \mathcal{V}, \mathcal{S})$; and $\mathcal{H}_k : \{h_z : \{0,1\}^{\text{poly}(k)} \rightarrow \{0,1\}^k\}_{z \in \{0,1\}^k}$, a family of universal one-way hash functions. Then the randomized hardware implementation presented above is secure against $\mathcal{F}^{\text{half}}$ tampering and $\mathcal{G}_{t,\text{all}}^{\text{half}}$ leakage.

Let us explain our proof approach. In the previous section, we have shown that the coding scheme is leakage-resilient and non-malleable. This intuitively means that one-time attacks on the hardware implementation $\text{Hardware}_{\text{rand}}(\mathcal{C}, G)$ are useless. Therefore, what we need to show is that these two types of attacks are still useless even when the adversary has launched a continuous attack.

Recall that, by definition, to prove tamper and leakage resilience, we need to exhibit a simulator that simulates the adversary's view of interaction with $\text{Hardware}_{\text{rand}}(\mathcal{C}, G)$ based solely on black-box access to $\langle G, s \rangle$. The simulator computes M_1 and M_2 almost correctly, except it uses $s_0 = 0^k$ instead of the correct s (which, of course, it cannot know). The technically involved part of the proof is to show that the resulting simulation is indistinguishable from the real view; this is done via a hybrid argument in which an adversary that detects that, in round i , the secret changed from s_0 to the real secret s , can be used to break the LR-NM code, since this adversary will be able to distinguish $\text{Tamper}_{s_0}^{f,\Sigma}$ from $\text{Tamper}_s^{f,\Sigma}$ or break the leakage resilience of the code. In doing this hybrid argument, care must be taken: by the time we even get to round i , the adversary may have overwritten the state of the device; also, there are several different ways in which the security may be broken and our reduction relies on a careful case analysis to rule out each way. The formal proof appears in the full version of this paper.

Deterministic Implementation. In the previous section, we showed that the hardware implementation $\text{Hardware}_{\text{rand}}$ with the LR-NM code is leakage-tampering-resilient. In this section, we show how to construct a deterministic implementation by derandomizing the construction. Our main observation is that, since the coding scheme also hides its input string (like an encryption scheme), we can store an encoding of a random seed, and then use a pseudorandom generator to obtain more (pseudo) random bits. Since this seed is protected, the output of the PRG will be pseudorandom, and can be used to update the encoding and the seed. Thus, we have pseudorandom strings for an arbitrary (polynomially bounded) number of rounds. The intuition is straightforward yet the reduction is subtle: we need to be careful to avoid a circular argument in which we rely on the fact that the seed is hidden in order to show that it is hidden.

To get a deterministic implementation for any given functionality $G(\cdot, \cdot)$, we use the coding scheme $\mathcal{C} = (\text{Init}, \text{Enc}, \text{Dec})$ defined in the previous section, and

a pseudorandom generator $g : \{0,1\}^k \rightarrow \{0,1\}^{k+2\ell}$, where ℓ will be defined later. Let $s \in \{0,1\}^k$ be the secret state of $G(\cdot, \cdot)$, and $\text{seed} \in \{0,1\}^k$ be a random k -bit string that will serve as a seed for the PRG. Now we define the compiler. The compiler first generates the common parameters $\Sigma \leftarrow \text{Init}(1^k)$. Then on input $s \in \{0,1\}^k$, $\text{MemCompile}(s)$ first samples a random seed $\text{seed} \in \{0,1\}^k$ and outputs $(M_1, M_2) \leftarrow \mathcal{E}nc(\Sigma, s \circ \text{seed})$ where \circ denotes concatenation. $\text{CircuitCompile}(G)$ outputs a deterministic implementation $\text{Hardware}_{\text{det}}(\mathcal{C}, G) \stackrel{\text{def}}{=} \langle G^*, \Sigma, \mathcal{E}nc, \mathcal{D}ec, \mathcal{E}nc(\Sigma, s \circ r) \rangle$ that works as follows:

G^* on input x does the followings:

- Decode $\mathcal{E}nc(\Sigma, s \circ \text{seed})$ to obtain $s \circ \text{seed}$. Recall that $\mathcal{D}ec$ is deterministic.
- Compute $\text{seed}' \circ r_1 \circ r_2 \leftarrow g(\text{seed})$, where $\text{seed}' \in \{0,1\}^k, r_1, r_2 \in \{0,1\}^\ell$.
- Calculate $(s_{\text{new}}, y) \leftarrow G(s, x)$ (using the string r_1 as a random tape if G is randomized), then outputs y , and updates the state to be s_{new} .
- Calculate the encoding of $s' \circ \text{seed}'$ using the string r_2 as a random tape. Then it stores the new encoding $\mathcal{E}nc(\Sigma, s_{\text{new}} \circ \text{seed}')$.

Fig. 3. The deterministic implementation

In this implementation $\text{Hardware}_{\text{det}}$, we only use truly random coins when initializing the device, and then we update it deterministically afterwards. Let us show that the implementation $\text{Hardware}_{\text{det}}(\mathcal{C}, G)$ is also secure against $\mathcal{F}^{\text{half}}$ tampering and $\mathcal{G}_{t,\text{all}}^{\text{half}}$ leakage. We achieve the following theorem.

Theorem 3. *Let $t : \mathbb{N} \rightarrow \mathbb{N}$ be some non-decreasing polynomial, and $\mathcal{G}_t, \mathcal{F}^{\text{half}}, \mathcal{G}_{t,\text{all}}^{\text{half}}$ be as defined in the previous section.*

Suppose we are given a crypto system $\mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ that is semantically secure against one-time leakage \mathcal{G}_t ; a robust NIZK $\Pi = (\ell, \mathcal{P}, \mathcal{V}, \mathcal{S})$; and $\mathcal{H}_k : \{h_z : \{0,1\}^{\text{poly}(k)} \rightarrow \{0,1\}^k\}_{z \in \{0,1\}^k}$, a family of universal one-way hash functions. Then the deterministic hardware implementation presented above is secure against $\mathcal{F}^{\text{half}}$ tampering and $\mathcal{G}_{t,\text{all}}^{\text{half}}$ leakage.

Combining the above theorem and the Naor-Segev Leakage-resilient encryption scheme [30], we are obtain the following corollary.

Corollary 1. *Under the decisional Diffie-Hellman assumption and the existence of robust NIZK, for any polynomial $t(\cdot)$, there exists a coding scheme with the deterministic hardware implementation presented above that is secure against $\mathcal{F}^{\text{half}}$ tampering and $\mathcal{G}_{t,\text{all}}^{\text{half}}$ leakage.*

The formal proof appears in the full version of this paper.

Acknowledgement. We thank Yevgeniy Dodis for useful discussions. This work was supported by NSF grants 1012060, 0964379, 0831293.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM Side-Channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
4. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
5. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: FOCS, pp. 501–510. IEEE (2010)
6. Choi, S.G., Kiayias, A., Malkin, T.: BiTR: Built-in Tamper Resilience. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 740–758. Springer, Heidelberg (2011)
7. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust Non-interactive Zero Knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
8. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS, pp. 511–520 (2010)
9. Dodis, Y., Lewko, A.B., Waters, B., Wichs, D.: Storing secrets on continually leaky devices. In: FOCS, pp. 688–697 (2011)
10. Dodis, Y., Pietrzak, K.: Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (2010)
11. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. 30(2), 391–437 (2000)
12. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302. IEEE (2008)
13. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: ICS, pp. 434–452 (2010)
14. Faust, S., Pietrzak, K., Venturi, D.: Tamper-Proof Circuits: How to Trade Leakage for Tamper-Resilience. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 391–402. Springer, Heidelberg (2011)
15. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
16. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic Tamper-Proof (ATP) Security: Theoretical Foundations for Security against Hardware Tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
17. Goldwasser, S., Rothblum, G.N.: Securing Computation against Continuous Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010)

18. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX Security Symposium, pp. 45–60 (2008)
19. Halevi, S., Lin, H.: After-the-Fact Leakage in Public-Key Encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 107–124. Springer, Heidelberg (2011)
20. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private Circuits II: Keeping Secrets in Tamperable Circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
21. Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
22. Juma, A., Vahlis, Y.: Protecting Cryptographic Keys against Continual Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
23. Kalai, Y.T., Kanukurthi, B., Sahai, A.: Cryptography with Tamperable and Leaky Memory. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 373–390. Springer, Heidelberg (2011)
24. Katz, J., Vaikuntanathan, V.: Signature Schemes with Bounded Leakage Resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
25. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
26. Lewko, A.B., Lewko, M., Waters, B.: How to leak on key updates. In: STOC, pp. 725–734. ACM (2011)
27. Lewko, A., Rouselakis, Y., Waters, B.: Achieving Leakage Resilience through Dual System Encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 70–88. Springer, Heidelberg (2011)
28. Liu, F.-H., Lysyanskaya, A.: Algorithmic Tamper-Proof Security under Probing Attacks. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 106–120. Springer, Heidelberg (2010)
29. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
30. Naor, M., Segev, G.: Public-Key Cryptosystems Resilient to Key Leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
31. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
32. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553. IEEE (1999)

Securing Circuits against Constant-Rate Tampering

Dana Dachman-Soled and Yael Tauman Kalai

Microsoft Research New England

Abstract. We present a compiler that converts any circuit into one that remains secure even if a *constant fraction* of its wires are tampered with. Following the seminal work of Ishai *et. al.* (Eurocrypt 2006), we consider adversaries who may choose an arbitrary set of wires to corrupt, and may set each such wire to 0 or to 1, or may toggle with the wire. We prove that such adversaries, who *continuously* tamper with the circuit, can learn at most *logarithmically many bits* of secret information (in addition to black-box access to the circuit). Our results are information theoretic.

Keywords: side-channel attacks, tampering, circuit compiler, PCP of proximity.

1 Introduction

In recent years, there has been a proliferation of physical attacks on cryptographic schemes. Such attacks exploit the implementation (rather than the functionality) of the scheme. For instance, Kocher *et al.* [46] demonstrated how one can possibly learn the secret key of an encryption scheme by measuring the power consumed during an encryption operation, or by measuring the time it takes for the operation to complete [45]. Other types of physical attacks include: inducing faults to the computation [5,7,46], using electromagnetic radiation [30,55,54], and several others [54,43,47,35]. These physical attacks have proven to be a significant threat to the practical security of cryptographic devices.

Traditional cryptographic models do not take such attacks into account since they idealize the parties' interaction and implicitly assume that an adversary may only observe an honest party's input-output behavior. Recently, a large and growing body of research has sought to introduce more realistic models and to secure cryptographic systems against such physical attacks. The vast majority of these works focus on securing cryptographic schemes against various *leakage* attacks (e.g. [10,38,49,31,37,20,51,1,50,42,18,17,24,39,34]). In these attacks an adversary plays a passive role, learning information about the honest party through side-channels but not attempting to interfere with the honest party's computation. However, as mentioned above, physical attacks are not limited to leakage, and include active *tampering* attacks, where an adversary may actively modify the honest party's memory or circuit. The focus of this work is to construct schemes that are secure even in the presence of tampering (we elaborate on related work in Section 1.3).

1.1 Our Results

We show a general compiler that converts any circuit into one that is resilient to (a certain form of) tampering. We consider the tampering model of Ishai *et al.* [37]. Namely, we consider *adversarial* faults, where during each run the computationally unbounded adversary can tamper with any (bounded) set of wires of his choice.

We note that we cannot hope to get correctness, since the adversary may simply tamper with the final output wire. Thus, following [37], we do not attempt to guarantee correctness, but instead devote our efforts to ensuring *privacy*. More specifically, we consider circuits that are associated with a secret state (such as a cryptographic key). We model such circuits as standard circuits (with AND, OR, and NOT gates), with additional secret memory that contains the secret state. The circuit itself is thought of as being public and known to the adversary, whereas the memory content is secret. Following the terminology of [37], we refer to such circuits as *private circuits*. Our goal is to protect the secret state of the circuit even when an adversary may run the circuit on arbitrary inputs while continuously tampering with the circuit.

Unlike the leakage regime, where we build cryptographic schemes that are secure against *arbitrary* (bounded) leakage, in the tampering regime, we focus on limited types of adversarial tampering attacks. Indeed, it is not hard to see that it is impossible to construct private circuits resilient to arbitrary tampering attacks, since an adversary may modify the circuit so that it simply outputs the entire secret state in memory. As in [37], we only allow the adversary to tamper with individual wires [37,25] and individual memory gates [13,31,21]. More specifically, in each run of the circuit we allow the adversary to specify a set of tampering instructions, where each instruction is of the form: Set a wire (or a memory gate) to 0 or 1, or toggle with the value on a wire (or a memory gate). However, in contrast to [37], we allow the adversary to tamper with any *constant fraction* of wires and memory gates in the circuit. We note that the tampering allowed in [37] is very local: To be resilient against tampering with t wires per run, they blow up the size of the circuit by a factor of at least t ; thus their overall tampering rate is very small compared to the size of the circuit.

As noted by [31], it is impossible to construct private circuits resilient against tampering on wires without allowing feedback into memory, i.e. without allowing the circuit to overwrite its own memory. Otherwise, an adversary may simply set to 0 or 1 one memory gate at a time and observe whether the final output is modified or not. Thus, the adversary may often learn the entire internal state of the circuit by setting one wire at a time.

Even if we allow feedback, and place limitations on the type of tampering we allow, it is not a priori clear how to build tamper-resilient circuits. As pointed out in [37], the fundamental problem is that the part of the circuit which is supposed to detect tampering and overwrite the memory, may itself be tampered with. Indeed, this “self-destruct” mechanism itself needs to be resilient to tampering.

As in [37], we prove security using a simulation based definition, where we require that for any adversary who continually tampers with the circuit (as described above), there exists a simulator who simulates the adversary’s view. However, whereas [37] give the simulator only black-box access to the circuit, we also give the simulator logarithmic number of leakage bits on the secret state. We note that for many applications, leakage

of only logarithmically many bits of the secret key does not break the security of the primitive. This is because an adversary may simply *guess* these bits of leakage on his own.

We also note that our compiler is deterministic, and for deterministic constructions such leakage is necessary. Loosely speaking, this is the case since an adversary can always leak a memory gate of its choice, by checking whether setting that memory gate to 0 affects the functionality of the circuit. We note that if the compiler is deterministic then many of the memory gates contain “sensitive” information. This is not necessarily the case if the compiler is randomized since then the secret state can be secret-shared in memory, in which case each memory gate on its own may contain a truly random bit.

Our Results More Formally. We present a general compiler \mathcal{T} that converts a circuit C with a secret state s (denoted by C_s) into a circuit $\tilde{C}_{\tilde{S}}$ with a secret state \tilde{S} . We consider computationally unbounded adversaries \mathcal{A} who receive access to $\tilde{C}_{\tilde{S}}$ and behave in the following way: \mathcal{A} runs the circuit many times with arbitrary and adaptively chosen inputs. In addition, during each run of the circuit the adversary \mathcal{A} may specify tampering instructions of the form “set wire w to 1”, “set wire w to 0”, “flip value of wire w ”, as well as “set memory gate g to 1”, “set memory gate g to 0”, “flip value of memory gate g ”, for any wire w or memory gate g . We restrict the number of tampering instructions \mathcal{A} may specify per run to be at most $\lambda \cdot \sigma$, where $\lambda > 0$ is some constant and σ is the size of the circuit $\tilde{C}_{\tilde{S}}$. Thus, in each run, \mathcal{A} may tamper with a constant fraction of wires and memory gates. Again, we emphasize that our result does not rely on computational assumptions and is completely information theoretic. Moreover, our result does not rely on any source of randomness and our compiler is deterministic.

Theorem 1 (Main Theorem, Informal). *There exists a universal constant $\lambda > 0$ and an efficient transformation \mathcal{T} which takes as input any circuit C_s with private state s and outputs a circuit $\tilde{C}_{\tilde{S}}$ with private state \tilde{S} such that the following two conditions hold:*

Correctness: For every input x , $C_s(x) = \tilde{C}_{\tilde{S}}(x)$.

Tamper-Resilience: For every adversary \mathcal{A} , which may tamper with a λ -fraction of wires and memory gates in $\tilde{C}_{\tilde{S}}$ per run, there exists a simulator Sim , which can simulate the view of \mathcal{A} given only black-box access to C_s and $\log(T)$ bits of leakage on s , where T is an upper bound on the number of times that \mathcal{A} runs the (tampered) circuit.¹ Moreover, the runtime of Sim is polynomial in the runtime of \mathcal{A} and C_s .

Intuitively, the theorem asserts that adversaries who may observe the input-output behavior of the circuit while tampering with at most a λ -fraction of wires and memory gates in each run, do not get too much extra knowledge over what they could learn from just input-output access to the circuit. More specifically, running the (tampered) circuit $\text{poly}(|s|)$ times leaks at most $O(\log |s|)$ bits.

We remark that this relaxation of allowing $O(\log |s|)$ bits of leakage was first considered by Faust *et. al.* [25]. We additionally mention, as we argued previously, that leaking at most $O(\log |s|)$ bits does not compromise security of many applications.

¹ The reader can think of T as polynomial in the security parameter.

This is because an adversary receiving only input-output access to the circuit may simply *guess* the $O(\log |s|)$ bits of leakage, and his guess will be correct with probability $1/2^{O(\log |s|)} = 1/\text{poly}(|s|)$. Thus, we don't lose much by leaking $O(\log |s|)$ bits of the secret key.

1.2 Comparison with Ishai *et al.* [37]

Our work is inspired by the work of [37]. As in our work, they too consider circuits with memory gates, and consider the same type of faults as we do. Similarly to us, they construct a general compiler that converts any private circuit into a tamper resilient one. However our work differs from their work in several ways.

- The tamper resilient circuits in [37] require the use of “randomness gates”, which output a fresh random bit in each run of the circuit. Alternatively, [37] can get rid of these randomness gates at the cost of relying on a computational assumption. Their computational result, which does not require randomness gates, relies on the existence of one-way functions and considers only polynomially bounded adversaries. In contrast, our compiler and the resulting tamper-resilient circuits are deterministic, and provide information theoretical security.
- As mentioned previously, [37] constructs tamper resilient circuits that are resilient only to local tampering. More specifically, in order to be resilient to tampering with t wires per run, they blow up the circuit size by a factor of at least t . In contrast, our tamper-resilient circuits are resilient to a constant fraction of tampering anywhere in the circuit, and thus are robust to global tampering.
- In the tamper resilient circuits of [37], there is one gate G that has a large fan-out. In practice, however, large fan-out gates do not exist, and instead they are implemented using a *splitter*, which takes as input the output wire of a gate and duplicates the wire many times. Unfortunately, such an implementation is not resilient to tampering with the output wire of G (which is the input wire to the splitter), since if this wire is tampered with, then it causes an immediate tampering with *all* the output wires of the splitter. In contrast, in our work all the gates of the tamper resilient circuits have constant fan-out (and some gates use splitters). However, due to lack of space, in this extended abstract we focus on the initial constructions that do assume the existence of a gate with large fan-out. We refer the reader to the full version [14] for details on how we remove this large fan-out gate.
- One advantage of the tampering model of [37] over our model, is that their model allows for “persistent faults”, e.g., if a value of some wire is fixed during one run, it remains set to that value in subsequent runs. We note that in our case, we cannot in general allow persistent faults across runs of the circuit. However, we allow “persistent faults” on memory gates so if a memory value is modified during one run, it remains modified for all subsequent runs.
- Another advantage of [37] is that their security definition is slightly stronger than ours. They guarantee that any adversary that continually tampers with the circuit (as described above) can be simulated given only black-box access to the circuit, whereas in our security definition, we give the simulator in addition logarithmically many bits of information.

Extending Our Result to Allow for Both Leakage and Tampering. Note that so far, we considered adversaries who only tamper with the wires and the memory of the circuit. Our result can be extended to consider adversaries who both tamper and leak. More specifically, we show a general compiler that converts any circuit into one that is resilient against both tampering (in the model described above) and leakage. The leakage model that we consider is OCL (“only computation leaks”) leakage, as defined by Micali and Reyzin [49].

Unfortunately, being resilient to leakage (in addition to tampering) comes at a price: First, the resulting circuit no longer tolerates a constant-fraction of tampering, but rather security is ensured as long as $1/\text{poly}(k)$ -fraction of the wires (or memory gates) can be tampered with. In addition, the result is no longer information theoretical, and relies on cryptographic assumptions. Specifically it relies on the existence of a non-committing encryption scheme [11] (which can be instantiated from hardness of factoring Blum integers, CDH and LWE [11,15,12]), and on the existence of a fully homomorphic encryption scheme (which can be instantiated from LWE [32,9,8]). Due to lack of space, we defer our result in the leakage setting to the full version.

1.3 Related Work

The problem of constructing error resilient circuits dates back to the work of Von Neumann from 1956 [56]. Von Neumann studied a model of *random* errors, where each gate has an (arbitrary) error independently with small fixed probability, and his goal was to obtain *correctness* (as opposed to privacy). There have been numerous follow up papers to this seminal work, including [16,53,52,27,23,36,28,22], who considered the same noise model, ultimately showing that any circuit of size σ can be encoded into a circuit of size $O(\sigma \log \sigma)$ that tolerates a fixed constant noise rate, and that any such encoding must have size $\Omega(\sigma \log \sigma)$.

There has been little work on constructing circuits resilient to *adversarial* faults, while guaranteeing correctness. The main works in this arena are those of Kalai *et al.* [41], Kleitman *et al.* [44], and Gál and Szegedy [29]. The works of [44] and [41] consider a different model where the only type of faults allowed are short-circuiting gates. [29] consider a model that allows arbitrary faults on gates, and show how to construct tamper-resilient circuits for symmetric Boolean functions. We note that [29] allow a constant fraction δ of adversarial faults *per level* of the circuit. Moreover, if there are less than $1/\delta$ gates on some level, they allow no tampering at all on that level. [29] also give a more general construction for any efficiently computable function which relies on PCP’s. However, in order for their construction to work, they require an entire PCP proof π of correctness of the output to be precomputed and handed along with the input to the tamper-resilient circuit. Thus, they assume that the input to the circuit is already encoded via an encoding which depends on the *output* value of that very circuit. We also use the PCP methodology in our result, but do not require any precomputations or that the input be encoded in some special format.

Recently, the problem of physical attacks has come to the forefront in the cryptography community. From the viewpoint of cryptography, the main focus is no longer to ensure correctness, but to ensure *privacy*. Namely, we would like to protect the honest party’s secret information from being compromised through the physical attacks

of an adversary. There has been much work on protecting circuits against leakage attacks [38,49,20,51,19,26,39,34]. However, there has not been much previous work on constructing circuits resilient to tampering attacks. In this arena, there have been two categories of works. The works of [31,21,13,48,40] allow the adversary to only tamper with and/or leak on the *memory* of the circuit in between runs of the circuit, but do not allow the adversary to tamper with the circuit itself. Due to lack of space we do not elaborate on these related works in this extended abstract, and refer the reader to the full version for details. We note that this model of allowing tampering only with memory is very similar to the problem of "related key attacks" (see [3,2] and references therein). In contrast, in our work, as well as in the works of [37,25], the focus is on constructing circuits resilient to tampering with both the memory as well as the wires of the circuit.

Faust *et al.* [25] consider a model that is reminiscent to the model of [37] and to the model we consider here. They consider adversarial faults where the adversary may actually tamper with all wires of the circuit but each tampering attack fails independently with some probability δ . As in our case, they also allow the adversary to learn a logarithmic number of bits of information on the secret key. However, their result requires the use of small tamper-proof hardware components.

1.4 Our Techniques

Our conceptual approach is similar to [37]: The tamper-resilient circuit has an "error-detection" mechanism, and if an error is detected then the circuit self-destructs (i.e. the memory is zeroed out). However, our techniques for achieving this are very different.

Recall that our goal is to guarantee security against an adversary who may tamper with a *constant fraction* of the wires and memory gates. We begin with the simple observation that circuits in NC^0 are inherently tamper-resilient. In particular, since each output bit of an NC^0 circuit is computed by a constant-size circuit (say of size $\tilde{\sigma}$), if we allow $\alpha \cdot 1/\tilde{\sigma}$ -fraction of tampering for some constant $\alpha < 1$, then we ensure that at least $(1 - \alpha)$ -fraction of the output bits are exactly correct.

In order to construct a good error-detection mechanism for our tamper-resilient circuit, we need to construct a mechanism that detects errors and is itself resilient to a constant-fraction of tampering. More specifically, we would like to construct a "robust" verifier in NC^0 for the computation of the original circuit, such that if an error is detected in the computation, then most of the circuit's output are set to 0. Fortunately, the PCPP (PCP of Proximity) Theorem [4] provides us with exactly such a verifier. Informally, in a PCPP for an NP language \mathcal{L} , the verifier is given oracle access to a PCP π and is also given oracle access to an instance z . The verifier tosses logarithmically many random coins and it queries a *constant* number of bits of π and z . It accepts if $z \in \mathcal{L}$ and if the PCP π was generated honestly, and it rejects (with probability $1/2$) if z is far from being in the language (even if π was generated maliciously). We refer the reader to Section 2 for details.

Note that for any fixed setting of the random coins, the corresponding verifier is of constant size since it has only a constant number of input bits. Thus, by constructing a separate verifier corresponding to each possible outcome of the random coins and outputting the output bits of all verifiers, we obtain a single (larger) verifier in NC^0 with the property that if z is far from \mathcal{L} then at least a $1/2$ -fraction of the verifier's

outputs are 0. Note that the soundness property only holds when z is "far" from every string in \mathcal{L} . Thus, our transformed circuit has the following basic paradigm: We encode the secret state s and the input x using an error-correcting code to obtain $S = \text{ECC}(s)$ and $X = \text{ECC}(x)$. We compute $b = C_s(x)$, and we compute a PCPP proof π that $(b, S \circ X) \in \mathcal{L}$ where $\mathcal{L} = \{(b, S \circ X) \mid \exists s, x : S = \text{ECC}(s), X = \text{ECC}(x), b = C_s(x)\}$. Then we verify the proof and self-destruct (i.e. erase all memory) if any of the output bits of the verifier are 0. Additional work is required to go from tolerating a constant-fraction of tampering in the error-detection stage to tolerating a constant-fraction of tampering overall. We elaborate on these in Section 4.2.

We mention that the resulting tamper-resilient circuit has one gate with large fan-out. Additional ideas are needed in order to remove the need of such a large fan-out gate. Due to lack of space we elaborate on these in the full version.

We note that the techniques mentioned so far are used to get resilience only against tampering attacks (and not leakage attacks). In order to get security against both (continual) leakage and tampering, we rely on techniques in the leakage regime, and in particular rely on recent results in the OCL model [49,39,34,33].² Due to lack of space we defer the details to the full version.

2 PCP of Proximity

In this section, we present necessary preliminaries on PCP of proximities (denoted by PCPP), and state the efficient PCPP theorem of [4]. A PCP of proximity is a relaxation of a standard PCP, that only verifies that the input is *close* to an element of the language. The advantage of this relaxation is that it allows the possibility that the verifier may read only a small number of bits from the input. For greater generality, the input is divided into two parts (a, z) , where a is given explicitly to the verifier, and z is given only as an oracle. Thus PCPs of proximity consider languages which consist of pairs of strings, and therefore are referred to as *pair languages*.

Definition 1 (Pair language). A pair language L is simply a subset of the set of string pairs $L \subseteq \{0, 1\}^* \times \{0, 1\}^*$. For every $a \in \{0, 1\}^*$, we denote $L_a = \{z \in \{0, 1\}^* : (a, z) \in L\}$. We usually denote $\ell = |a|$ and $K = |z|$.

Definition 2 (Relative Hamming distance). Let $z, z' \in \{0, 1\}^K$ be two strings. The relative Hamming distance between z and z' is defined as

$$\Delta(z, z') \triangleq |\{i \in [K] : z_i \neq z'_i\}|/K.$$

We say that z is δ -far from z' if $\Delta(z, z') \geq \delta$. More generally, let $S \subseteq \{0, 1\}^K$; we say z is δ -far from S if $\Delta(z, z') \geq \delta$ for every $z' \in S$.

Definition 3 (PCP Verifiers). A verifier is a probabilistic polynomial time algorithm V that, on an input x , tosses $r = r(|x|)$ random coins and generates a sequence of

² We actually rely on the recently constructed LDS compiler [6], which in turn is based on OCL compilers.

$q = q(|x|)$ queries $I = (i_1, \dots, i_q)$ and a circuit $D : \{0, 1\}^q \rightarrow \{0, 1\}$ of size at most $d(|x|)$.³

We think of V as representing a probabilistic oracle machine that queries its oracle π at positions I , receives the q answer bits $\pi|_I \triangleq (\pi_{i_1}, \dots, \pi_{i_q})$, and accepts if and only if $D(\pi|_I) = 1$. We write $(I, D) \leftarrow V(x)$ to denote the queries and circuit generated by V on input x (and random coin tosses). We call r the randomness complexity, q the query complexity, and d the decision complexity of V .

Definition 4 (PCPP verifier for a pair language [4]). For functions $s, \delta : \mathbb{N} \rightarrow [0, 1]$, a verifier V is a probabilistically checkable proof of proximity (PCPP) system for a pair language L with proximity parameter δ and soundness error s , if the following two conditions hold for every pair of strings (a, z) :

Completeness: If $(a, z) \in L$ then there exists π such that $V(a)$ accepts oracle $z \circ \pi$ with probability 1. Formally

$$\exists \pi \Pr_{(I, D) \leftarrow V(a)} [D((z, \pi)|_I) = 1] = 1.$$

Soundness: If z is $\delta(|a|)$ -far from $L(a)$, then for every π , the verifier $V(a)$ accepts oracle $z \circ \pi$ with probability strictly less than $s(|a|)$. Formally,

$$\forall \pi \Pr_{(I, D) \leftarrow V(a)} [D((z \circ \pi)|_I) = 1] < s(|a|).$$

Theorem 2 (Efficient PCPP for Pair-language (Theorem 3.3 of [4])). Let $T : \mathcal{N} \rightarrow \mathcal{N}$ be a non-decreasing function, and let L be a pair language. If L can be decided in time T ,⁴ then for every constant $\rho \in (0, 1)$ there exists a PCP of proximity for L with randomness complexity $O(\log T)$, query complexity $q = O(1/\rho)$, perfect completeness, soundness error $1/2$, and proximity parameter ρ .

We mention that [4] does not discuss the complexity of constructing the PCPP proof, but the efficiency follows by a close inspection of their construction.

3 The Tampering Model

3.1 Circuits with Memory Gates

Similarly to [37], we consider a circuit model that includes memory gates. Namely, a circuit consists of (the usual) AND, OR, and NOT gates, connected to each other via wires, as well as input wires and output wires. In addition, a circuit may have memory gates. Each memory gate has one (or more) input wires and one (or more) output wires. Each memory gate is initialized with a bit value 0 or 1. This value can be updated during each run of the circuit.

³ For the sake of simplicity we consider only bit queries.

⁴ L can be decided in time T if there exists a Turing machine M such that for every input $(a, b) \in \{0, 1\}^* \times \{0, 1\}^*$, $M(a, b) = 1$ if and only if $(a, b) \in L$, and $M(a, b)$ runs in time $T(|a| + |b|)$.

Each time the circuit is run with some input x , all the wires obtain a 0/1 value. The values of the input wires to the memory gates define the way the memory is updated. We allow only two types of updates: delete or unchange. Specifically, if an input wire to a memory gate has the value 0, then the memory gate is overwritten with the value 0. If an input wire to a memory gate has the value 1, then the value of the memory gate remains unchanged. We denote a circuit C initialized with memory s by C_s .

3.2 Tampering Attacks

We consider adversaries, that can carry out the following attack: The adversary has black-box access to the circuit, and thus can repeatedly run the circuit on inputs of his choice. Each time the adversary runs the circuit with some input x , he can tamper with the wires and the memory gates. We consider the following type of faults: Setting a wire (or a memory gate) to 0 or 1, or toggling with the value on a wire (or a memory gate).

More specifically, the adversary can adaptively choose an input x_i and a set of tampering instructions (as above), and he receives the output of the tampered circuit on input x_i . He can do this adaptively as many times as he wishes. We emphasize that once the memory has been updated, say from s to s' , the adversary no longer has access to the original circuit C_s , and now only has access to $C_{s'}$. Namely, the memory errors are persistent, while the wire errors are not persistent.

We denote by $\text{TAMP}_{\mathcal{A}}(C_s)$ the output of an adversary \mathcal{A} that carries out the above tampering attack on a circuit C_s . We say that an adversary \mathcal{A} is a λ -tampering adversary if during each run of the circuit he tampers with at most a λ -fraction of the circuit. Namely, \mathcal{A} can make at most $\lambda \cdot |C_s|$ tampering instructions for each run, where each instruction corresponds either to a wire tampering or to a memory gate tampering.

Remark. In this work, we define the size of a circuit C , denoted by $|C|$, as the number of wires in C plus the number of memory gates in C . Note that this is not the common definition (where usually the size includes also the gates); however, it is equivalent to the common definition up to constant factors.

To define security of a circuit against tampering attacks we use a simulation-based definition, where we compare the real world, where an adversary \mathcal{A} (repeatedly) tampers with a circuit C_s as above, to a simulated world, where a simulator Sim tries to simulate the output of \mathcal{A} , while given only black-box access to the circuit C_s , and without tampering with the circuit at all.

In this work, we give the simulator Sim , in addition to black box access to the circuit C_s , the privilege to request $\log T$ bits of information about s , where T is the number of times \mathcal{A} runs the tampered circuit. More specifically, the simulator, in addition to black-box access to the circuit C_s , is also given oracle access to a leakage oracle that takes as input any leakage request, represented as a boolean circuit $L : \{0, 1\}^k \rightarrow \{0, 1\}$ and returns $L(s)$. The simulator Sim may query the leakage oracle at most $\log T$ times. We denote the output of Sim by $\text{LeakBB}_{\text{Sim}, \log T}(C_s)$. As noted in the Introduction, this leakage is necessary if we restrict ourselves to deterministic constructions.

Definition 5. We say that a circuit C_s is secure against λ -tampering adversaries, if for every λ -tampering adversary \mathcal{A} there exists a simulator Sim , that runs in time $\text{poly}(|\mathcal{A}|, |C_s|)$, such that

$$\{\text{TAMP}_{\mathcal{A}}(C_s)\}_{k \in \mathbb{N}} \equiv \{\text{LeakBB}_{\text{Sim}, \log T}(C_s)\}_{k \in \mathbb{N}},$$

where T is an upper bound on the number of times that \mathcal{A} runs the (tampered) circuit.

We give an efficient method for constructing circuits that remain secure against adversaries that tamper with a constant fraction of the wires in the circuit. Namely, we prove that there exists a constant $\lambda > 0$ such that the resulting circuit is secure against λ -tampering adversaries.

4 The Compiler

In this section, we present our efficient compiler \mathcal{T} , which takes a circuit C_s , with memory containing a secret $s \in \{0, 1\}^k$, and transforms it into a tamper-resilient circuit.

We describe our compiler in stages:

- First, we present a compiler that takes as input a circuit C_s and outputs a circuit $C_S^{(1)}$, which we partition into four segments. We prove that $C_S^{(1)}$ is secure against adversaries that tamper with at most a constant fraction of the memory gates, and tamper with at most a constant fraction of the wires in Segment 2, Segment 3, and Segment 4 of the circuit (and may tamper arbitrarily with Segment 1 of the circuit). We refer to such security as security against *local tampering*.
- Then, we show how to make the size of the memory, and the size of Segments 2-4, large enough so that the resulting circuit, denoted by $C_{\tilde{S}}^{(2)}$, is secure against adversaries who tamper with at most a constant fraction of the circuit *overall*. Namely, we prove that there is a constant $\lambda > 0$ such that the resulting circuit, $C_{\tilde{S}}^{(2)}$, is secure against any adversary that for each run of the circuit sends at most $\lambda \cdot |C_{\tilde{S}}^{(2)}|$ tampering instructions, where each tampering instruction is either a wire tampering or a memory tampering.

We note that both constructions have an AND gate (denoted by G_{cas}) which has a large fan-out. In the full version we show how using some additional ideas, one can bypass the need for such a gate.

4.1 The Compiler: First Construction

Let $\text{ECC}(\cdot)$ be a binary error correcting code which can efficiently correct a constant fraction of errors $\delta > 0$. We denote the (efficient) decoding procedure by Dec .

In what follows we present a compiler that takes as input a circuit C_s and outputs a circuit $C_S^{(1)}$. The circuit $C_S^{(1)}$ takes as input a string $x \in \{0, 1\}^n$ and outputs a bit b . In the case of no tampering, we show the correctness property: $C_S^{(1)}(x) = C_s(x)$. Moreover, we prove that the circuit $C_S^{(1)}$ is resilient to *local tampering*.

Remark. For simplicity we assume that $s = 0^k$ is “illegal.” We note that this assumption is not necessary, and is made to simplify the construction and the analysis.

We next describe the circuit $C_S^{(1)}$, which we partition into four segments:

Memory: Encoding Secret s . The encoding $S = \text{ECC}(s)$ is placed in the memory of $C^{(1)}$.

Segment 1: This segment consists of three sub-segments.

1. **Encoding Input x .** The first sub-computation encodes the public input x using ECC. Namely, it takes as input x and outputs $\text{ECC}(x)$.

We assume without loss of generality that $|\text{ECC}(x)| = |\text{ECC}(s)|$. This is without loss of generality since if, for example, $|x| < |s|$ then the encoding procedure will first artificially increase x by appending zeros to it, and then encode; and similarly if $|s| < |x|$.

2. **Circuit Computation.** The second sub-computation computes the output bit $b = C_s(x)$. Namely, it takes as input x and S and outputs $b = C_s(x)$, where $s = \text{Dec}(S)$.
3. **PCPP Computation.** The third sub-computation computes a PCPP for the following pair language:

$$\mathcal{L} = \{(b, X \circ S) \mid \exists x \in \{0, 1\}^n, s \in \{0, 1\}^k \setminus \{0^k\} : X = \text{ECC}(x), S = \text{ECC}(s), b = C_s(x)\}$$

Namely, it takes as input the pair $(b, X \circ S)$ and outputs π , which is a PCP of proximity for the statement $(b, X \circ S) \in \mathcal{L}$.

The PCPP we use is one with randomness complexity $r = O(\log |C_s|)$, query complexity $q = O(1/\rho)$ for $\rho = \frac{\delta}{6}$,⁵ perfect completeness, and soundness $1/2$, where the verifier rejects with probability at least $1/2$ statements $(b, X \circ S)$, for which $X \circ S$ is ρ -far from the language L_b . The existence of such a PCPP follows from Theorem 2 (see Section 2).

We note that the outputs of each of the above sub-computations are used by many other subcomputations. Thus, each output wire of each of the above sub-computations is split into many output wires. These output wires all belong to Segment 2, except one output wire (which computes the bit $b = C_s(x)$, and belongs to Segment 4).

Segment 2: PCPP Verification. This segment consists of $\tau \triangleq 2^r = \text{poly}(|C_s|)$ circuits, C_{v_i} , one for every possible PCPP verifier. Note that each verifier circuit has constant size, which we denote by $\tilde{\sigma}_v$.⁶ Each verifier circuit C_{v_i} takes as input q bits from $(X \circ S, b, \pi)$, and outputs a single bit β_i . All the output wires of the circuits C_{v_i} are fed as input to a single AND gate G_{cas} with fan-in τ . Thus, the output of G_{cas} is $\bigwedge_{1 \leq i \leq \tau} \beta_i$.

Let K be the number of bits in S , the encoding of the secret input s . The AND gate G_{cas} (from Segment 2) has fan-out $2K$, where the first K of the output wires belong to Segment 3 and the other K output wires belong to Segment 4. We denote the values of the output wires of G_{cas} by $\{\gamma_j\}_{j=1}^{2K}$.

Segment 3: Error Cascade. This segment contains only the first K output wires of G_{cas} , which have values $\{\gamma_j\}_{j=1}^K$. For every $j \in [K]$, the j 'th output wire of G_{cas} is fed as input to memory gate j . Thus, if $\gamma_j = 0$, memory position j is overwritten with a 0. If $\gamma_j = 1$, memory position j is unchanged.

⁵ Recall that δ is the fraction of errors that the error-correcting code ECC can correct.

⁶ We note that the amount of tampering we ultimately tolerate depends on $\tilde{\sigma}_v$.

Segment 4: The Output. This segment consists of the rest of the circuit: The other K output wires of G_{cas} , which have values $\{\gamma_j\}_{j=K+1}^{2K}$, and a single AND gate G_{out} which has fan-in $K+1$ and outputs a single bit. G_{out} takes as input $(b, \{\gamma_j\}_{j=K+1}^{2K})$ (all these input wires are part of this segment), and outputs $\tilde{b} = b \wedge (\bigwedge_{K+1 \leq j \leq 2K} \gamma_j)$. We note that Segment 4 also includes the output wire of G_{out} , which is the final output of the circuit $C_S^{(1)}(x)$.

The compiled circuit $C_S^{(1)}$ is depicted in Figure 1.

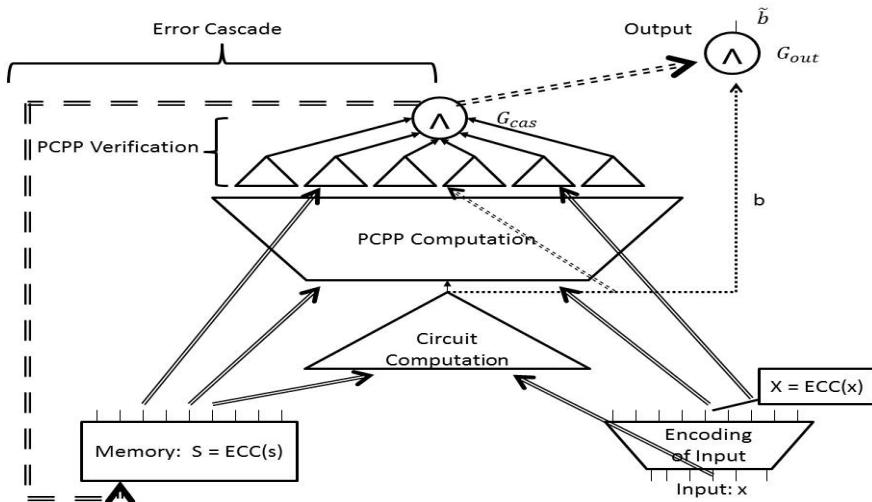


Fig. 1. The compiled circuit $C_S^{(1)}$. We show here the Memory and the 4 segments of the compiled circuit. Note that the encoding S and the input x are fed into the Circuit Computation Stage of Segment 1. Additionally, the encodings S and X are fed into the PCPP Computation Stage of Segment 1, and are fed to Segment 2 (the PCPP Verification Stage) along with the bit b , the output of the Circuit Computation Stage. The outputs of G_{cas} feed back into memory (this is Segment 3—Error Cascade) and to the final output gate G_{out} (in Segment 4) along with the output bit b of the Circuit Computation Stage. The final output of the circuit is \tilde{b} .

The circuit $C_S^{(1)}$ is secure against an adversary \mathcal{A} who may tamper as follows:

- \mathcal{A} can tamper with any number of wires in Segment 1 of the circuit (which consists of the encoding computation of the input x , the circuit computation $C_s(x)$, and the PCPP computation).
- \mathcal{A} can tamper with at most λ -fraction of the wires in Segment 2 (PCPP verification), λ -fraction of the wires in Segment 3 (error cascade), λ -fraction of the wires in Segment 4 (output segment) and with at most λ -fraction of the memory gates, where $\lambda = \min\left\{\frac{1}{3\sigma_V}, \frac{\delta}{6}\right\}$.

We call such an adversary λ -locally tampering, and we denote the output of any such adversary by $\text{LocalTAMP}_{\mathcal{A}}(C_S^{(1)})$.

Lemma 1. Let $\lambda = \min\{\frac{1}{3\sigma_V}, \frac{\delta}{6}\}$. Then for any λ -locally tampering adversary \mathcal{A} there exists a simulator Sim , that runs in time $\text{poly}(|\mathcal{A}|, |C_s|)$, such that

$$\{\text{LocalTAMP}_{\mathcal{A}}(C_S^{(1)})\}_{k \in \mathbb{N}} \equiv \{\text{LeakBB}_{\text{Sim}, \log T}(C_s)\}_{k \in \mathbb{N}},$$

where T is an upper bound on the number of times that \mathcal{A} runs the (tampered) circuit.

We give the proof idea of Lemma 1, and defer the formal proof to the full version.

Proof idea. Fix any λ -locally tampering adversary \mathcal{A} . Let T be an upper bound on the number of times the adversary \mathcal{A} runs a possibly tampered version of the circuit $C_s^{(1)}$. We construct a simulator Sim with black-box access to C_s , which is allowed to request $\log T$ bits of leakage on the secret s , and simulates the view of \mathcal{A} .

Sim chooses the leakage function to be the function $L : \{0, 1\}^k \rightarrow \{0, 1\}^{\log T}$, which has the adversary \mathcal{A} hard-wired into it, and on input a secret s , outputs the largest index i^* , such that in the first $i^* - 1$ runs of the (possibly tampered) circuit by \mathcal{A} , all the (possibly tampered) input wires to the gate G_{cas} have the value 1.

After querying this leakage function and receiving an output i^* , Sim internally emulates \mathcal{A} , by emulating the output of each run of the circuit. Each time \mathcal{A} calls the circuit with input x_i and a set of tampering instructions, the simulator Sim simulates the run of the (possibly tampered) circuit as follows:

- Simulate the output X'_i of the (possibly tampered) Encoding Input Segment. Efficiently decode X'_i to obtain $x'_i = \text{Dec}(X'_i)$.

Note that X'_i may be far from a codeword, in which case x'_i may be \perp . Also, note that the simulator Sim is indeed able to carry out the above computation exactly as in the real computation, since this part of the computation does not use the secret s , and since Sim knows \mathcal{A} 's tampering instructions.

- If $i < i^*$, set all the input wires of G_{cas} to be 1, and set the output of the Circuit Computation Segment to be $b'_i = C_s(x'_i)$. As we prove in the full version, the fact that $i < i^*$ implies that it must be the case that $x'_i \neq \perp$, and that the output of the Circuit Computation Segment is indeed $C_s(x'_i)$.

Simulate the output b_{out} of G_{out} , using the tampering instructions of \mathcal{A} , assuming that the values of the $K + 1$ incoming wires (before applying the tampering instructions to these wires) are $(1^K, b'_i)$. Return b_{out} .

Note that it is not necessarily the case that $b_{\text{out}} = b'_i$ since the adversary \mathcal{A} may tamper with some of the incoming wires of G_{out} and may tamper with the output wire.

- If $i \geq i^*$, return $b_{\text{out}} = 0$, unless \mathcal{A} tampers with the output wire, in which case Sim returns b if the tamper is “set to b ”, and returns 1 if the tamper is “toggle”.

We will argue that in this case, even if \mathcal{A} tampers with the input wires to G_{out} , the output of G_{out} is always going to be 0, since he cannot tamper with all the input wires (recall that he can tamper with at most a λ -fraction of the wires). Thus, the output wire is always 0, unless the output wire itself is tampered with.

We defer to the full version the proof that this simulator indeed simulates the view of \mathcal{A} correctly.

4.2 The Compiler: Final Construction

Note that $C_S^{(1)}$ is secure against any adversary that tampers with only a constant fraction of the memory, and a constant fraction of the wires in Segments 2, 3 and 4. We would like to construct a circuit that is secure against adversaries that tamper with a constant fraction of the circuit overall. Namely, we would like our circuit, which will be denoted by $C_{\tilde{S}}^{(2)}$, to have the property that there is a constant $\lambda > 0$ such that $C_{\tilde{S}}^{(2)}$ is secure against any adversary that for each run of the circuit sends at most $\lambda \cdot |C_{\tilde{S}}^{(2)}|$ tampering instructions, where each tampering instruction is either a wire tampering or a memory tampering.

We do this by adding enough memory gates, and enough wires to the critical segments: the PCPP Verification Segment, the Error Cascade Segment and the Output Segment, so that the following two conditions hold.

1. Each of these segments, as well as the memory, remains resilient to a constant fraction of tampering.
2. The total number of wires in each of these segments, and the size of the memory, is a constant fraction of the total number of wires in the entire circuit.

More specifically, let $\sigma_{\text{comp}} = \sigma_{\text{comp}}(k)$ be the size of Segment 1 of the circuit $C_S^{(1)}$. We transform the circuit $C_S^{(1)}$, to ensure that the size $K = K(k)$ of the encoding of s in memory, the size $\sigma_v = \sigma_v(k)$ of Segment 2 (the PCPP Verification Segment), the size $\sigma_{\text{cas}} = \sigma_{\text{cas}}(k)$ of Segment 3 (the Error Cascade Segment), and the size $\sigma_{\text{out}} = \sigma_{\text{out}}(k)$ of Segment 4 (the Output Segment), satisfy

$$\Theta(K(k)) = \Theta(\sigma_v(k)) = \Theta(\sigma_{\text{cas}}(k)) = \Theta(\sigma_{\text{out}}(k)) = \Omega(\sigma_{\text{comp}}(k)),$$

without changing the size σ_{comp} of Segment 1, and while keeping each of the above segments (Segments 2, 3, and 4), and the memory gates, resilient to constant fraction of tampering.

This transformation will allow us to prove security against adversaries who tamper with a constant fraction of the circuit overall. We proceed with the formal construction.

Let

$$M = M(k) = \max\{\sigma_v(k), K(k), \sigma_{\text{comp}}(k)\}.$$

Let $p_1(k) \triangleq M/\sigma_v(k)$, and let $\sigma'_v(k) \triangleq p_1(k) \cdot \sigma_v(k)$. Similarly, let $p_2(k) \triangleq M(k)/K(k)$, and let $K'(k) \triangleq p_2(k) \cdot K(k)$. For the sake of simplicity we assume that p_1 and p_2 are integers.⁷ Note that under this simplifying assumption $K' = \sigma'_v = M$.

Memory: Encoding Secret s . Let $\widetilde{\text{ECC}}(s) = \text{ECC}(s)^{p_2(k)}$, where by $\text{ECC}(s)^{p_2(k)}$ we denote $p_2(k)$ concatenated copies of $\text{ECC}(s)$. Note that $\widetilde{\text{ECC}}$ is an error-correcting code which has the same distance as ECC , which is at least 2δ . (The reason the distance of ECC is at least 2δ follows from the fact that it can correct up to δ -fraction of errors.) Let $\tilde{S} = \widetilde{\text{ECC}}(s)$. We denote by $\tilde{S}(i, j)$ the j -th bit of the i -th copy of S .

⁷ This simplifying assumption makes the analysis somewhat cleaner. Without this assumption, we would need to define $p_1(k) \triangleq \lceil M/\sigma_v(k) \rceil$ and $p_2(k) \triangleq \lceil M(k)/K(k) \rceil$.

We denote by S the first row of the encoding \tilde{S} : $(\tilde{S}(1, j))_{j \in [K]}$. The encoding \tilde{S} is placed in the memory of $C^{(2)}$.

Segment 1. This segment, which includes the encoding of the input x , the circuit computation, and the PCPP computation, remains exactly the same as Segment 1 of $C_S^{(1)}$. In order to keep this segment the same, instead of using the entire new memory $\widetilde{\text{ECC}}(S)$, this segment only uses the first row of $\widetilde{\text{ECC}}(s)$, which is exactly $S = \text{ECC}(s)$, the memory content of $C^{(1)}$.

Segment 2: PCPP Verification. This stage consists of $\tau' \triangleq p_1(k) \cdot \tau$ number of circuits $\{C_{v_{i,j}}\}_{i \in [\tau], j \in [p_1(k)]}$, where each PCPP verifier from $C_S^{(1)}$ is simply copied $p_1(k)$ times. Namely, for each $i \in [\tau]$ and each $j \in [p_1(k)]$, the circuit $C_{v_{i,j}}$ is simply a copy of C_{v_i} . We note that each $C_{v_{i,j}}$ only accesses the first row of memory $S = (\tilde{S}_1, j)_{j \in [k]}$, and as before, each verifier circuit $C_{v_{i,j}}$ takes as input a constant number of bits from $(X \circ S, b, \pi)$ and outputs a single bit $\beta_{i,j}$. Note that each verifier circuit still has constant size $\tilde{\sigma}_v$.

All these τ' output wires of the circuits $C_{v_{i,j}}$ are inputs to the AND gate G_{cas} . This gate has K' additional input wires that belong to Segment 3 below (where K' is the number of bits in \tilde{S}). The gate G_{cas} has $2K'$ output wires, and we denote the values on these wires by $\{\gamma_{\ell,m}\}_{i \in [p_2(k)], j \in [2K']}$. The first K' output wires (with values $\{\gamma_{\ell,m}\}_{i \in [p_2(k)], j \in [K]}$) belong to Segment 3, and the other K' output wires belong to Segment 4.

Segment 3: Error Cascade. This segment has two parts:

- A circuit $C_{\text{code},i,j}$ of constant size σ_{code} for each bit $\tilde{S}_{i,j}$ of \tilde{S} : $1 \leq i \leq p_2(k)$, $1 \leq j \leq K$:
Input: $\tilde{S}(1, j), \tilde{S}(i, j)$.
Output: $\psi_{i,j} = \neg(\tilde{S}(1, j) \oplus \tilde{S}(i, j))$.

All these output wires with values $\psi_{i,j}$ are inputs to G_{cas} . Thus, in total, G_{cas} has $K' + \tau'$ input wires (τ' from Segment 2 and K' from Segment 3), and it outputs

$$\left(\bigwedge_{i \in [\tau], j \in [p_1(k)]} \beta_{i,j} \right) \wedge \left(\bigwedge_{i \in [p_2(k)], j \in [K]} \psi_{i,j} \right)$$

- The first K' output wires of G_{cas} , denoted by $(\gamma_{\ell,m})_{\ell \in [p_2(k)], m \in [K]}$, are fed to the memory gates. More specifically, for every $\ell \in [p_2(k)]$ and $m \in [K]$, the (ℓ, m) -th output $\gamma_{\ell,m}$ of G_{cas} is the input to memory gate (ℓ, m) . Thus, if $\gamma_{\ell,m} = 0$, memory position (ℓ, m) is overwritten with a 0. If $\gamma_{\ell,m} = 1$, memory position (ℓ, m) is unchanged.

Segment 4: The Output. This segment of the circuit consists of K' output wires of G_{cas} , which have the values $\{\gamma_{\ell,m}\}_{\ell \in [p_2(k)], m \in [K+1, 2K]}$. This segment has a single AND gate G_{out} which has fan-in $K' + 1$. This segment contains all the $K' + 1$ input wires to G_{out} : The first K' input wires are exactly the K' output wires of G_{cas} (with values $\{\gamma_{\ell,m}\}_{\ell \in [p_2(k)], m \in [K+1, 2K]}$), and the other input wire of G_{out} is an output wire of the Circuit Computation in Segment 1, which computes the bit $b = C_s(x)$.

The AND gate G_{out} has a single output wire which is

$$\tilde{b} = b \wedge \left(\bigwedge_{\ell \in [p_2(k)], m \in [K'+1, 2K']} \gamma_{\ell, m} \right).$$

The final output of the circuit $C_{\tilde{S}}^{(2)}(x)$ is \tilde{b} .

Recall that we denote by $\sigma'_v = \sigma'_v(k)$ the size of the PCPP Verification Segment in $C_{\tilde{S}}^{(2)}$, and we denote by K' the number of memory gates in $C_{\tilde{S}}^{(2)}$. We also denote by $\sigma'_{\text{cas}} = \sigma'_{\text{cas}}(k)$ the size of the Error Cascade Segment in $C_{\tilde{S}}^{(2)}$ and denote by $\sigma'_{\text{out}} = \sigma'_{\text{out}}(k)$ be the size of the Output Segment in $C_{\tilde{S}}^{(2)}$. It follows by construction that

$$\Theta(\sigma'_v(k)) = \Theta(\sigma'_{\text{cas}}(k)) = \Theta(\sigma'_{\text{out}}(k)) = \Theta(K'(k)) = \Omega(\sigma_{\text{comp}}(k)).$$

In what follows, we denote by σ' the total size of the circuit $C_{\tilde{S}}^{(2)}$. Let

$$\alpha = \alpha(k) = \min \left\{ \frac{\sigma'_v(k)}{\sigma'(k)}, \frac{\sigma'_{\text{cas}}(k)}{\sigma'(k)}, \frac{\sigma'_{\text{out}}(k)}{\sigma'(k)} \right\}.$$

Note that $\alpha(k) = \Theta(1)$.

Theorem 3. *Let $\lambda' = \min\{\alpha \frac{1}{3\bar{\sigma}_v}, \alpha \frac{\delta}{6\sigma_{\text{code}}}\}$. Then $C_{\tilde{S}}^{(2)}$ is secure against λ' -tampering adversaries (as defined in Definition 5).*

We defer the proof of Theorem 3 to the full version. We note that there we use the fact that every λ' -globally tampering adversary in $C_{\tilde{S}}^{(2)}$ is also a $\lambda = \min\{\frac{1}{3\bar{\sigma}_v}, \frac{\delta}{6\sigma_{\text{code}}}\}$ -locally tampering adversary in $C_{\tilde{S}}^{(2)}$.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Applebaum, B., Harnik, D., Ishai, Y.: Semantic security under related-key attacks and applications. Cryptology ePrint Archive, Report 2010/544 (2010), <http://eprint.iacr.org/>
3. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
4. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of proximity, shorter PCPs, and applications to coding. SIAM J. Comput. 36(4), 889–974 (2006)
5. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
6. Bitansky, N., Canetti, R., Goldwasser, S., Halevi, S., Kalai, Y.T., Rothblum, G.N.: Program Obfuscation with Leaky Hardware. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 722–739. Springer, Heidelberg (2011)

7. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. IACR Cryptology ePrint Archive, 2011:277 (2011)
9. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) IWE. In: FOCS, pp. 97–106 (2011)
10. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-Resilient Functions and All-or-Nothing Transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
11. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC, pp. 639–648 (1996)
12. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved Non-committing Encryption with Applications to Adaptively Secure Protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
13. Choi, S.G., Kiayias, A., Malkin, T.: BiTR: Built-in Tamper Resilience. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 740–758. Springer, Heidelberg (2011)
14. Dachman-Soled, D., Kalai, Y.T.: Securing circuits against constant-rate tampering (2012), http://www.cs.columbia.edu/~dglasner/MyPapers/tamper_pcp.pdf
15. Damgård, I., Nielsen, J.B.: Improved Non-committing Encryption Schemes Based on a General Complexity Assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
16. Dobrushin, R.L., Ortyukov, S.I.: Upper bound for the redundancy of self-correcting arrangements of unreliable functional elements, vol. 13, pp. 203–218 (1977)
17. Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-Key Encryption Schemes with Auxiliary Inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (2010)
18. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: STOC, pp. 621–630 (2009)
19. Dodis, Y., Pietrzak, K.: Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (2010)
20. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302 (2008)
21. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: ICS, pp. 434–452 (2010)
22. Evans, W., Schulman, L.: On the maximum tolerable noise of k-input gates for reliable computation by formulas
23. Evans, W., Schulman, L.: Signal propagation and noisy circuits. IEEE Trans. Inform. Theory 45(7), 2367–2373 (1999)
24. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-Resilient Signatures. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010)
25. Faust, S., Pietrzak, K., Venturi, D.: Tamper-Proof Circuits: How to Trade Leakage for Tamper-Resilience. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 391–402. Springer, Heidelberg (2011)
26. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
27. Feder, T.: Reliable computation by networks in the presence of noise. IEEE Trans. Inform. Theory 35(3), 569–571 (1989)
28. Gács, P., Gál, A.: Lower bounds for the complexity of reliable boolean circuits with noisy gates. IEEE Transactions on Information Theory 40(2), 579–583 (1994)

29. Gál, A., Szegedy, M.: Fault tolerant circuits and probabilistically checkable proofs. In: Structure in Complexity Theory Conference, pp. 65–73 (1995)
30. Gondolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
31. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic Tamper-Proof (ATP) Security: Theoretical Foundations for Security against Hardware Tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
32. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
33. Goldwasser, S., Rothblum, G.N.: How to compute in the presence of leakage. Electronic Colloquium on Computational Complexity (2012)
34. Goldwasser, S., Rothblum, G.N.: Securing Computation against Continuous Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010)
35. Govindavajala, S., Appel, A.W.: Using memory errors to attack a virtual machine. In: IEEE Symposium on Security and Privacy, pp. 154–165 (2003)
36. Hajek, B.E., Weller, T.: On the maximum tolerable noise for reliable computation by formulas. IEEE Transactions on Information Theory 37(2), 388–391 (1991)
37. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private Circuits II: Keeping Secrets in Tamperable Circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
38. Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
39. Juma, A., Vahlis, Y.: Protecting Cryptographic Keys against Continual Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
40. Kalai, Y.T., Kanukurthi, B., Sahai, A.: Cryptography with Tamperable and Leaky Memory. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 373–390. Springer, Heidelberg (2011)
41. Kalai, Y.T., Rao, A., Lewko, A.: Formulas resilient to short-circuit errors (2011) (manuscript)
42. Katz, J., Vaikuntanathan, V.: Signature Schemes with Bounded Leakage Resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
43. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Side Channel Cryptanalysis of Product Ciphers. In: Quisquater, J.-J., Deswart, Y., Meadows, C., Gollmann, D. (eds.) ESORICS 1998. LNCS, vol. 1485, pp. 97–110. Springer, Heidelberg (1998)
44. Kleitman, D.J., Leighton, F.T., Ma, Y.: On the design of reliable boolean circuits that contain partially unreliable gates. In: FOCS, pp. 332–346 (1994)
45. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
46. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
47. Kuhn, M.G., Anderson, R.J.: Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. In: Aucsmith, D. (ed.) IH 1998. LNCS, vol. 1525, pp. 124–142. Springer, Heidelberg (1998)
48. Liu, F.-H., Lysyanskaya, A.: Algorithmic Tamper-Proof Security under Probing Attacks. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 106–120. Springer, Heidelberg (2010)
49. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)

50. Naor, M., Segev, G.: Public-Key Cryptosystems Resilient to Key Leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
51. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
52. Pippenger, N.: Reliable computation by formulas in the presence of noise. IEEE Trans. Inform. Theory 34(2), 194–197 (1988)
53. Pippenger, N.: On networks of noisy gates. In: FOCS, pp. 30–38. IEEE (1985)
54. Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
55. Rao, J.R., Rohatgi, P.: Empowering side-channel attacks. Cryptology ePrint Archive, Report 2001/037 (2001), <http://eprint.iacr.org/>
56. von Neumann, J.: Probabilistic logics and the synthesis of reliable organisms from unreliable components (1956)

How to Compute under \mathcal{AC}^0 Leakage without Secure Hardware

Guy N. Rothblum*

Microsoft Research, Silicon Valley Campus

Abstract. We study the problem of computing securely in the presence of leakage on the computation’s internals. Our main result is a general compiler that compiles any algorithm P , viewed as a boolean circuit, into a functionally equivalent algorithm P' . The compiled P' can then be run repeatedly on adversarially chosen inputs in the presence of leakage on its internals: In each execution of P' , an \mathcal{AC}^0 adversary can (adaptively) choose any leakage function that can be computed in \mathcal{AC}^0 and has bounded output length, apply it to the values on P' ’s internal wires in that execution, and view its output. We show that no such leakage adversary can learn more than P ’s input-output behavior. In particular, the internals of P are protected.

Security does not rely on any secure hardware, and is proved under a computational intractability assumption regarding the hardness of computing inner products for \mathcal{AC}^0 circuits with pre-processing. This new assumption has connections to long-standing open problems in complexity theory.

1 Introduction

Modern cryptographic algorithms are often modeled as “black boxes”. It is commonly assumed that their keys, internal computations, and randomness are opaque to external adversaries. In practice, however, these algorithms might be run in adversarial settings where keys are compromised and computations are not be fully opaque, e.g. because of side channel attacks.

Side channel attacks exploit the physical implementation of cryptographic algorithms. The physical implementation might enable observations and measurements on the cryptographic algorithm’s internals. Attacks such as these can and have broken systems with a mathematical security proof, without violating any of the underlying mathematical principles. (see [KJJ99, RCL] for just two examples). A growing body of recent research on *cryptography resilient to leakage or side-channel attacks* aims to build robust mathematical models of realistic side channel attacks, and to develop methods grounded in modern cryptography to provably resist such attacks.

One line of research aims to construct specific primitives (e.g. encryption schemes) resilient to side-channel attacks, e.g. [AGV09, BKKV10]. A different

* Some of this research was done while the author was at the Center for Computational Intractability at Princeton University, supported by a Computing Innovation Fellowship and by NSF grant CCF-0832797.

line, which we pursue in this work, aims to construct leakage-resilience compilers for transforming general algorithms (represented as stateful circuits) into functionally equivalent stateful circuits that are resilient to side-channel attacks on any polynomial number of executions (the polynomial is not fixed in advance). Typically, these results consider a family \mathcal{L} of leakage attacks, and show (via a simulation argument) that even an adversary who can adaptively choose inputs to each execution, and launch a leakage attack from the family \mathcal{L} on each execution, learns no more about the underlying functionality than it would from black-box (i.e. input-output) access. It is usually assumed that the initial circuit compilation is done (once only) without any leakage. Afterwards, every execution of this stateful circuit—including any and all state updates—is subject to leakage attacks from the family \mathcal{L} .

A fascinating question for research on leakage-resilience compilers, from both a foundational and a practical perspective, is: “*for which leakage families \mathcal{L} do there exist secure compilers?*” There are, unfortunately, inherent limitations on the leakage functions that can be handled. For example, they must be of bounded length—otherwise the entire circuit internals can leak, and we enter the more challenging domain of code obfuscation. In particular, the impossibility results of Barak *et al.* [BGI⁺01] imply that there does not exist a leakage-resilience compiler that can protect against leakage of unbounded length. In fact, the impossibility result of [BGI⁺01] can be used to show that there is no leakage-resilience compiler that protects against even *a single bit* of polynomial-time leakage. Given this impossibility, work on leakage-resilience compilers has considered various additional restrictions on the leakage functions (on top of bounded output length):

Wire-Probe/Bit Leakage. Ishai Sahai and Wagner [ISW03] view algorithms as boolean circuits. They considered leakage functions that expose the values on a bounded number of wires in each circuit evaluation. For this class of leakage functions they show (unconditionally) a leakage-resilience compiler for general circuits. Ajtai [Ajt11] considered the RAM model. He divided each RAM computation into sub-computations, and considered leakage functions that exposed a constant fraction of the memory words involved in each sub-computation. He obtained a leakage-resilience compiler for general RAM computations. The leakage model is qualitatively similar to that of [ISW03], in the sense that the (length bounded) leakage operates separately on each bit of the computation—either exposing it in its entirety or revealing nothing at all. We view the main (and important) improvement in [Ajt11] versus [ISW03] as the quantitative improvement to fraction of leaked bits that can be tolerated in each execution of the transformed computation.

Computationally Bounded Leakage. Faust, Rabin, Reyzin, Tromer and Vaikuntanathan [FRR⁺10] considered leakage functions with two restrictions: (*i*) the functions are *computationally bounded*, e.g. capable of computing only \mathcal{AC}^0 functions of the values on the circuit’s wires,¹ and (*ii*) the computation can use *perfectly secure hardware components*, whose internals never leak. We view this second assumption as a strong restriction on the leakage functions:

¹ Their full result is actually more general, and can handle $ACC^0[p]$ or noisy leakage.

they cannot even compute a single bit of information about the internals of the secure hardware components, which are used multiple times throughout the execution. For this family of leakage functions, Faust *et al.* showed (unconditionally) a general leakage-resilience compiler for transforming any circuit.

\mathcal{AC}^0 leakage is a qualitatively richer class than wire-probe leakage. In particular, for a fixed leakage bound λ , \mathcal{AC}^0 leakage can output the values of λ wires, but potentially also much more (e.g. the AND of all circuit wire values). The results of [FRR⁺10], however, are qualitatively incomparable to [ISW03] because of the secure hardware restriction.

Only-Computation (OC) Leakage. Goldwasser and Rothblum [GR10] and Juma and Vhalis [JV10] considered leakage under the restriction that “*only computation leaks information*”, as pioneered by Micali and Reyzin [MR04]. Here, each execution of the algorithm is divided into ordered sub-computations, and the leakage function operates separately (if adaptively) on each sub-computation. Those works also assumed the existence of perfect (simple) secure hardware components, and showed general leakage-resilience compilers under different cryptographic assumptions. More recently, Goldwasser and Rothblum [GR12] showed how to remove both the use of secure hardware and the computational assumption, obtaining an unconditional compiler for protecting computations from the “only-computation leaks information” (OC) family of leakage functions.

Comparing this to the other models described above, we note that (for a fixed leakage bound) OC leakage is qualitatively richer than wire probe leakage, but incomparable to \mathcal{AC}^0 leakage.

Remark 1 (A Foundational Perspective.). While the study of leakage-resilient cryptography is motivated by real-world attacks and security considerations, it explores a foundational question: The issue at its heart is the difference between giving an adversary *black-box access to a program* and *access to the program’s code or internals*. This question is central to the foundations and the practice of cryptography. [GR12] note that the connection between obfuscation and leakage-resilience hinted at above is no accident: Obfuscation considers the task of compiling a program to make it completely unintelligible, or “impervious to all leakage” (i.e. even to an adversary with complete access to the program’s code). Unfortunately, full-blown obfuscation is provably impossible in many settings [BGI⁺01, GK05], and is considered intractable in practice. Perhaps as a result of this impossibility, much of cryptography only considers adversaries that have (at best) “black box” access to the programs under attack. Leakage-resilience compilation can be viewed as exploring the middle ground between full access to the code and black-box access: Giving the adversary *limited access* to the program’s internals and its code. Our primary motivation in this work is understanding which kinds of restricted access to code permit secure compilation (i.e. leakage resilience). On this note, an interpretation of this paper’s main result for the setting of obfuscation is provided below, following Theorem 1.

From a real-world security perspective, we note that we do not view \mathcal{AC}^0 leakage as a realistic model for *all* side-channel attacks. In fact, some common real-world side channel attacks are not covered by \mathcal{AC}^0 leakage (or any of the leakage families considered in the leakage-resilience literature). See e.g. the work

of Renauld *et al.* [RSVC⁺11]). We view this as motivation for further study of leakage-resilience compilation against more and richer classes of attacks.

$\lambda(\cdot)$ -IPPP Assumption (informal). The Inner-Product with Pre-Processing (IPPP) Problem considers predicting the inner product of two uniformly random vectors $x, y \in \{0, 1\}^\kappa$ using an \mathcal{AC}^0 circuit and an arbitrary polynomial-time pre-processing step that is *run separately* on x and on y (with polynomial output length). Without pre-processing, it is known that predicting the inner product is hard for \mathcal{AC}^0 [Raz87, Smo87]. With *joint* pre-processing on x and y , one can simply compute the inner product, and so the problem becomes easy. With (polynomial time) pre-processing that is run separately on x and on y , there is no known \mathcal{AC}^0 predictor with non-negligible advantage (we emphasize that the pre-processing step's output length can be polynomial). This question has been explicitly considered in the literature for some time (e.g. [BFS86]). We introduce the 1-IPPP Assumption, which says that even given polynomial-time pre-processing (separately on x and on y), no \mathcal{AC}^0 circuit ensemble can predict the inner product with non-negligible advantage.

More generally, we consider also the problem of *compressing* the instance size of inner product from κ to $\lambda(\kappa) < \kappa$ bits using an \mathcal{AC}^0 circuit and arbitrary polynomial-time pre-processing on x and on y separately. By “compressing”, we mean that the instance size is reduced while still maintaining noticeable statistical difference between the distribution of YES and NO instances (inner product 1 and 0 respectively), see [HN10, DI06]. Without pre-processing, Dubrov and Ishai [DI06] showed that it is hard for \mathcal{AC}^0 to compress parity instances to sub-linear length $\kappa^{1-\delta}$ (in fact this was later used in the result of [FRR⁺10]). We introduce the $\lambda(\kappa)$ -IPPP Assumption, which says that even given polynomial-time pre-processing (on x and on y , each separately), no \mathcal{AC}^0 circuit ensemble can compress the instance size of inner product to length $\lambda(\kappa)$.

See Section 2.2 and the full version for formal definitions and a further study of IPPP. We note here that (even for $\lambda(\kappa) = \kappa^{1-\delta}$), the $\lambda(\kappa)$ -IPPP Assumption might be viewed as relatively mild compared to many standard cryptographic assumptions. In particular, it may hold even if $P = NP$.

Main Result. Our main result is a leakage resilience compiler for \mathcal{AC}^0 leakage. Security relies on the $\lambda(\cdot)$ -IPPP assumption. For security parameter κ , the transformed circuit is resilient to $\lambda(\kappa)$ bits of \mathcal{AC}^0 leakage from each execution.

Theorem 1. *For any function $\lambda(\cdot) : \mathbb{N} \rightarrow \mathbb{N}$, under the $\lambda(\cdot)$ -IPPP assumption there exists a leakage resilience compiler for \mathcal{AC}^0 leakage. For security parameter $\kappa \in \mathbb{N}$, and for a poly(κ)-size input circuit C to be transformed, the adversary's runtime can be polynomial in κ , and the leakage from each execution can be any (adaptively chosen) \mathcal{AC}^0 function of length $\lambda(\kappa)$. The size of the transformed circuit is $O(|C| \cdot \kappa^3)$.*

See Definition 2 for the formal definition of a secure compiler for \mathcal{AC}^0 leakage.

We emphasize that the leakage bound in this result is equal to the “compression” factor in the IPPP assumption. In particular, assuming the $\lambda(\kappa)$ -IPPP Assumption for all sub-linear $\lambda(\kappa) = \kappa^{1-\delta}$ (we find this to be a plausible assumption), we get resilience to any sub-linear leakage amount of leakage per execution

(similar to the leakage bound in [FRR⁺10]). We remark that even a compiler that handles only a single bit of leakage from each execution is already non-trivial—in particular, since the number of executions is an unbounded polynomial, the total combined leakage from the repeated executions can be much larger than the size of the transformed circuit. We also recall that for polynomial-time leakage (i.e. leakage not restricted to \mathcal{AC}^0), it is impossible to build a leakage-resilience compiler that handles even a single bit of leakage (see above).

\mathcal{AC}^0 -Leakage Resilience and Obfuscation. Theorem 1 can also be interpreted as providing an obfuscator that is secure against \mathcal{AC}^0 -adversaries. Obfuscation is the task of “garbling” programs to make them unintelligible. Barak *et al.* [BGI⁺01] define an obfuscator as a compiler that takes an input program P and outputs a secure obfuscation P' : a different program with the same functionality as P . The security requirement is that access to (the code of) P' “leaks no more” than black-box access to P . More formally, they require that for any efficient adversary \mathcal{A} there exists a simulator \mathcal{S} , s.t. any *predicate* that \mathcal{A} can compute from the obfuscated P' , can also be computed by \mathcal{S} from black-box access to P . [BGI⁺01] show that this strong notion of obfuscation is impossible to achieve in general.

We note that one can view the predicate computed by the obfuscation adversary \mathcal{A} as a *single bit of leakage* on the internals of P' . Taking this view, Theorem 1 shows that (under the 1-IPPP Assumption) *obfuscation against bounded \mathcal{AC}^0 adversaries is possible*. In fact, obfuscation is possible even when the \mathcal{AC}^0 adversary can run the program on inputs of its choice, and observe these executions in their entirety. We note that previous works, such as [FRR⁺10], that rely on secure hardware do not provide this strong obfuscation guarantee: There is no analogue to secure hardware in the standard obfuscation setting (their work can be interpreted as providing obfuscation against an \mathcal{AC}^0 adversary who can only observe an a-priori bounded number of executions).

Theorem 1 side-steps the impossibility result of [BGI⁺01], because there the adversary needs to run computations that are as complex as the obfuscated circuit. In our construction and setting, on the other hand, the compiled circuits run computations (such as parities) that provably cannot be done in \mathcal{AC}^0 , and the adversary is bounded to \mathcal{AC}^0 computations. We find the general question of obfuscation against bounded adversaries to be a fascinating one, and note that it is closely related to leakage-resilience compilation.

Comparison to Prior Work. We now compare the result of Theorem 1 to prior works on leakage-resilience compilers (see also the discussion above on these prior works).

Wire Probe Leakage. Comparing to the work of [ISW03] on wire probe leakage and to the more recent work of Ajtai [Ajt11], the main novelty of our result is in handling the richer class of \mathcal{AC}^0 leakage. On the other hand, those results did not rely on unproven assumptions and the quantitative leakage bounds (as a fraction of the transformed computation’s size) were better.

\mathcal{AC}^0 Leakage. The most closely related work is that of Faust *et al.* [FRR⁺10], and their construction is a starting point for ours (see below). The main added

benefit of our work is in removing the secure hardware assumption (again, this can be viewed as handling a larger class of leakage functions). The main qualitative disadvantage is in the introduction of the unproved IPPP assumption. Quantitatively, the amount of leakage we can handle depends on the function $\lambda(\cdot)$ for which IPPP is hard. If we assume hardness for any sub-linear $\lambda(\cdot)$ function, we get similar leakage bounds to [FRR⁺10]. Finally, the circuit blowup of the [FRR⁺10] compiler is $O(\kappa^2)$, whereas ours is $O(\kappa^3)$.

OC Leakage. Our end result is qualitatively incomparable to Goldwasser and Rothblum [GR12], because only-computation leakage and \mathcal{AC}^0 leakage are incomparable. We do note, however, that their result is unconditional and the circuit blowup is smaller (κ^ω , where ω is the exponent for matrix multiplication, versus κ^3 in our work). Both our work and theirs tackle the challenge of leakage-resilience compilation for a rich class of leakage functions without using secure hardware. In fact, we use the “ciphertext bank” machinery introduced in [GR12] for handling this challenge.

Our high-level approach is to build on the construction of [FRR⁺10] and remove the secure hardware using the techniques of [GR10]. Unfortunately, this is far from straightforward. In a nutshell, the main technical challenge is constructing an \mathcal{AC}^0 and length-preserving security reduction from the problem of compressing inner product instances (or rather from the IPPP problem), to distinguishing real and simulated leakage on repeated executions of the transformed circuit. The problem is that the [GR10] machinery relies (extensively) on computations that cannot be performed in \mathcal{AC}^0 (e.g. matrix multiplication). We elaborate in Section 1.1.

1.1 Overview of the Construction and Security Reduction

At a high level, one central difficulty in leakage-resilience compilation without secure hardware is that it requires simulating a *complete view of multiple executions in their entirety*. The simulated view needs to be indistinguishable from the real execution under a wide family of leakage functions. Intuitively, secure hardware makes the simulation task considerably easier because some regions of the computation are opaque to the leakage, and their internals need not be simulated. Work on specific leakage-resilient cryptographic primitives (e.g. [AGV09, DP08, BKKV10]) avoids this difficulty because the security definitions do not require simulation. We follow [GR12] in tackling on this simulation challenge.

The simulator has no knowledge of the computation’s internals (beyond its input and output). Moreover, the *entire computation* in the simulated view should be consistent with the initial state and the choice of random coins; otherwise, an \mathcal{AC}^0 leakage function that checks consistency of the internal computations will distinguish the real and simulated views. This means that *the simulator has to generate a self-consistent view of the computation, and cannot make any “illegal” computational steps*. Presumably, however, the simulator is still acting very differently from the real execution. Note that this is a crucial difference from the setting where secure hardware is used: the simulated outputs of the secure

hardware can essentially be an “illegal” output that would never be generated in a real execution (but the leakage functions cannot tell the difference).² Indeed, this is what the [FRR⁺10] simulator does (see more below).

In our setting, without secure hardware, the simulator cannot make any “illegal” steps. Its only freedom to diverge from a “real” execution is in generating the initial state (where there is no leakage), and in choosing the random coins. Our simulator generates (only once) a “trapdoor” initial state, and this gives it extensive freedom in shaping the simulated view, even under repeated leakage from multiple executions and state updates.

Against this backdrop, we recall the approach of [FRR⁺10]. They proved security by showing a reduction from compressing an instance of the inner product problem (or rather the problem of computing a vector’s parity) to distinguishing the real and simulated views (or rather various hybrids of these views). They showed that the security reduction can be computed using length-bounded \mathcal{AC}^0 access to the inner product instance, and so the real and simulated views are statistically close. Our high-level approach is to build on the construction of [FRR⁺10], and remove the secure hardware using the techniques of [GR10]. As hinted above, this is far from straightforward, mainly because running the [GR10] machinery requires computations, such as matrix multiplication, that cannot be implemented in \mathcal{AC}^0 . This creates (multiple) difficulties in implementing a reduction from compressing inner product instances to distinguishing the real and simulated views (or hybrids thereof) that only uses length-bounded \mathcal{AC}^0 access to the inner product instance. We relax the requirement from the security reduction: we reduce from the IPPP problem, rather than the full-blown inner product problem without pre-processing. We use the additional power of pre-processing to implement a reduction from the IPPP problem to distinguishing (hybrids of) the real and simulated views. This is our main technical contribution.

We proceed with an overview of our construction and security proof. In what follows we restrict our attention to transforming *stateless* computations to be leakage resilient, but the results all hold also for stateful circuits (as considered say in [FRR⁺10]). We note that the transformed computations will be stateful circuits (even if the original computation is stateless), and their state is updated (under leakage) between executions.

Overview of [FRR⁺10]. In the Faust *et al.* construction, every wire i in the original circuit, say carrying value a_i (on a certain input), was replaced by a *bundle* of κ wires carrying a uniformly random vector of bits whose parity/XOR is a_i . Intuitively, an adversary with bounded-length \mathcal{AC}^0 leakage access to all of an execution’s wire-bundles cannot distinguish the true value on any wire (i.e. the parity of any wire-bundle), and so the adversary learns nothing about the internals of the original computation. The main challenge is for the transformed circuit to emulate the computation of each gate in the original circuit, while maintaining the invariant that the bundle corresponding to each wire is a uniformly random vector with the correct parity, and without leaking anything about the parity. For example, to emulate an AND gate, the transformed circuit

² This difficulty is avoided in [ISW03] as their leakage functions are too weak to check consistency of the computation.

needs to take two wire bundles and output a wire bundle carrying a uniformly random vector whose parity is the AND of the parities of the input wire bundles.

The gate computations of the original circuits are emulated using “gate gadgets”, one for every gate in the original circuit. In their elegant security proof, [FRR⁺10] separate the wires of the transformed circuit into the “external wires” described above, where each *wire* in the original circuit corresponds to the κ “*external wires*” in the transformed circuit. In each execution (on a certain input), these κ wires carry a bundle whose parity equals the wire’s value in the original circuit (on that input). Each *gate* in the original circuit is replaced by a “*gate gadget*” in the transformed circuit. We call the wires in these gate gadgets “internal wires”. In their security proof, [FRR⁺10] show that: (i) the *external wire distributions* (the distributions of values on the external wires) in the real and simulated executions are indistinguishable using bounded length \mathcal{AC}^0 leakage. As sketched above, this follows from the hardness of compressing parity instances in \mathcal{AC}^0 . Then (ii) they show that the values on the internal wires of each gate gadget can be simulated *in* \mathcal{AC}^0 given only the gate gadget’s input and output external wires. In fact, the simulation for gate gadgets is not perfect, but rather indistinguishable under bounded length \mathcal{AC}^0 leakage. The \mathcal{AC}^0 simulators for gate gadgets are called *reconstruction procedures*, and their existence guarantees that indistinguishability of the external wire distributions in the real and simulated executions implies indistinguishability of the complete view (including the internal wires of the gate gadgets).

Given this framework, the main challenge is implementing the gate gadgets and their \mathcal{AC}^0 reconstruction procedures. This is where the secure hardware devices come into play. The secure hardware outputs a uniformly random bundle of κ values with parity 0. The simulator can simulate the secure hardware’s output to be a vector with any desired parity, and the leakage cannot tell the difference. As an example of how such a device is used, consider the (relatively simple) case of addition: the gadget gets two input wire bundles, \mathbf{d}_i and \mathbf{d}_j and computes the pairwise XORs of their bits (call this bundle \mathbf{q}). Rather than simply output this \mathbf{q} (which already has the correct XOR), the gadget calls the secure hardware to compute a bundle \mathbf{o} whose XOR is 0, and finally outputs the pairwise XOR of \mathbf{q} and this “masking” bundle \mathbf{o} . This is not the only way in which the secure hardware is used (e.g. it is called more extensively for multiplication gates), but it is instructive. Intuitively, masking the gadget’s output with the output \mathbf{o} of the secure hardware helps secure simulation: the gate gadget’s reconstruction procedure has some freedom in choosing a bundle (with arbitrary XOR) as the output of the secure hardware. Another important point here is that using the secure hardware “erases” any accumulated leakage on the input bundles: the gadget’s output bundle is statistically independent (given its XOR) from the input bundles. In particular, for any given values for the gate gadget’s input and output bundles, we can (in \mathcal{AC}^0) compute an output of the secure hardware (i.e. a “secure hardware” bundle) for which the gate gadget, on the given input bundles, outputs the given output bundle.

Construction. The secure hardware in [FRR⁺10] generates a random bundle whose parity is 0, but can be simulated as having generated bundles whose parity is 0 or 1 (as needed by the simulator). An initial idea is simply to pre-compute

all the needed 0-bundles as part of the initial state (i.e. without leakage). The simulator can then choose whatever bundles it wants (0 or 1) to put in the initial state. The main drawback to this approach, of course, is that we can only pre-compute a finite and bounded number of these bundles, and so this construction cannot support repeated executions (alternatively, the initial state grows linearly with the number of executions).

This recalls the situation in [GR12]. They suggested using a “ciphertext bank”. Translating that idea to our setting, our construction can use a small number of pre-computed 0-bundles, a “*bundle bank*”, to generate an essentially unbounded (polynomial) number of new 0-bundles. The simulator can generate an (illegally formed) initial bundle bank, and then control each subsequent generation arbitrarily to create a 0-bundle or a 1-bundle. All of this is done in a way that is indistinguishable, even under repeated \mathcal{AC}^0 leakage, from the real execution.

We implement the bundle banks as follows. Recall that each bundle encodes a bit $b \in \{0, 1\}$ as a vector in $\{0, 1\}^\kappa$ with parity b . The initial bundle bank includes 2κ such bundles, whose parities (in the real execution) are all 0. Within each execution of the circuit we generate 0-bundles as needed by taking random linear combinations of the bundle bank (a random linear combination of vectors whose parities are all 0 also has parity 0). Between executions we “regenerate” or update the entire bundle bank by taking 2κ new random linear combinations, and then we erase the old bundle bank. In the simulation, the initial bundle bank is generated so that each bundle is a uniformly random vector encoding a uniformly random bit. Now when we generate a new bundle, some linear combinations of the bundles in the bank give a new bundle encoding 0, and some give a new bundle encoding 1. The simulator chooses a uniformly random linear combination yielding an encoding of whatever value it wants. Between executions, as in the real view, the bundle bank is refreshed by taking 2κ new random linear combination, giving a new bank of uniformly random bit encodings (independent of the old bank). The full construction is in Section 3.

We note that our implementation of the bundle bank is considerably simpler than the ciphertext banks of [GR12]. In particular, there is no need for their “piecemeal matrix multiplication” procedure, and we compute matrix multiplication using the straightforward naive procedure (in time κ^3). We also remark that proving the security properties of the bundle bank in our setting requires completely different arguments (due to the completely different nature of the leakage attack).

Security. The intuition for indistinguishability of the real and simulated views is that an adversary cannot distinguish (under bounded length \mathcal{AC}^0 leakage) whether the bundles in the bank encode 0’s (real execution) or are uniformly random (simulated execution). Nor can the adversary distinguish whether the linear combinations are uniformly random (real execution) or chosen from a $(\kappa - 1)$ -dimensional subspace so as to fix the value of the resulting bundle to 0 or 1 (simulation).

Transforming this intuition into a reduction from compressing parity/inner-product instances to distinguishing the real and simulated views, however, is

far from straightforward. The major source of difficulty is that neither the real construction nor the simulator are actually \mathcal{AC}^0 algorithms, and neither are the computations in the “gate gadgets” used to emulate each gate in the original circuit.³ In particular, if we want to use the bundle bank machinery, the gate gadgets need to compute linear combinations of bundles in the bank. Moreover, this difficulty is compounded by the fact that, unlike [FRR⁺10], we cannot use leakage-free hardware to make the bundles used between gates and across executions statistically independent of each other. Even within a single circuit execution, this is already a serious concern. Each bundle is dependant on the bank, and through it on all other bundles. If (as seems natural) we want to use hybrid arguments to focus on differences in the distribution of a single bundle or gate emulation, we need to generate the rest of the view (on which both hybrids agree). This generation, however, is both not in \mathcal{AC}^0 and is not independent of the hybrid bundle. We now give intuition for how these two difficulties are overcome, further details are in Section 3.

Step I: \mathcal{AC}^0 Reconstruction Via “Beefed Up” External Wire Distributions. Our first step towards resolving these difficulties is to add more information to the *external wire distribution* so that we can reconstruct the internal view of each gate-gadget in \mathcal{AC}^0 . For example, adapting the addition gate gadget of [FRR⁺10] to our setting, we take the bundle bank to be a matrix $G \in \{0, 1\}^{\kappa \times 2\kappa}$. Our addition gate gadget takes as input two bundles \mathbf{d}_i and \mathbf{d}_j , generates a “masking” 0-bundle $\mathbf{o} = G \times \mathbf{r}$ using the bundle bank and a random linear combination $\mathbf{r} \in_R \{0, 1\}^{2\kappa}$. It then computes an output bundle $\mathbf{d}_k = ((\mathbf{d}_i + \mathbf{d}_j) + \mathbf{o})$. The reconstruction procedure gets the input and output bundles $\mathbf{d}_i, \mathbf{d}_j, \mathbf{d}_k$, as well as the bundle bank G (a $\kappa \times 2\kappa$ matrix), and needs to generate the internal view of the computation. This requires finding a vector \mathbf{r} s.t. $G \times \mathbf{r} = (\mathbf{d}_k - (\mathbf{d}_i + \mathbf{d}_j))$ and generating the wire values of the matrix-vector multiplication $G \times \mathbf{r}$. For arbitrary $\mathbf{d}_i, \mathbf{d}_j, \mathbf{d}_k$ and G this cannot be done in \mathcal{AC}^0 . To resolve this, we give the reconstruction some additional “advice”; We “beef up” the external wire distribution with vectors $\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k$ s.t. $\mathbf{d}_i = G \times \mathbf{r}_i$, $\mathbf{d}_j = G \times \mathbf{r}_j$, and $\mathbf{d}_k = G \times \mathbf{r}_k$, and with all the wire values for computing these matrix-vector multiplications. The reconstruction procedure (for addition gates) can now compute in \mathcal{AC}^0 the vector $\mathbf{r}_o = (\mathbf{r}_k - (\mathbf{r}_i - \mathbf{r}_j))$ and (by linearity of matrix multiplication) the wire values for the matrix-vector multiplication $\mathbf{o} = G \times \mathbf{r}_o$ to output a consistent view of the addition gate gadget’s internal wires. We show that when the external wire distribution is distributed as in the real or simulated execution, our reconstruction procedures generate an indistinguishable view for the internal wires of each gate gadget. We note that reconstructing the multiplication gadget is significantly more complicated, and requires further “beefing up” of the external wire distribution.

³ We note that a similar difficulty also arose in [FRR⁺10], where in the real view (but not the simulated one) the emulation of multiplication gates required computing parities. In their work this difficulty was solved using the secure hardware (essentially replacing each real emulation of a multiplication gate with an indistinguishable emulation that could be computed in \mathcal{AC}^0). We, on the other hand, want to avoid the use of secure hardware.

Step II: Indistinguishability of “Beefed Up” External Wire Distributions. Given the \mathcal{AC}^0 reconstruction procedures, it remains to argue that the “beefed up” external wire distributions of the real and simulated executions are indistinguishable under bounded-length \mathcal{AC}^0 leakage. The external wire distribution now includes a lot of information about the bundles, and in particular the bundles are no longer independent of each other because they are tied together via the bundle bank G and, for each bundle \mathbf{d} , the vector \mathbf{r} for which $\mathbf{d} = G \times \mathbf{r}$. We will argue indistinguishability using a hybrid argument, replacing the bundle bank from real (i.e. 0 parities) to simulated (i.e. uniform), and replacing the parities of bundles on the external wires one-by-one from real to simulated values. As noted above, using a hybrid argument over the bundles one-by-one creates a challenge because the bundles (which all depend on the same bundle bank) are not independent of each other.

To use a hybrid argument over bundles, we *decompose* the execution’s view into two parts: the first part depends on the bundle bank, *but not on the hybrid bundle*, and contains essentially all the information needed to generate the parts of the computation that do not involve the hybrid bundle. The second part depends only on the randomness used to form the hybrid bundle. I.e., if \mathbf{d} is the hybrid bundle then we take \mathbf{r} to be the linear combination of the bundle bank that yields \mathbf{d} . We use \mathbf{r} to pre-compute non- \mathcal{AC}^0 information that helps in generating the part of the view involving the hybrid bundle. The key point is that we give *an \mathcal{AC}^0 procedure* for combining these two separate parts and generating the entire view of one of the hybrid distributions. Of the two hybrids, which one is generated is determined by the inner product of \mathbf{r} with a vector \mathbf{x} that depends only on the bundle bank. Now, by the IPPP assumption (assuming we can use \mathbf{x} to generate the bundle bank), we know that even given these two “pre-processed” parts of the view (computed from \mathbf{x} and \mathbf{r} separately), we can combine them in \mathcal{AC}^0 to get one of the hybrid external wire distributions, and thus no \mathcal{AC}^0 leakage on the hybrid can be statistically correlated with the inner product of \mathbf{x} and \mathbf{r} . Thus, under the IPPP Assumption, \mathcal{AC}^0 leakage should not be able to distinguish the hybrids.

In summary, we construct (hybrids of) the real and simulated external wire distributions using “pre-processing”, where two pre-computed pieces are combined in \mathcal{AC}^0 to give the appropriate external wire distribution. We get a reduction from the IPPP Problem to distinguishing the real and simulated external wire distributions.

2 Model and Definitions

Preliminaries. For a vector or string x we denote by $|x|$ the length of the vector, and by x_i or $x[i]$ the i ’th item in the vector. We denote by $x|_i$ the restriction of a vector x to its first i items. For a vector x in $\{0,1\}^n$ we say that x is an *encoding* of $b \in \{0,1\}$ if the XOR (or sum over $\mathbb{GF}[2]$) of x ’s entries equals b . For vectors $\mathbf{x}, \mathbf{y} \in \{0,1\}^\kappa$, we use $\mathbf{x} + \mathbf{y}$ to denote bitwise addition over $\mathbb{GF}[2]$ (i.e. XOR) of the vectors’ entries. We use $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n$ to denote the unit vectors over $\{0,1\}^n$ (n will be clear from the context), and we use U_n to denote the uniform distribution over $\{0,1\}^n$. For a finite set S we denote by $y \in_R S$

that y is a uniformly distributed sample from S . For a distribution D we use $y \sim D$ to denote the experiment of sampling y by D . We use $\Delta(D, F)$ to denote the statistical (L_1) distance between distributions D and F . For a circuit (or function) C and an oracle PPTM M we use $M^C(x)$ to denote M run on input x with oracle access to C .

2.1 Leakage Attacks and Security

Definition 1 (\mathcal{AC}^0 -Leakage Attack $\mathcal{A}^\lambda[C, \text{Update}](1^\kappa)$). A continual λ -bit \mathcal{AC}^0 -leakage attack of adversary \mathcal{A} on C with update procedure *Update* proceeds in rounds. In each round $t = 1, 2, \dots$, there is a circuit C_t computed in the previous round (where $C_1 = C$). The \mathcal{AC}^0 adversary \mathcal{A} chooses an input x_t for C_t and an \mathcal{AC}^0 and λ -bit output leakage function $\ell_t(\cdot)$, which takes as input: (i) the entire computation of C_t on input x_t (including all circuit wire values and all random coins), (ii) the entire computation of the update procedure *Update* run on C_t and outputting C_{t+1} (including all circuit wire values and all random coins of *Update*). The adversary's view in round t is $\text{view}_t = (x_t, C_t(x_t), \ell_t(\text{entire computation of } C(x) \text{ and } \text{Update}(C_t)))$. The adversary's choices of the input x_t and the leakage ℓ_t are adaptive can depend on the views in all previous rounds.

The attack proceeds for T rounds (where T is chosen by the adversary). The attack's output (or adversary view in the attack) is $\text{view} = (\text{view}_1, \text{view}_2, \dots, \text{view}_T)$. The running time of \mathcal{A} is its total running time in the attack, i.e. a polynomial-time adversary can only run for $\text{poly}(\kappa)$ rounds.

Remark 2. Throughout this work when any of our algorithms compute matrix multiplication (or matrix-vector/vector-vector multiplication), this is done in the straightforward (if inefficient) way. The *partial sums* are the sub-computations in the multiplication, e.g. in computing the inner product $\langle x, y \rangle$, the partial sums are $(\langle x|_1, y|_1 \rangle, \langle x|_2, y|_2 \rangle, \dots, \langle x, y \rangle)$. The leakage on the computation takes all of these partial sums as a part of its input.

Definition 2 ($\lambda(\cdot)$ -Secure Compiler for \mathcal{AC}^0 leakage). Let *Init* and *Update* be two PPTMs that take as input a circuit C and security parameter κ . We say that $(\text{Init}, \text{Update})$ is a $\lambda(\cdot)$ -secure compiler for \mathcal{AC}^0 leakage, for any circuit C the following two requirements hold:

- *Functionality:* the circuits $C_1 = \text{Init}(C, \kappa), C_2 = \text{Update}(C_1, \kappa), C_3 = \text{Update}(C_2, \kappa), \dots$ are each with all but negligible probability functionally equivalent to the original circuit C . For all $i \geq 1$ the circuits C_i are of the same size.
- *Security:* for any PPTM adversary \mathcal{A} there exists a PPTM simulator \mathcal{S} such that the view $\mathcal{A}^{\lambda(\kappa)}[\text{Init}(C, \kappa), \text{Update}(\cdot, \kappa)](1^\kappa)$ of the adversary in an \mathcal{AC}^0 leakage attack is indistinguishable from the view $\mathcal{S}^C(1^\kappa)$ generated by the simulator from black-box access to C . Note that there is no leakage on the *Init* procedure.

Remark 3. Note we may consider many relaxations and strengthenings of this definition. For example, we could relax functionality to require only that each C_i

is functionally equivalent to C w.h.p. on each input over the coins of the *Init* and *Update* procedures and/or the coins of C_i . We could also strengthen security to restrict the simulator \mathcal{S} to only have oracle access to the inputs queried by the adversary (and in fact our construction meets this more stringent requirement). The choices in the definition above were made mainly for the sake of simplicity.

2.2 The IPPP Problem and Assumption

We give formal definitions of the IPPP Problem and Assumption, see also the discussion in the introduction. See the full version for further details and discussion, including connections to problems in complexity theory, as well as further properties such as a worst-case to average-case hardness reduction.

Problem 1 (($\lambda(\cdot), s(\cdot), \varepsilon(\cdot)$)-Inner Product with Pre-Processing (IPPP)). A triplet $(\mathcal{C}, \mathcal{C}^1, \mathcal{C}^2)$ of circuits ensembles solves the $(\lambda(\cdot), s(\cdot), \varepsilon(\cdot))$ -Inner Product with Pre-Processing Problem (IPPP) if:

1. there exists a polynomial $p(\cdot) : \mathbb{N} \rightarrow \mathbb{N}$, such that $\forall \kappa \in \mathbb{N}$:
 - (a) \mathcal{C}_κ^1 and \mathcal{C}_κ^2 are of size at most $p(\kappa)$, with input length κ (and output length at most $p(\kappa)$). These are both randomized circuits with a common random input. We call \mathcal{C}^1 and \mathcal{C}^2 the *pre-processing circuits ensembles* (or circuits for short)
 - (b) \mathcal{C}_κ is of size at most $s(\kappa)$, with input length at most $p(\kappa)$ and output length $\lambda(\kappa)$.
We call \mathcal{C} the *output circuit ensemble* (or circuit for short).
 - (c) the combined output lengths of \mathcal{C}_κ^1 and \mathcal{C}_κ^2 equal the input length of \mathcal{C}_κ
2. for infinitely many $\kappa \in \mathbb{N}$:

$$\Delta(\{C_\kappa(C_\kappa^1(x), C_\kappa^2(y)) : x, y \in_R \{0, 1\}^\kappa \text{ s.t. } \langle x, y \rangle = 0\}, \{C_\kappa(C_\kappa^1(x), C_\kappa^2(y)) : x, y \in_R \{0, 1\}^\kappa \text{ s.t. } \langle x, y \rangle = 1\}) \geq \varepsilon(\kappa)$$

randomness in both distributions is over the choice of x, y and the (shared) coins of $\mathcal{C}^1, \mathcal{C}^2$.

Assumption 2 ($\lambda(\cdot)$ -IPPP Assumption) *For any polynomial $s(\cdot)$ and inverse polynomial $\varepsilon(\cdot)$, no triplet of circuit ensembles $(\mathcal{C}, \mathcal{C}^1, \mathcal{C}^2)$ with \mathcal{C} in \mathcal{AC}^0 can solve the $(\lambda(\cdot), s(\cdot), \varepsilon(\cdot))$ -IPPP problem.*

3 Transformation against \mathcal{AC}^0 Leakage

In this section we detail a secure compiler for \mathcal{AC}^0 leakage and give more details on the proof of Theorem 1. For ease of exposition we consider throughout this section a circuit $C(\cdot, \cdot)$ that is known to the adversary and takes two inputs x and y . The input x is the one chosen adaptively by the adversary in each round, whereas the input y is *secret* and protected by the compiler. In particular this gives a compiler a la Definition 2 by viewing C as a universal circuit and y as

the description of a particular circuit to be compiler. This is similar to what is done in the secure multi-party computation literature.

The compiler transforms y into a secret state for an emulation of the universal computation. This secret state includes a *wire bundle* for each y -input wire of the circuit. The XOR of this bundle is the value of its y -input wire. The secret state also includes a “bundle bank” of bundles encoding 0 (see the overview in the introduction). Given bundles for the input wires, the transformed circuit’s computation then proceeds gate by gate, using the bundles computed for that gate’s input wires and the bundles in the bank to compute an output bundle. In the construction below, we present *Init* and *Update* as procedures for initializing and updating the secret state — a collection Y (viewed as a matrix whose columns are the bundles) of bundles for the y input wires, and the bundle bank G (also a matrix). We call the “bundle bank” in round t the “*generating matrix*” for that round (as it is used to generate fresh 0-bundles).

In Figure 3 we present the procedures for emulating the computation of each gate in the original circuit. Given this view of the compiler’s operation (slightly modified from the one in Definition 2), we view the *Init* and *Update* procedure’s outputs as secret states that are later plugged into the gate-by-gate emulator of (public) circuit’s computation. They are in Figures 1 and 2. The secret states can be transformed into a full-blown circuit a la Definition 2 by augmenting them with the (publicly and known to all parties) emulation of the circuit C .

Initialization $Init(1^\kappa, y)$

1. for every input wire i , corresponding to bit j of the input y , generate a new encoding: $d_i \in \{0, 1\}^\kappa$, a uniformly random vector whose XOR is y_j .
Let Y be the matrix whose columns are these encodings.
2. generate a new uniformly random $\kappa \times 2\kappa$ matrix G whose columns are all encodings of 0.
3. output $state_1 \leftarrow (Y, G)$.

Fig. 1. *Init* procedure, to be run in an offline stage on circuit C and secret y

State Update $Update(1^\kappa, state_t = (Y, G))$

1. for each column Y_i of Y : $Y'_i \leftarrow Y_i + G \times r$, where $r \sim U_{2\kappa}$
2. pick a uniformly random $2\kappa \times 2\kappa$ matrix R . $G' \leftarrow G \times R$.
3. output $state_{t+1} \leftarrow (Y', G')$.

Fig. 2. *Update* procedure, to be run under leakage between evaluations

Security Proof: Further Details and Organization. The simulator is specified in Figure 4. We want to prove that the view it generates is indistinguishable from the real view, under bounded length \mathcal{AC}^0 leakage in each round. See the

Gate Computations (with bundle-bank/generating matrix G)

Addition ($\mathbf{d}_i, \mathbf{d}_j$):

1. $\mathbf{q} \leftarrow \mathbf{d}_i + \mathbf{d}_j$
2. $\mathbf{o} \leftarrow G \times \mathbf{r}$, where $\mathbf{r} \sim U_{2\kappa}$
3. output $\mathbf{d}_k \leftarrow \mathbf{q} + \mathbf{o}$

Multiplication ($\mathbf{d}_i, \mathbf{d}_j$):

1. $B \leftarrow \mathbf{d}_i \times \mathbf{d}_j^T$,
i.e. the $\kappa \times \kappa$ matrix where entry $[\ell, m]$ equals $\mathbf{d}_i[\ell] \cdot \mathbf{d}_j[m]$
2. $S \leftarrow G \times R$,
where R is a $\{0, 1\}$ uniformly random $2\kappa \times \kappa$ matrix
3. $U \leftarrow B + S$
4. for each row ℓ of U , compute $\mathbf{q}[\ell] = \bigoplus_{m \in [\kappa]} U[\ell, m]$
5. output $\mathbf{d}_k \leftarrow \mathbf{q}$

Constant Gate (b_i):

1. $\mathbf{q} \leftarrow [b_i, 0, 0, \dots, 0]$
2. $\mathbf{o} \leftarrow G \times \mathbf{r}$, where $\mathbf{r} \sim U_{2\kappa}$
3. output $\mathbf{d}_k \leftarrow \mathbf{q} + \mathbf{o}$

Duplication (\mathbf{d}_i):

1. $\mathbf{o} \leftarrow G \times \mathbf{r}$, where $\mathbf{r} \sim U_{2\kappa}$
2. $\mathbf{o}' \leftarrow G \times \mathbf{r}'$, where $\mathbf{r}' \sim U_{2\kappa}$
3. output $\mathbf{d}_j \leftarrow \mathbf{d}_i + \mathbf{o}$ and $\mathbf{d}_k \leftarrow \mathbf{d}_i + \mathbf{o}'$

Output Gate (\mathbf{d}_{output}):

1. $\mathbf{o} \leftarrow G \times \mathbf{r}$, where $\mathbf{r} \sim U_{2\kappa}$
2. $\mathbf{d} \leftarrow \mathbf{d}_{output} + \mathbf{o}$
3. output $\bigoplus_{m \in [\kappa]} \mathbf{d}[m]$

Fig. 3. Gate computations during evaluation, all run under leakage

introduction for the high-level intuition behind the security proof. The formal argument used in the reduction is more involved. We consider the real and simulated views, *Real* and *Simulated* (respectively). We want to use a distinguisher between the real and simulated view to build an \mathcal{AC}^0 circuit for solving the IPPP problem, leading to a contradiction. This requires care, and in particular we will use several hybrid views and seek methods for generating them in \mathcal{AC}^0 (with some pre-processing, see below).

One important difference between the *Real* and *Simulated* views is in the generating matrices they use (whose columns all encode 0 in *Real*, but are uniformly random in *Simulated*). In both views we want to use each generating matrix G to generate encodings of 0's (or 1's in *Simulated*) upon request. The immediate way of doing this is choosing randomness \mathbf{r} from the proper distribution and

Simulator

Initialize G_0 as a uniformly random $\kappa \times 2\kappa$ matrix, and Y_0 as a uniformly random $\kappa \times n$ matrix (where n is the bit length of y).

For rounds $t \leftarrow 1, 2, \dots$, on input x_t , generate the view for that round gate by gate:

1. For addition, multiplication, constant and duplication gates, operate exactly as the real gate computations in Figure 3 (albeit here G is uniformly random).
2. For the output gate, on encoding \mathbf{d}_{output} , retrieve the output bit $C(x_t, y)$. Generate \mathbf{o} as a uniform linear combination of the columns of G s.t. $\bigoplus_{m \in [\kappa]} (\mathbf{d}_{output} + \mathbf{o})[m] = C(x_t, y)$.
3. For the state update, operate as in the real state update in Figure 2. I.e. generate Y_{t+1} by adding random linear combinations of G_t to the columns of Y_t . To generate G_{t+1} , multiply G_t by a random invertible $2\kappa \times 2\kappa$ matrix.

Fig. 4. Simulator \mathcal{S}

computing $G \times \mathbf{r}$ together with all of the partial sums (the partial sums are needed for generating the entire view of each wire in the matrix-vector multiplication). Unfortunately, this is not an \mathcal{AC}^0 computation. See the discussion in Section 1.1 for intuition regarding this obstacle and the high-level ideas for overcoming it.

We will use several hybrid views, and set up the views (real, simulated or hybrid) as follows. For every round t we will have a generating matrix G_t (either uniformly random, or random s.t. all columns encode 0). We generate an *external wire distribution* which specifies the encoding/bundle on each of the circuit wires together with additional information about it and global information about this round. This recalls the high-level structure of the security proof in [FRR⁺10], though here even the *simulated* view cannot be generated in \mathcal{AC}^0 and so we need a “beefed up” external wire distribution (as discussed in Section 1.1). In particular, for each wire encoding we also include its decomposition into a linear combination of G ’s columns (the bundle bank) XORed with a 0 or a 1 bit. See the full version for a full specification of the external wire distribution.

We argue that distinguishing the external wire distributions for any pair of adjacent views (simulated, hybrid, real) is hard given only bounded length \mathcal{AC}^0 leakage from each round (this usually involves another hybrid argument over the rounds and/or wires).

To finish the argument and show the complete views in their entirety are also indistinguishable we need to generate the actual views, including also the internal gate wires. As discussed in Section 1.1, we give *reconstruction procedures* for completing the view and setting values for all of the internal gate wires in both the *Real* and *Simulated* views. The reconstruction procedures are in \mathcal{AC}^0 , and so indistinguishability of the views follows immediately from the indistinguishability of their external wire distributions.

The full security proof has been omitted for lack of space. See the full version for details.

Acknowledgements. Much of this research is joint work with Shafi Goldwasser. I am grateful for her countless insights, suggestions and contributions. Thanks also to Vinod Vaikuntanathan for suggesting that the main result implies obfuscation secure against \mathcal{AC}^0 adversaries.

References

- [AGV09] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
- [Ajt11] Ajtai, M.: Secure computation with information leaking to an adversary. In: STOC, pp. 715–724 (2011)
- [BFS86] Babai, L., Frankl, P., Simon, J.: Complexity classes in communication complexity theory (preliminary version). In: FOCS, pp. 337–347 (1986)
- [BGI⁺01] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (Im)possibility of Obfuscating Programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
- [BKKV10] Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: FOCS, pp. 501–510 (2010)
- [DI06] Dubrov, B., Ishai, Y.: On the randomness complexity of efficient sampling. In: STOC, pp. 711–720 (2006)
- [DP08] Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302 (2008)
- [FRR⁺10] Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
- [GK05] Goldwasser, S., Kalai, Y.T.: On the impossibility of obfuscation with auxiliary input. In: FOCS, pp. 553–562 (2005)
- [GR10] Goldwasser, S., Rothblum, G.N.: Securing Computation against Continuous Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010)
- [GR12] Goldwasser, S., Rothblum, G.N.: How to compute in the presence of leakage. Electronic Colloquium on Computational Complexity (ECCC) (010) (2012)
- [HN10] Harnik, D., Naor, M.: On the compressibility of np instances and cryptographic applications. SIAM J. Comput. 39(5), 1667–1713 (2010)
- [ISW03] Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
- [JV10] Juma, A., Vahlis, Y.: Protecting Cryptographic Keys against Continual Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
- [KJJ99] Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
- [MR04] Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)

- [Raz87] Razborov, A.: Lower bounds for the size of circuits of bounded depth with basis and, xor. Math. Notes of the Academy of Science of the USSR 41 (1987)
- [RCL] Boston University Reliable Computing Laboratory. Side channel attacks database, <http://www.sidechannelattacks.com>
- [RSVC⁺11] Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N., Kamel, D., Flandre, D.: A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 109–128. Springer, Heidelberg (2011)
- [Smo87] Smolensky, R.: Algebraic methods in the theory of lower bounds for boolean circuit complexity. In: STOC, pp. 77–82 (1987)

Recent Advances and Existing Research Questions in Platform Security

Ernie Brickell

Chief Security Architect for Intel Corporation, USA

Abstract. In this talk I will provide a description of recent uses Intel has made of cryptography in our platforms, including providing a hardware random number generator, using anonymous signatures, and improving performance of cryptographic algorithms. I will discuss how processor capabilities could be used more effectively by cryptographic algorithms. I will then discuss research questions in cryptographic protocols and platform security that are motivated by our goals.

Group Signatures with Almost-for-Free Revocation

Benoît Libert^{1,*}, Thomas Peters^{1,**}, and Moti Yung²

¹ Université Catholique de Louvain, ICTEAM Institute, Belgium

² Google Inc. and Columbia University, USA

Abstract. Group signatures are a central cryptographic primitive where users can anonymously and accountably sign messages in the name of a group they belong to. Several efficient constructions with security proofs in the standard model (*i.e.*, without the random oracle idealization) appeared in the recent years. However, like standard PKIs, group signatures need an efficient revocation system to be practical. Despite years of research, membership revocation remains a non-trivial problem: many existing solutions do not scale well due to either high overhead or constraining operational requirements (like the need for all users to update their keys after each revocation). Only recently, Libert, Peters and Yung (Eurocrypt'12) suggested a new scalable revocation method, based on the Naor-Naor-Lotspeich (NNL) broadcast encryption framework, that interacts nicely with techniques for building group signatures in the standard model. While promising, their mechanism introduces important storage requirements at group members. Namely, membership certificates, which used to have constant size in existing standard model constructions, now have polylog size in the maximal cardinality of the group (NNL, after all, is a tree-based technique and such dependency is naturally expected). In this paper we show how to obtain private keys of *constant* size. To this end, we introduce a new technique to leverage the NNL subset cover framework in the context of group signatures but, perhaps surprisingly, without logarithmic relationship between the size of private keys and the group cardinality. Namely, we provide a way for users to efficiently prove their membership of one of the generic subsets in the NNL subset cover framework. This technique makes our revocable group signatures competitive with ordinary group signatures (*i.e.*, without revocation) in the standard model. Moreover, unrevoked members (as in PKIs) still do not need to update their keys at each revocation.

1 Introduction

Group signatures, as suggested by Chaum and van Heyst [29], allow members of a group managed by some authority to sign messages in the name of the group

* This author acknowledges the Belgian Fund for Scientific Research (F.R.S.-F.N.R.S.) for his “Collaborateur scientifique” fellowship.

** Supported by the IUAP B-Crypt Project and the Walloon Region Camus Project.

while hiding their identity. At the same time, a tracing authority can identify the signer if necessary. A crucial problem is the revocation of the anonymous signing capability of users who leave (or are banned from) the group.

1.1 Related Work

ORDINARY GROUP SIGNATURES. The first efficient and provably coalition-resistant group signature dates back to the work of Ateniese, Camenisch, Joye and Tsudik [6]. By the time their scheme appeared, the security of the primitive was not appropriately formalized yet. Suitable security definitions remained lacking until the work of Bellare, Micciancio and Warinschi [8] (BMW) who captured all the requirements of group signatures in three properties. In (a variant of) this model, Boneh, Boyen and Shacham [14] obtained very short signatures using the random oracle methodology [9].

The BMW model assumes static groups where no new member can be introduced after the setup phase. The setting of dynamically changing groups was analyzed later on by Bellare-Shi-Zhang [10] and, independently, by Kiayias and Yung [40]. In the models of [10,40], constructions featuring relatively short signatures were proposed in [49,30]. A construction in the standard model was also suggested by Ateniese *et al.* [5] under interactive assumptions. At the same time, Boyen and Waters gave a different solution [18] without random oracles using more standard assumptions. By improving upon their own scheme, they managed [19] to obtain signatures of constant size. Their constructions [18,19] were both presented in the BMW model [8] and provide anonymity in the absence of signature opening oracle. In the dynamic model [10], Groth [34] showed a system in the standard model with $O(1)$ -size signatures but, due to very large hidden constants, his scheme was mostly a feasibility result. Later on, Groth came up with an efficient realization [35] (and signatures of about 50 group elements) with the strongest anonymity level.

REVOCATION. As in ordinary PKIs, where certificate revocation is a critical issue, membership revocation is a complex problem that has been extensively studied [20,7,26,17] in the last decade. Generating a new group public key and distributing new signing keys to unrevoked members is a simple solution. In large groups, it is impractical to update the public key and provide members with new keys after they joined the group. Bresson and Stern suggested a different approach [20] consisting of having the signer prove that his membership certificate does not belong to a list of revoked certificates. Unfortunately, the length of signatures grows with the number of revoked members. In forward-secure group signatures, Song [50] chose a different way to handle revocation but verification takes linear time in the number of excluded users.

Camenisch and Lysyanskaya [26] proposed an elegant method using accumulators¹ [11]. Their technique, also used in [52,24], allows revoking members while

¹ An accumulator is a kind of “hash” function mapping a set of values to a short, constant-size string while allowing to efficiently prove that a specific value was accumulated.

keeping $O(1)$ costs for signing and verifying. The downside of this approach is its history-dependence: it requires users to follow the dynamic evolution of the group and keep track of all changes: each revocation incurs a modification of the accumulator value, so that unrevoked users have to upgrade their membership certificate before signing new messages. In the worst case, this may require up to $O(r)$ exponentiations, if r is the number of revoked users.

Another drawback of accumulator-based approaches is their limited applicability in the standard model. Indeed, for compatibility reasons with the central tool of Groth-Sahai proofs, pairing-based accumulators are the only suitable candidates. However, in known pairing-based accumulators [48,24], public keys have linear size in the maximal number of accumulations, which would result in linear-size group public keys in immediate implementations. To address this concern in delegatable anonymous credentials, Acar and Nguyen [4] chose to sacrifice the constant size of proofs of non-membership but, in group signatures, this would prevent signatures from having constant size. Boneh, Boyen and Shacham [14] managed to avoid linear dependencies in a revocation mechanism along the lines of [26]. Unfortunately, their technique does not seem to readily interact² with Groth-Sahai proofs [36] so as to work in the standard model.

In [21], Brickell considered the notion of *verifier-local revocation* group signatures, for which formal definitions were given by Boneh and Shacham [17] and other extensions were proposed in [46,53,42]. In this approach, revocation messages are only sent to verifiers and the signing algorithm is completely independent of the number of revocations. Verifiers take as additional input a revocation list (RL), maintained by the group manager, and have to perform a revocation test for each RL entry in order to be convinced that signatures were not issued by a revoked member (a similar revocation mechanism is used in [22]). The verification cost is thus inevitably linear in the number of expelled users.

In 2009, Nakanishi, Fuji, Hira and Funabiki [45] came up with a revocable group signature with constant complexities for signing/verifying. At the same time, group members never have to update their keys. On the other hand, their proposal suffers from linear-size group public keys in the maximal number N of users, although a variant reduces the group public key size to $O(N^{1/2})$.

In anonymous credentials, Tsang *et al.* [51] showed how to blacklist users without compromising their anonymity or involving a trusted third party. Their schemes either rely on accumulators (which may be problematic in our setting) or have linear proving complexity in the number of revocations. Camenisch, Kohlweiss and Soriente [25] dealt with revocations in anonymous credentials by periodically updating users credentials in which a specific attribute indicates a validity period. In group signatures, their technique would place an important

² In [14], signing keys consist of pairs $(g^{1/(\omega+s)}, s) \in \mathbb{G} \times \mathbb{Z}_p$, where $\omega \in \mathbb{Z}_p$ is the secret key of the group manager, and the revocation method relies on the availability of the exponent $s \in \mathbb{Z}_p$. In the standard model, the Groth-Sahai techniques would require to turn the membership certificates into triples $(g^{1/(\omega+s)}, g^s, u^s)$, for some $u \in \mathbb{G}$ (as in [19]), which is not compatible with the revocation mechanism.

burden on the group manager who would have to generate updates for each unrevoked individual credential.

While, for various reasons, none of the above constructions conveniently supports large groups, a highly scalable revocation mechanism borrowed from the literature on broadcast encryption was recently described by Libert, Peters and Yung [44] (LPY). Using the Subset Cover framework of Naor, Naor and Lotspiech [47] (NNL), they described a history-independent revocable group signature in the standard model with constant verification time and at most polylogarithmic complexity in other parameters. The technique of [44] blends well with structure-preserving signatures [1,2] and the Groth-Sahai proofs [36]. The best tradeoff of [44] builds on the Subset Difference (SD) method [47] in its public-key variant due to Dodis and Fazio [31]. It features constant signature size and verification time, $O(\log N)$ -size group public keys, revocation lists of size $O(r)$ (as in standard PKIs and group signatures with verifier-local revocation) and membership certificates of size $O(\log^3 N)$. This can be reduced to $O(\log N)$ using the Complete Subtree method [47] but revocation lists are then inflated by a factor of $O(\log N/r)$. Although the Layered Subset Difference method [37] allows for noticeable improvements, the constructions of [44] suffer from relatively large membership certificates. However, some logarithmic dependency on the group size is expected when basing revocation on a tree-like NNL methodology.

1.2 Our Contributions

To date, in the only scalable revocable group signatures with constant verification time in the standard model [44], group members have to store a polylogarithmic number of group elements. In many applications, however, this can rapidly become unwieldy even for moderately large groups: for example, using the Subset Difference method with $N = 1000 \approx 2^{10}$, users may have to privately store thousands of group elements. In order to be competitive with other group signatures in the standard model such as [35] and still be able to revoke members while keeping them “stateless”, it is highly desirable to reduce this complexity.

In this paper, we start with the approach of [44] so as to instantiate the Subset Difference method, but obtain private keys of *constant* size without degrading other performance criteria. This may sound somewhat surprising since, in the SD method, (poly)logarithmic complexities inherently seem inevitable in several metrics. Indeed, in the context of broadcast encryption [47], it requires private keys of size $O(\log^2 N)$ (and even $O(\log^3 N)$ in the public key setting [31] if the result of Boneh-Boyten-Goh [13] is used). Here, we reduce this overhead to a constant while the only dependency on N is a $O(\log N)$ -size group public key.

The key idea is as follows. As in the NNL framework, group members are assigned to a leaf of a binary tree and each unrevoked member should belong to exactly one subset in the cover of authorized leafs determined by the group manager. Instead of relying on hierarchical identity-based encryption [15,38,33] as in the public-key variant [31] of NNL, we use a novel way for users to non-interactively prove their membership of some generic subset of the SD method using a proof of constant size.

To construct these “compact anonymous membership proofs”, we use *concise* vector commitment schemes [43,27], where each commitment can be opened w.r.t. individual coordinates in a space-efficient manner (namely, the size of a coordinate-wise opening does not depend on the length of the vector). These vector commitments interact nicely with the specific shape of subsets – as differences between two subtrees – in the SD method. Using them, we compactly encode as a vector the path from the user’s leaf to the root. To provide evidence of their inclusion in one of the SD subsets, group members successively prove the equality and the inequality between two coordinates of their vector (*i.e.*, two nodes of the path from their leaf to the root) and specific node labels indicated by an appropriate entry of the revocation list. This is where the position-wise openability of concise commitments is very handy. Of course, for anonymity purposes, the relevant entry of the revocation list only appears in committed form in the group signature. In order to prove that he is using a legal entry of the revocation list, the user generates a set membership proof [23] and proves knowledge of a signature from the group manager on the committed RL entry.

Our technique allows making the most of the LPY approach [44] by reducing the size of membership certificates to a small constant: at the cost of lengthening signatures by a factor of only 1.5, we obtain membership certificates consisting of only 9 group elements and a small integer. For $N = 1000$, users’ private keys are thus compressed by a multiplicative factor of several hundreds and this can only become more dramatic for larger groups. At the same time, our main scheme retains all the useful properties of [44]: like the construction of Nakanishi *et al.* [45], it does not require users to update their membership certificates at any time but, unlike [45], our group public key size is $O(\log N)$. Like the SD-based construction of [44], our system uses revocation lists of size $O(r)$, which is on par with Certificate Revocation Lists (CRLs) in PKIs. It is worth noting that RLs are *not* part of the group public key: verifiers only need to know the number of the latest revocation epoch and should not bother to read RLs entirely.

Eventually, our novel approach yields revocable group signatures that become competitive with the regular CRL approach in PKIs: signature generation and verification have constant cost, signatures and membership certificates being of $O(1)$ -size while revocation lists have size $O(r)$. A detailed efficiency comparison with previous approaches is given in the full version of the paper. Finally, it is conceivable that our improved revocation technique can find applications beyond group signatures.

2 Background

2.1 Bilinear Maps and Complexity Assumptions

We use bilinear maps $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ over groups of prime order p where $e(g, h) \neq 1_{\mathbb{G}_T}$ if and only if $g, h \neq 1_{\mathbb{G}}$. In these groups, we rely on hardness assumptions that are all non-interactive.

Definition 1 ([14]). *The Decision Linear Problem (DLIN) in \mathbb{G} , is to distinguish the distributions $(g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ and $(g^a, g^b, g^{ac}, g^{bd}, g^z)$, with*

$a, b, c, d \xleftarrow{R} \mathbb{Z}_p^*, z \xleftarrow{R} \mathbb{Z}_p^*$. The **Decision Linear Assumption** is the intractability of DLIN for any PPT distinguisher D.

Definition 2 ([12]). The **q -Strong Diffie-Hellman problem** (q -SDH) in \mathbb{G} is, given $(g, g^a, \dots, g^{(a^q)})$, for some $g \xleftarrow{R} \mathbb{G}$ and $a \xleftarrow{R} \mathbb{Z}_p$, to find a pair $(g^{1/(a+s)}, s) \in \mathbb{G} \times \mathbb{Z}_p$.

We use a signature scheme proposed by Abe *et al.* [1], the security of which relies on this assumption.

Definition 3 ([1]). In a group \mathbb{G} , the **q -Simultaneous Flexible Pairing Problem** (q -SFP) is, given $(g_z, h_z, g_r, h_r, a, \tilde{a}, b, \tilde{b} \in \mathbb{G})$ and $q \in \text{poly}(\lambda)$ tuples $(z_j, r_j, s_j, t_j, u_j, v_j, w_j) \in \mathbb{G}^7$ such that

$$\begin{aligned} e(a, \tilde{a}) &= e(g_z, z_j) \cdot e(g_r, r_j) \cdot e(s_j, t_j), \\ e(b, \tilde{b}) &= e(h_z, z_j) \cdot e(h_r, u_j) \cdot e(v_j, w_j), \end{aligned} \quad (1)$$

to find a new tuple $(z^*, r^*, s^*, t^*, u^*, v^*, w^*) \in \mathbb{G}^7$ satisfying relations (1) and such that $z^* \notin \{1_{\mathbb{G}}, z_1, \dots, z_q\}$.

The paper will appeal to an assumption that was implicitly introduced in [16].

Definition 4 ([16]). Let \mathbb{G} be a group of prime order p . The **ℓ -Diffie-Hellman Exponent** (ℓ -DHE) problem is, given elements $(g, g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) \in \mathbb{G}^{2\ell}$ such that $g_i = g^{(\alpha^i)}$ for each i and where $\alpha \xleftarrow{R} \mathbb{Z}_p^*$, to compute the missing element $g_{\ell+1} = g^{(\alpha^{\ell+1})}$.

We actually need a stronger variant, used in [39], of the ℓ -DHE assumption.

Definition 5. In a group \mathbb{G} of prime order p , the **Flexible ℓ -Diffie-Hellman Exponent** (ℓ -FlexDHE) problem is, given $(g, g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) \in \mathbb{G}^{2\ell}$ such that $g_i = g^{(\alpha^i)}$ for each i and where $\alpha \xleftarrow{R} \mathbb{Z}_p^*$, to compute a non-trivial triple $(g^\mu, g_{\ell+1}^\mu, g_{2\ell}^\mu) \in (\mathbb{G} \setminus \{1_{\mathbb{G}}\})^3$, for some $\mu \in \mathbb{Z}_p^*$ and where $g_{\ell+1} = g^{(\alpha^{\ell+1})}$.

The reason why we need to rely on the above assumption instead of the weaker ℓ -DHE assumption is that, in our proofs, the exponent $\mu \in \mathbb{Z}_p$ will appear inside Groth-Sahai commitments [36], from which only values of the form $(g^\mu, g_{\ell+1}^\mu)$ will be efficiently extractable. The additional element $g_{2\ell}^\mu$ will thus prevent the adversary from simply choosing $\mu = \alpha$ or $\mu = \alpha^{-1}$.

A proof of the generic hardness of the ℓ -FlexDHE problem is given in [39]. We note that, while the strength of the assumption grows with ℓ , ℓ is only logarithmic in the maximal number of users here.

2.2 Groth-Sahai Proof Systems

The fundamental Groth-Sahai (GS) techniques [36] can be based on the DLIN assumption, where they use prime order groups and a common reference string containing three vectors $\vec{f}_1, \vec{f}_2, \vec{f}_3 \in \mathbb{G}^3$, where $\vec{f}_1 = (f_1, 1, g)$, $\vec{f}_2 = (1, f_2, g)$ for

some $f_1, f_2 \in \mathbb{G}$. To commit to $X \in \mathbb{G}$, one chooses $r, s, t \xleftarrow{R} \mathbb{Z}_p^*$ and computes $\vec{C} = (1, 1, X) \cdot \vec{f}_1^r \cdot \vec{f}_2^s \cdot \vec{f}_3^t$. In the soundness setting, we have $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2}$ where $\xi_1, \xi_2 \in \mathbb{Z}_p^*$. Commitments $\vec{C} = (\vec{f}_1^{r+\xi_1 t}, \vec{f}_2^{s+\xi_2 t}, X \cdot g^{r+s+t(\xi_1+\xi_2)})$ are then extractable using $\beta_1 = \log_g(f_1)$, $\beta_2 = \log_g(f_2)$. In the witness indistinguishability (WI) setting, vectors $\vec{f}_1, \vec{f}_2, \vec{f}_3$ are linearly independent and \vec{C} is a perfectly hiding commitment. Under the DLIN assumption, the two kinds of CRS are indistinguishable.

To commit to an exponent $x \in \mathbb{Z}_p$, one computes $\vec{C} = \vec{\varphi}^x \cdot \vec{f}_1^r \cdot \vec{f}_2^s$, where $r, s \xleftarrow{R} \mathbb{Z}_p^*$, using a CRS consisting of vectors $\vec{\varphi}, \vec{f}_1, \vec{f}_2$. In the perfect soundness setting, $\vec{\varphi}, \vec{f}_1, \vec{f}_2$ are linearly independent whereas, in the WI setting, choosing $\vec{\varphi} = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2}$ gives a perfectly hiding commitment.

To prove that committed variables satisfy a set of relations, the prover computes one commitment per variable and one proof element per relation. Such non-interactive witness indistinguishable (NIWI) proofs are available for pairing-product equations, which are relations of the type

$$\prod_{i=1}^n e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^n \cdot \prod_{j=1}^n e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = t_T, \quad (2)$$

for variables $\mathcal{X}_1, \dots, \mathcal{X}_n \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \dots, \mathcal{A}_n \in \mathbb{G}$, $a_{ij} \in \mathbb{Z}_p$, for $i, j \in \{1, \dots, n\}$. Efficient NIWI proofs also exist for multi-exponentiation equations, which are of the form

$$\prod_{i=1}^m \mathcal{A}_i^{y_i} \cdot \prod_{j=1}^n \mathcal{X}_j^{b_j} \cdot \prod_{i=1}^m \cdot \prod_{j=1}^n \mathcal{X}_j^{y_i \gamma_{ij}} = T, \quad (3)$$

for variables $\mathcal{X}_1, \dots, \mathcal{X}_n \in \mathbb{G}$, $y_1, \dots, y_m \in \mathbb{Z}_p$ and constants $T, \mathcal{A}_1, \dots, \mathcal{A}_m \in \mathbb{G}$, $b_1, \dots, b_n \in \mathbb{Z}_p$ and $\gamma_{ij} \in \mathbb{G}$, for $i \in \{1, \dots, m\}, j \in \{1, \dots, n\}$.

In pairing-product equations, proofs for quadratic equations consist of 9 group elements whereas linear equations (*i.e.*, where $a_{ij} = 0$ for all i, j in equation (2)) only demand 3 group elements each. Linear multi-exponentiation equations of the type $\prod_{i=1}^m \mathcal{A}_i^{y_i} = T$ demand 2 group elements.

Multi-exponentiation equations admit zero-knowledge (NIZK) proofs at no additional cost. On a simulated CRS (prepared for the WI setting), a trapdoor allows simulating proofs without using the witnesses.

2.3 Structure-Preserving Signatures

Many anonymity-related protocols (e.g., [28,1,2,32,3]) require to sign elements of bilinear groups while maintaining the feasibility of conveniently proving that a committed signature is valid for a committed message.

Abe, Haralambiev and Ohkubo [1,2] (AHO) showed how to sign n group elements using signatures consisting of $O(1)$ group elements. In the context of symmetric pairings, the description hereafter assumes public parameters

pp = $((\mathbb{G}, \mathbb{G}_T), g)$ consisting of groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$, where $\lambda \in \mathbb{N}$ is a security parameter, with a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and a generator $g \in \mathbb{G}$.

Keygen(pk, n): given an upper bound $n \in \mathbb{N}$ on the number of group elements per signed message, choose generators $G_r, H_r \xleftarrow{R} \mathbb{G}$. Pick $\gamma_z, \delta_z \xleftarrow{R} \mathbb{Z}_p$ and $\gamma_i, \delta_i \xleftarrow{R} \mathbb{Z}_p$, for $i = 1$ to n . Then, compute $G_z = G_r^{\gamma_z}$, $H_z = H_r^{\delta_z}$ and $G_i = G_r^{\gamma_i}$, $H_i = H_r^{\delta_i}$ for each $i \in \{1, \dots, n\}$. Finally, choose $\alpha_a, \alpha_b \xleftarrow{R} \mathbb{Z}_p$ and define $A = e(G_r, g^{\alpha_a})$ and $B = e(H_r, g^{\alpha_b})$. The public key is defined to be

$$pk = (G_r, H_r, G_z, H_z, \{G_i, H_i\}_{i=1}^n, A, B) \in \mathbb{G}^{2n+4} \times \mathbb{G}_T^2$$

while the private key is $sk = (\alpha_a, \alpha_b, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^n)$.

Sign($sk, (M_1, \dots, M_n)$): to sign a vector $(M_1, \dots, M_n) \in \mathbb{G}^n$ using the private key $sk = (\alpha_a, \alpha_b, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^n)$, choose $\zeta, \rho_a, \rho_b, \omega_a, \omega_b \xleftarrow{R} \mathbb{Z}_p$ and compute $\theta_1 = g^\zeta$ as well as

$$\begin{aligned} \theta_2 &= g^{\rho_a - \gamma_z \zeta} \cdot \prod_{i=1}^n M_i^{-\gamma_i}, & \theta_3 &= G_r^{\omega_a}, & \theta_4 &= g^{(\alpha_a - \rho_a)/\omega_a}, \\ \theta_5 &= g^{\rho_b - \delta_z \zeta} \cdot \prod_{i=1}^n M_i^{-\delta_i}, & \theta_6 &= H_r^{\omega_b}, & \theta_7 &= g^{(\alpha_b - \rho_b)/\omega_b}, \end{aligned}$$

The signature consists of $\sigma = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)$.

Verify($pk, \sigma, (M_1, \dots, M_n)$): parse σ as $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7) \in \mathbb{G}^7$ and return 1 iff these equalities hold:

$$\begin{aligned} A &= e(G_z, \theta_1) \cdot e(G_r, \theta_2) \cdot e(\theta_3, \theta_4) \cdot \prod_{i=1}^n e(G_i, M_i), \\ B &= e(H_z, \theta_1) \cdot e(H_r, \theta_5) \cdot e(\theta_6, \theta_7) \cdot \prod_{i=1}^n e(H_i, M_i). \end{aligned}$$

The scheme was proved [1,2] existentially unforgeable under chosen-message attacks under the q -SFP assumption, where q is the number of signing queries.

Signatures can be publicly re-randomized to obtain a different signature $\{\theta'_i\}_{i=1}^7 \leftarrow \text{ReRand}(pk, \sigma)$ on the same message (M_1, \dots, M_n) . After randomization, we have $\theta'_1 = \theta_1$ while $\{\theta'_i\}_{i=2}^7$ are uniformly distributed among the values such that $e(G_r, \theta'_2) \cdot e(\theta'_3, \theta'_4) = e(G_r, \theta_2) \cdot e(\theta_3, \theta_4)$ and $e(H_r, \theta'_5) \cdot e(\theta'_6, \theta'_7) = e(H_r, \theta_5) \cdot e(\theta_6, \theta_7)$. Moreover, $\{\theta'_i\}_{i \in \{3,4,6,7\}}$ are statistically independent of the message and other signature components. This implies that, in privacy-preserving protocols, re-randomized $\{\theta'_i\}_{i \in \{3,4,6,7\}}$ can be safely given in the clear as long as (M_1, \dots, M_n) and $\{\theta'_i\}_{i \in \{1,2,5\}}$ are given in committed form.

2.4 Vector Commitment Schemes

We use concise vector commitment schemes, where commitments can be opened with a short de-commitment string for each individual coordinate. Such commitments based on ideas from [16,24] were described by Libert and Yung [43]

and, under weaker assumptions, by Catalano and Fiore [27]. In [43], the commitment key is $ck = (g, g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) \in \mathbb{G}^{2\ell}$, where $g_i = g^{(\alpha^i)}$ for each i . The trapdoor of the commitment is $g_{\ell+1}$, which does not appear in ck . To commit to a vector $\vec{m} = (m_1, \dots, m_\ell)$, the committer picks $r \xleftarrow{R} \mathbb{Z}_p$ and computes $C = g^r \cdot \prod_{\kappa=1}^{\ell} g_{\ell+1-\kappa}^{m_\kappa}$. A single group element $W_i = g_i^r \cdot \prod_{\kappa=1, \kappa \neq i}^{\ell} g_{\ell+1-\kappa+i}^{m_\kappa}$ provides evidence that m_i is the i -th component of \vec{m} as it satisfies the relation $e(g_i, C) = e(g, W_i) \cdot e(g_1, g_\ell)^{m_i}$. The infeasibility of opening a commitment to two distinct messages for some coordinate i relies on the ℓ -DHE assumption. For our purposes, we only rely on the position-wise binding property of vector commitments and do not need them to be hiding. The randomizer r will thus be removed from the expression of C .

2.5 The NNL Framework for Broadcast Encryption

The important Subset Cover framework [47] considers secret-key broadcast encryption schemes with $N = 2^\ell$ registered receivers. Each receiver is associated with a leaf of a complete binary tree T of height ℓ where each node is assigned a secret key. If \mathcal{N} denotes the universe of users and $\mathcal{R} \subset \mathcal{N}$ is the set of revoked receivers, the framework's idea is to partition the set of non-revoked users into m disjoint subsets S_1, \dots, S_m such that $\mathcal{N} \setminus \mathcal{R} = S_1 \cup \dots \cup S_m$. Depending on the way to divide $\mathcal{N} \setminus \mathcal{R}$, different tradeoffs are possible.

The Subset Difference (SD) method yields a transmission cost of $O(|\mathcal{R}|)$ and a storage complexity in $O(\log^2 N)$. For each node $x_j \in T$, we call T_{x_j} the subtree rooted at x_j . The unrevoked set $\mathcal{N} \setminus \mathcal{R}$ is partitioned into disjoint subsets $S_{k_1, u_1}, \dots, S_{k_m, u_m}$. For each $i \in \{1, \dots, m\}$, the subset S_{k_i, u_i} is determined by a node x_{k_i} and one of its descendants x_{u_i} – which are called *primary* and *secondary* roots of S_{k_i, u_i} , respectively – and it consists of the leaves of $T_{x_{k_i}}$ that are not in $T_{x_{u_i}}$. Each user belongs to many generic subsets, so that the number of subsets bounded by $m = 2 \cdot |\mathcal{R}| - 1$, as proved in [47].

In the broadcast encryption scenario, a sophisticated key distribution process is necessary to avoid a prohibitive storage overhead. Each subset S_{k_i, u_i} is assigned a “proto-key” $P_{x_{k_i}, x_{u_i}}$ that allows deriving the actual symmetric encryption key K_{k_i, u_i} for S_{k_i, u_i} and as well as proto-keys $P_{x_{k_i}, x_{u_l}}$ for any descendant x_{u_l} of x_{u_i} . Eventually, each user has to store $O(\log^2 N)$ keys. In the setting of group signatures, we will show that, somewhat unexpectedly, the use of vector commitment schemes allows reducing the private storage to a constant: the size of users' private keys only depends on the security parameter λ , and not on N .

2.6 Revocable Group Signatures

As in [45, 44], we consider schemes that have their lifetime divided into revocation epochs at the beginning of which group managers update their revocation lists.

The syntax and the security model are similar to those used by Kiayias and Yung [40]. Like the Bellare-Shi-Zhang model [10], the Kiayias-Yung model

assumes an interactive join protocol whereby the user becomes a group member by interacting with the group manager.

SYNTAX. We denote by $N \in \text{poly}(\lambda)$ the maximal number of group members. At the beginning of each revocation epoch t , the group manager publicizes an up-to-date revocation list RL_t and we denote by $\mathcal{R}_t \subset \{1, \dots, N\}$ the corresponding set of revoked users (we assume that \mathcal{R}_t is part of RL_t). A revocable group signature (R-GS) scheme consists of the following algorithms or protocols.

Setup(λ, N): given a security parameter $\lambda \in \mathbb{N}$ and a maximal number of group members $N \in \mathbb{N}$, this algorithm (which is run by a trusted party) generates a group public key \mathcal{Y} , the group manager's private key S_{GM} and the opening authority's private key S_{OA} . S_{GM} and S_{OA} are given to the appropriate authority while \mathcal{Y} is publicized. The algorithm initializes a public state St consisting of set and string data structures $St_{users} = \emptyset$ and $St_{trans} = \epsilon$.

Join: is an interactive protocol between the group manager GM and a prospective group member \mathcal{U}_i . The protocol involves two interactive Turing machines J_{user} and J_{GM} that both take as input \mathcal{Y} . The execution, denoted as $[J_{user}(\lambda, \mathcal{Y}), J_{GM}(\lambda, St, \mathcal{Y}, S_{GM})]$, ends with \mathcal{U}_i obtaining a membership secret sec_i , that no one else knows, and a membership certificate cert_i . If the protocol is successful, the group manager updates the public state St by setting $St_{users} := St_{users} \cup \{i\}$ as well as $St_{trans} := St_{trans} || \langle i, \text{transcript}_i \rangle$.

Revoke: is a (possibly probabilistic) algorithm allowing the GM to generate an updated revocation list RL_t for the new revocation epoch t . It takes as input a public key \mathcal{Y} and a set $\mathcal{R}_t \subset St_{users}$ that identifies the users to be revoked. It outputs an updated revocation list RL_t for epoch t .

Sign: given a revocation epoch t with its revocation list RL_t , a membership certificate cert_i , a membership secret sec_i and a message M , this algorithm outputs \perp if $i \in \mathcal{R}_t$ and a signature σ otherwise.

Verify: given a signature σ , a revocation epoch t , the corresponding revocation list RL_t , a message M and a group public key \mathcal{Y} , this deterministic algorithm returns either 0 or 1.

Open: takes as input a message M , a valid signature σ w.r.t. \mathcal{Y} for the indicated revocation epoch t , the opening authority's private key S_{OA} and the public state St . It outputs $i \in St_{users} \cup \{\perp\}$, which is the identity of a group member or a symbol indicating an opening failure.

A R-GS scheme must satisfy three security notions that are formally defined in the full version of the paper. The first one is called *security against misidentification attacks*. It requires that, even if the adversary can introduce and revoke users at will, it cannot produce a signature that traces outside the set of unrevoked adversarially-controlled users. The notion of *security against framing attacks* captures that under no circumstances should an honest user be held accountable for messages that he did not sign, even if the whole system conspires against that user. Finally, the notion of *anonymity* is also defined (by granting the adversary access to a signature opening oracle) as in the models of [10,40].

3 A Revocable Group Signature with Compact Keys and Constant Verification Time

The number of users is assumed to be $N = 2^{\ell-1} \in \text{poly}(\lambda)$, for some integer ℓ , so that each group member is assigned to a leaf of the tree. Each node is assigned a unique identifier. For simplicity, the root is identified by $\text{ID}(\epsilon) = 1$ and, for each other node x , we define the identifier $\text{ID}(x) \in \{1, \dots, 2N - 1\}$ to be $\text{ID}(x) = 2 \cdot \text{ID}(\text{parent}(x)) + b$, where $\text{parent}(x)$ denotes x 's father in the tree and $b = 0$ (resp. $b = 1$) if x is the left (resp. right) child of its father. The root of the tree is assigned the identifier $\text{ID}_\epsilon = 1$.

At the beginning of each revocation epoch t , the GM generates an up-to-date revocation list RL_t containing one entry for each generic subset $S_{k_1, u_1}, \dots, S_{k_m, u_m}$ produced by the Subset Difference method. These subsets are encoded in such a way that unrevoked users can anonymously prove their membership of one of them. Our technique allows to do this using a proof of *constant size*.

In the generation of RL_t , for each $i \in \{1, \dots, m\}$, if x_{k_i} (resp. x_{u_i}) denotes the primary (resp. secondary) root of S_{k_i, u_i} , the GM encodes S_{k_i, u_i} as a vector of group elements R_i that determines the levels of nodes x_{k_i} and x_{u_i} in the tree (which are called ϕ_i and ψ_i hereafter) and the identifiers $\text{ID}(x_{k_i})$ and $\text{ID}(x_{u_i})$. Then, the vector R_i is authenticated by means of a structure preserving signature Θ_i , which is included in RL_t so as to serve in a set membership proof [23].

During the join protocol, users obtain from the GM a structure-preserving signature on a compact encoding C_v – which is computed as a commitment to a vector of node identifiers (I_1, \dots, I_ℓ) – of the path (I_1, \dots, I_ℓ) between their leaf v and the root ϵ . This path is encoded as a single group element.

In order to anonymously prove his non-revocation, a group member \mathcal{U}_i uses RL_t to determine the generic subset S_{k_l, u_l} , with $l \in \{1, \dots, m\}$, where his leaf v_i lies. He commits to the corresponding vector of group elements R_l that encodes the node identifiers $\text{ID}(x_{k_l})$ and $\text{ID}(x_{u_l})$ of the primary and secondary roots of S_{k_l, u_l} at levels ϕ_l and ψ_l , respectively. If (I_1, \dots, I_ℓ) identifies the path from his leaf v_i to ϵ , the unrevoked member \mathcal{U}_i generates a membership proof for the subset S_{k_l, u_l} by proving that $\text{ID}(x_{k_l}) = I_{\phi_l}$ and $\text{ID}(x_{u_l}) \neq I_{\psi_l}$ (in other words, that x_{k_l} is an ancestor of v_i and x_{u_l} is not). To succinctly prove these statements, \mathcal{U}_i uses the properties of the commitment scheme recalled in Section 2.4. Finally, in order to convince the verifier that he used a legal element of RL_t , \mathcal{U}_i follows the technique of [23] and proves knowledge of a signature Θ_l on the committed vector of group elements R_l . By doing so, \mathcal{U}_i thus provides evidence that his leaf v_i is a member of some authorized subset S_{k_l, u_l} without revealing l .

In order to obtain the strongest flavor of anonymity (*i.e.*, where the adversary has access to a signature opening oracle), the scheme uses Kiltz's tag-based encryption scheme [41] as in Groth's construction [35]. In non-frameability concerns, the group member \mathcal{U}_i also generates a weak Boneh-Boyen signature [12] (which yields a fully secure signature when combined with a one-time signature) using $x = \log_g(X)$, where $X \in \mathbb{G}$ is a group element certified by the GM and bound to the path (I_1, \dots, I_ℓ) during the join protocol.

3.1 Construction

As in the security models of [10,40], we assume that, before joining the group, user \mathcal{U}_i chooses a long term key pair $(\text{usk}[i], \text{upk}[i])$ and registers it in some PKI.

Setup(λ, N): given a security parameter $\lambda \in \mathbb{N}$ and $N = 2^{\ell-1}$,

1. Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$, with $g \xleftarrow{R} \mathbb{G}$.
2. Define $n_0 = 2$ and $n_1 = 5$. Generate two key pairs $(sk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(0)})$ and $(sk_{\text{AHO}}^{(1)}, pk_{\text{AHO}}^{(1)})$ for the AHO signature in order to sign messages of n_0 and n_1 group elements, respectively. These key pairs are

$$pk_{\text{AHO}}^{(d)} = \left(G_r^{(d)}, H_r^{(d)}, G_z^{(d)} = G_r^{\gamma_z^{(d)}}, H_z^{(d)} = H_r^{\delta_z^{(d)}}, \{G_i^{(d)} = G_r^{\gamma_i^{(d)}}, H_i^{(d)} = H_r^{\delta_i^{(d)}}\}_{i=1}^{n_d}, A^{(d)}, B^{(d)} \right)$$

and $sk_{\text{AHO}}^{(d)} = (\alpha_a^{(d)}, \alpha_b^{(d)}, \gamma_z^{(d)}, \delta_z^{(d)}, \{\gamma_i^{(d)}, \delta_i^{(d)}\}_{i=1}^{n_d})$, where $d \in \{0, 1\}$.

3. Generate a public key $ck = (g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) \in \mathbb{G}^{2\ell-1}$ for vectors of dimension ℓ in the vector commitment scheme recalled in section 2.4. The trapdoor $g_{\ell+1}$ is not needed and can be discarded.
4. As a CRS for the NIWI proof system, select vectors $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ s.t. $\vec{f}_1 = (f_1, 1, g) \in \mathbb{G}^3$, $\vec{f}_2 = (1, f_2, g) \in \mathbb{G}^3$, and $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2}$, with $f_1 = g^{\beta_1}, f_2 = g^{\beta_2} \xleftarrow{R} \mathbb{G}$ and $\beta_1, \beta_2, \xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$. We also define the vector $\vec{\varphi} = \vec{f}_3 \cdot (1, 1, g)$.
5. Choose $(U, V) \xleftarrow{R} \mathbb{G}^2$ that, together with generators $f_1, f_2, g \in \mathbb{G}$, will form a public encryption key.
6. Select a strongly unforgeable one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$.
7. Set $\mathcal{S}_{\text{GM}} := (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$, $\mathcal{S}_{\text{OA}} := (\beta_1, \beta_2)$ as authorities' private keys and the group public key is

$$\mathcal{Y} := \left(g, pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}, ck = (g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}), \mathbf{f}, \vec{\varphi}, (U, V), \Sigma \right).$$

Join^(GM, \mathcal{U}_i): the GM and the prospective user \mathcal{U}_i run the following protocol $[\mathsf{J}_{\text{user}}(\lambda, \mathcal{Y}), \mathsf{J}_{\text{GM}}(\lambda, St, \mathcal{Y}, \mathcal{S}_{\text{GM}})]$:

1. $\mathsf{J}_{\text{user}}(\lambda, \mathcal{Y})$ draws $x \xleftarrow{R} \mathbb{Z}_p$ and sends $X = g^x$ to $\mathsf{J}_{\text{GM}}(\lambda, St, \mathcal{Y}, \mathcal{S}_{\text{GM}})$. If $X \in \mathbb{G}$ already appears in some entry transcript_j of the database St_{trans} , J_{GM} halts and returns \perp to J_{user} .
2. J_{GM} assigns to \mathcal{U}_i an available leaf v of identifier $\text{ID}(v)$ in the tree T . Let x_1, \dots, x_ℓ be the path from $x_\ell = v$ to the root $x_1 = \epsilon$ of T . Let also $(I_1, \dots, I_\ell) = (\text{ID}(x_1), \dots, \text{ID}(x_\ell))$ be the corresponding vector of identifiers (with $I_1 = 1$ and $I_\ell = \text{ID}(v) \in \{N, \dots, 2N - 1\}$). Then, J_{GM} does the following.
 - a. Encode (I_1, \dots, I_ℓ) as $C_v = \prod_{\kappa=1}^{\ell} g_{\ell+1-\kappa}^{I_\kappa} = g_\ell^{I_1} \cdots g_1^{I_\ell}$.

- b. Using $sk_{\text{AHO}}^{(0)}$, generate an AHO signature $\sigma_v = (\theta_{v,1}, \dots, \theta_{v,7})$ on the pair $(X, C_v) \in \mathbb{G}^2$ so as to bind the encoded path C_v to the value X that identifies \mathcal{U}_i .
- 3. J_{GM} sends $\text{ID}(v) \in \{N, \dots, 2N - 1\}$ and C_v to J_{user} that halts if $\text{ID}(v) \notin \{N, \dots, 2N - 1\}$ or if C_v is found incorrect. Otherwise, J_{user} sends a signature $sig_i = \text{Sign}_{\text{usk}[i]}(X \parallel (I_1, \dots, I_\ell))$ to J_{GM} .
- 4. J_{GM} checks that $\text{Verify}_{\text{upk}[i]}((X \parallel (I_1, \dots, I_\ell)), sig_i) = 1$. If not J_{GM} aborts. Otherwise, J_{GM} returns the AHO signature σ_v to J_{user} and stores the transcript $\text{transcript}_i = (X, \text{ID}(v), C_v, \sigma_v, sig_i)$ in the database St_{trans} .
- 5. J_{user} defines $\text{cert}_i = (\text{ID}(v), X, C_v, \sigma_v) \in \{N, \dots, 2N - 1\} \times \mathbb{G}^9$, where X will identify \mathcal{U}_i . The membership secret sec_i is defined as $\text{sec}_i = x \in \mathbb{Z}_p$.

Revoke($\mathcal{Y}, \mathcal{S}_{\text{GM}}, t, \mathcal{R}_t$): Parse \mathcal{S}_{GM} as $\mathcal{S}_{\text{GM}} := (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$.

- 1. Using the covering algorithm of the SD method, find a cover of the unrevoked user set $\{1, \dots, N\} \setminus \mathcal{R}_t$ as the union of disjoint subsets of the form $S_{k_1, u_1}, \dots, S_{k_m, u_m}$, with $m \leq 2 \cdot |\mathcal{R}_t| - 1$.
- 2. For $i = 1$ to m , do the following.
 - a. Consider S_{k_i, u_i} as the difference between sub-trees rooted at an internal node x_{k_i} and one of its descendants x_{u_i} . Let $\phi_i, \psi_i \in \{1, \dots, \ell\}$ be the depths of x_{k_i} and x_{u_i} , respectively, in T assuming that the root ϵ is at depth 1. Encode S_{k_i, u_i} as a vector $(g_{\phi_i}, g_1^{\text{ID}(x_{k_i})}, g_{\psi_i}, g^{\text{ID}(x_{u_i})})$.
 - b. To authenticate S_{k_i, u_i} and bind it to the revocation epoch t , use $sk_{\text{AHO}}^{(1)}$ to generate an AHO signature $\Theta_i = (\Theta_{i,1}, \dots, \Theta_{i,7}) \in \mathbb{G}^7$ on the message $R_i = (g^t, g_{\phi_i}, g_1^{\text{ID}(x_{k_i})}, g_{\psi_i}, g^{\text{ID}(x_{u_i})}) \in \mathbb{G}^5$, where the epoch number t is interpreted as an element of \mathbb{Z}_p .

Return the revocation data

$$RL_t = \left(t, \mathcal{R}_t, \{\phi_i, \psi_i, \text{ID}(x_{k_i}), \text{ID}(x_{u_i}), \Theta_i = (\Theta_{i,1}, \dots, \Theta_{i,7})\}_{i=1}^m \right). \quad (4)$$

Sign($\mathcal{Y}, t, RL_t, \text{cert}_i, \text{sec}_i, M$): return \perp if $i \in \mathcal{R}_t$. Otherwise, to sign $M \in \{0, 1\}^*$, generate a one-time signature key pair $(\text{SK}, \text{VK}) \leftarrow \mathcal{G}(\lambda)$. Parse cert_i as $\text{cert}_i = (\text{ID}(v_i), X, C_{v_i}, \sigma_{v_i}) \in \{N, \dots, 2N - 1\} \times \mathbb{G}^9$ and sec_i as $x \in \mathbb{Z}_p$. Let $\epsilon = x_1, \dots, x_\ell = v_i$ be the path connecting v_i to the root ϵ of T and let $(I_1, \dots, I_\ell) = (\text{ID}(x_1), \dots, \text{ID}(x_\ell))$ be the vector of node identifiers. First, \mathcal{U}_i generates a commitment $com_{C_{v_i}}$ to the encoding C_{v_i} of the path (I_1, \dots, I_ℓ) from v_i to the root. Then, he does the following.

- 1. Using RL_t , find the set S_{k_l, u_l} , with $l \in \{1, \dots, m\}$, containing the leaf v_i identified by $\text{ID}(v_i)$. Let x_{k_l} and x_{u_l} denote the primary and secondary roots of S_{k_l, u_l} at depths ϕ_l and ψ_l , respectively. Since x_{k_l} is an ancestor of v_i but x_{u_l} is not, it must be the case that $I_{\phi_l} = \text{ID}(x_{k_l})$ and $I_{\psi_l} \neq \text{ID}(x_{u_l})$.
- 2. To prove that v_i belongs to S_{k_l, u_l} without leaking l , \mathcal{U}_i first re-randomizes the l -th AHO signature Θ_l of RL_t as $\{\Theta'_{l,i}\}_{i=1}^7 \leftarrow \text{ReRand}(pk_{\text{AHO}}^{(1)}, \Theta_l)$.

Then, he commits to the l -th revocation message

$$R_l = (R_{l,1}, R_{l,2}, R_{l,3}, R_{l,4}, R_{l,5}) = (g^t, g_{\phi_l}, g_1^{\text{ID}(x_{k_l})}, g_{\psi_l}, g^{\text{ID}(x_{u_l})}) \quad (5)$$

and its signature $\Theta'_l = (\Theta'_{l,1}, \dots, \Theta'_{l,7})$ by computing Groth-Sahai commitments $\{com_{R_{l,\tau}}\}_{\tau=2}^5$, $\{com_{\Theta'_{l,j}}\}_{j \in \{1,2,5\}}$ to $\{R_{l,\tau}\}_{\tau=2}^5$ and $\{\Theta'_{l,j}\}_{j \in \{1,2,5\}}$.

- a. To prove that $I_{\phi_l} = \text{ID}(x_{k_l})$, \mathcal{U}_i computes $W_{\phi_l} = \prod_{\kappa=1, \kappa \neq \phi_l}^{\ell} g_{\ell+1-\kappa+\phi_l}^{I_{\kappa}}$ that satisfies the equality $e(g_{\phi_l}, C_{v_i}) = e(g_1, g_e)^{I_{\phi_l}} \cdot e(g, W_{\phi_l})$. Then, \mathcal{U}_i generates a commitment $com_{W_{\phi_l}}$ to W_{ϕ_l} . He computes a NIWI proof π_{eq} that committed variables $(R_{l,2}, R_{l,3}, C_{v_i}, W_{\phi_l})$ satisfy

$$e(R_{l,2}, C_{v_i}) = e(R_{l,3}, g_e) \cdot e(g, W_{\phi_l}). \quad (6)$$

- b. To prove that $I_{\psi_l} \neq \text{ID}(x_{u_l})$, \mathcal{U}_i computes $W_{\psi_l} = \prod_{\kappa=1, \kappa \neq \psi_l}^{\ell} g_{\ell+1-\kappa+\psi_l}^{I_{\kappa}}$ that satisfies the equality $e(g_{\psi_l}, C_{v_i}) = e(g_1, g_e)^{I_{\psi_l}} \cdot e(g, W_{\psi_l})$. Then, he computes a commitment $com_{W_{\psi_l}}$ to W_{ψ_l} as well as commitments com_{Γ_l} and $\{com_{\Psi_{l,\tau}}\}_{\tau \in \{0,1,2\ell\}}$ to the group elements

$$(\Gamma_l, \Psi_{l,0}, \Psi_{l,1}, \Psi_{l,2\ell}) = (g^{1/(I_{\psi_l} - \text{ID}(x_{u_l}))}, g^{I_{\psi_l}}, g_1^{I_{\psi_l}}, g_{2\ell}^{I_{\psi_l}}).$$

Then, \mathcal{U}_i proves that $(R_{l,4}, R_{l,5}, C_{v_i}, \Gamma_l, \Psi_{l,0}, \Psi_{l,1}, \Psi_{l,2\ell})$ satisfy

$$e(R_{l,4}, C_{v_i}) = e(\Psi_{l,1}, g_e) \cdot e(g, W_{\psi_l}), \quad e(\Psi_{l,0}/R_{l,5}, \Gamma_l) = e(g, g) \quad (7)$$

$$e(\Psi_{l,1}, g) = e(g_1, \Psi_{l,0}), \quad e(\Psi_{l,2\ell}, g) = e(g_{2\ell}, \Psi_{l,0}). \quad (8)$$

We denote this NIWI proof by $\pi_{eq} = (\pi_{eq,1}, \pi_{eq,2}, \pi_{eq,3}, \pi_{eq,4})$.

- 3. \mathcal{U}_i proves that the tuple R_l of (5) is a certified revocation message for epoch t : namely, he computes a NIWI proof π_{R_l} that committed message elements $\{R_{l,\tau}\}_{\tau=2}^5$ and signature components $\{\Theta'_{l,j}\}_{j \in \{1,2,5\}}$ satisfy

$$A^{(1)} \cdot e(\Theta'_{l,3}, \Theta'_{l,4})^{-1} \cdot e(G_1^{(1)}, g^t)^{-1} = e(G_z^{(1)}, \Theta'_{l,1}) \cdot \quad (9)$$

$$e(G_r^{(1)}, \Theta'_{l,2}) \cdot \prod_{\tau=2}^5 e(G_{\tau}^{(1)}, R_{l,\tau}),$$

$$B^{(1)} \cdot e(\Theta'_{l,6}, \Theta'_{l,7})^{-1} \cdot e(H_1^{(1)}, g^t)^{-1} = e(H_z^{(1)}, \Theta'_{l,1}) \\ \cdot e(H_r^{(1)}, \Theta'_{l,5}) \cdot \prod_{\tau=2}^5 e(H_{\tau}^{(1)}, R_{l,\tau}),$$

Since $\{\Theta'_{l,j}\}_{j \in \{3,4,6,7\}}$ are constants, equations (9) are both linear and thus require 3 elements each. Hence, π_{R_l} takes 6 elements altogether.

- 4. Let $\sigma_{v_i} = (\theta_{v_i,1}, \dots, \theta_{v_i,7})$ be the AHO signature on (X, C_{v_i}) . Compute a commitment com_X to X . Set $\{\theta'_{v_i,j}\}_{j=1}^7 \leftarrow \text{ReRand}(pk_{\text{AHO}}^{(0)}, \sigma_{v_i})$ and generate commitments $\{com_{\theta'_{v_i,j}}\}_{j \in \{1,2,5\}}$ to $\{\theta'_{v_i,j}\}_{j \in \{1,2,5\}}$. Then, generate a NIWI proof $\pi_{\sigma_{v_i}}$ that committed variables satisfy

$$A^{(0)} \cdot e(\theta'_{l,3}, \theta'_{l,4})^{-1} = e(G_z^{(0)}, \theta'_{l,1}) \cdot e(G_r^{(0)}, \theta'_{l,2}) \cdot e(G_1^{(0)}, X) \cdot e(G_2^{(0)}, C_{v_i}),$$

$$B^{(0)} \cdot e(\theta'_{l,6}, \theta'_{l,7})^{-1} = e(H_z^{(0)}, \theta_{l,1}) \cdot e(H_r^{(0)}, \theta'_{l,5}) \cdot e(H_1^{(0)}, X) \cdot e(H_2^{(0)}, C_{v_i})$$

5. Using VK as a tag, compute a tag-based encryption [41] of X by computing $(\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = (f_1^{z_1}, f_2^{z_2}, X \cdot g^{z_1+z_2}, (g^{\text{VK}} \cdot U)^{z_1}, (g^{\text{VK}} \cdot V)^{z_2})$ with $z_1, z_2 \xleftarrow{R} \mathbb{Z}_p$.
6. Generate a NIZK proof that $\text{com}_X = (1, 1, X) \cdot \vec{f}_1^{w_{X,1}} \cdot \vec{f}_2^{w_{X,2}} \cdot \vec{f}_3^{w_{X,3}}$ and $(\gamma_1, \gamma_2, \gamma_3)$ are BBS encryptions of the same value X . If we write $\vec{f}_3 = (f_{3,1}, f_{3,2}, f_{3,3})$, the Groth-Sahai commitment com_X can be written as $(f_1^{w_{X,1}} \cdot f_{3,1}^{w_{X,3}}, f_2^{w_{X,2}} \cdot f_{3,2}^{w_{X,3}}, X \cdot g^{w_{X,1}+w_{X,2}} \cdot f_{3,3}^{w_{X,3}})$, so that we have

$$\text{com}_X \cdot (\gamma_1, \gamma_2, \gamma_3)^{-1} = (f_1^{\chi_1} \cdot f_{3,1}^{\chi_3}, f_2^{\chi_2} \cdot f_{3,2}^{\chi_3}, g^{\chi_1+\chi_2} \cdot f_{3,3}^{\chi_3}) \quad (10)$$

with $\chi_1 = w_{X,1} - z_1$, $\chi_2 = w_{X,2} - z_2$, $\chi_3 = w_{X,3}$. To prove (10), compute $\text{com}_{\chi_j} = \vec{\varphi}^{\chi_j} \cdot \vec{f}_1^{w_{\chi_j,1}} \cdot \vec{f}_2^{w_{\chi_j,2}}$, with $w_{\chi_j,1}, w_{\chi_j,2} \xleftarrow{R} \mathbb{Z}_p$ for $j \in \{1, 2, 3\}$, as commitments to $\{\chi_j\}_{j=1}^3$ and generates proofs $\{\pi_{eq-com,j}\}_{j=1}^3$ that χ_1, χ_2, χ_3 satisfy the three linear relations (10).

7. Compute a weak Boneh-Boyen signature $\sigma_{\text{VK}} = g^{1/(x+\text{VK})}$ on VK and a commitment $\text{com}_{\sigma_{\text{VK}}}$ to σ_{VK} . Then, generate a NIWI proof $\pi_{\sigma_{\text{VK}}} = (\vec{\pi}_{\sigma_{\text{VK}},1}, \vec{\pi}_{\sigma_{\text{VK}},2}, \vec{\pi}_{\sigma_{\text{VK}},3}) \in \mathbb{G}^9$ that committed variables $(\sigma_{\text{VK}}, X) \in \mathbb{G}^2$ satisfy the quadratic equation $e(\sigma_{\text{VK}}, X \cdot g^{\text{VK}}) = e(g, g)$.
8. Compute $\sigma_{ots} = \mathcal{S}(\text{SK}, (M, RL_t, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \Omega, \text{com}, \Pi))$ where $\Omega = \{\Theta'_{l,i}, \theta'_{l,i}\}_{i \in \{3,4,6,7\}}$, $\Pi = (\pi_{eq}, \pi_{neq}, \pi_{R_t}, \pi_{\sigma_{v_i}}, \{\pi_{eq-com,j}\}_{j=1}^3, \pi_{\sigma_{\text{VK}}})$ and

$$\begin{aligned} \text{com} = & (\text{com}_{C_{v_i}}, \text{com}_X, \{\text{com}_{R_{l,\tau}}\}_{\tau=2}^5, \text{com}_{W_{\phi_l}}, \text{com}_{W_{\psi_l}}, \{\text{com}_{\Theta'_{l,j}}\}_{j \in \{1,2,5\}}, \\ & \text{com}_{T_l}, \{\text{com}_{\Psi_{l,\tau}}\}_{\tau \in \{0,1,2\ell\}}, \{\text{com}_{\theta'_{l,j}}\}_{j \in \{1,2,5\}}, \{\text{com}_{\chi_j}\}_{j=1}^3, \text{com}_{\sigma_{\text{VK}}}) \end{aligned}$$

Return the signature $\sigma = (\text{VK}, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \Omega, \text{com}, \Pi, \sigma_{ots})$.

Verify($\sigma, M, t, RL_t, \mathcal{Y}$): If $\mathcal{V}(\text{VK}, (M, RL_t, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \Omega, \text{com}, \Pi), \sigma_{ots}) = 0$ or if $(\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5)$ is not a well-formed tag-based encryption (that is, if $e(\gamma_1, g^{\text{VK}} \cdot U) \neq e(f_1, \gamma_4)$ or $e(\gamma_2, g^{\text{VK}} \cdot V) \neq e(f_2, \gamma_5)$), return 0. Then, return 1 if all proofs properly verify. Otherwise, return 0.

Open($M, t, RL_t, \sigma, \mathcal{S}_{OA}, \mathcal{Y}, St$): Return \perp if $\text{Verify}(\sigma, M, t, RL_t, \mathcal{Y}) = 0$. Otherwise, given $\mathcal{S}_{OA} = (\beta_1, \beta_2)$, compute $\tilde{X} = \gamma_3 \cdot \gamma_1^{-1/\beta_1} \cdot \gamma_2^{-1/\beta_2}$. In the database St_{trans} , find a record $\langle i, \text{transcript}_i = (X_i, \text{ID}(v_i), C_{v_i}, \sigma_{v_i}, sig_i) \rangle$ such that $X_i = \tilde{X}$. If no such record exists in St_{trans} , return \perp . Otherwise, return i .

At first glance, the variable $\Psi_{l,2\ell}$ and the proof of the second equality (8) may seem unnecessary in step 2.b of the signing algorithm. However, as detailed in the full version of the paper, this element plays a crucial role when it comes to prove the security under the ℓ -FlexDHE assumption.

As far as efficiency goes, each entry of RL_t contains 7 group elements and two node identifiers of $O(\log N)$ bits each. If $\lambda_{\mathbb{G}}$ is the bitlength of a group element, we have $\log N \ll \lambda_{\mathbb{G}}/2$ (since $\lambda \leq \lambda_{\mathbb{G}}$ and N is polynomial), so that the number of bits of RL_t is bounded by $2 \cdot |\mathcal{R}_t| \cdot (7 \cdot \lambda_{\mathbb{G}} + 2 \log N + 2 \log \log N) < 2 \cdot |\mathcal{R}_t| \cdot (9 \lambda_{\mathbb{G}})$ bits. The size of RL_t is thus bounded by that of $18 \cdot |\mathcal{R}_t|$ group elements.

Unlike [44], group members only need to store 9 group elements in their membership certificate. As far as the size of signature goes, **com** and **Pi** require

66 and 60 group elements, respectively. If the one-time signature of [34] is used, VK and σ_{ots} consist of 3 elements of \mathbb{G} and 2 elements of \mathbb{Z}_p , respectively. The global size σ amounts to that of 144 group elements, which is about 50% longer than [44]. In comparison with [35] (which does not natively support revocation), signatures are only longer by a factor of 3. At the 128-bit security level, each group element should have a 512-bit representation and a signature takes 9 kB.

Verifying signatures takes constant time. The signer has to compute at most $2\ell = O(\log N)$ exponentiations to obtain W_{ϕ_l} and W_{ψ_l} at the beginning of each revocation epoch. Note that these exponentiations involve short exponents of $O(\log N)$ bits each. Hence, computing W_{ϕ_l} and W_{ψ_l} requires $O(\log^2 N)$ multiplications in \mathbb{G} . For this reason, since we always have $\log^2 N \ll \lambda$ (as long as $N \ll 2^{\lambda^{1/2}}$), this cost is dominated by that of a single exponentiation in \mathbb{G} .

From a security point of view, we prove the following theorem in the full version of the paper.

Theorem 1. *Under the SFP, FlexDHE, SDH and DLIN assumptions, the scheme provides anonymity and security against misidentification and framing attacks .*

References

1. Abe, M., Haralambiev, K., Ohkubo, M.: Signing on Elements in Bilinear Groups for Modular Protocol Design. Cryptology ePrint Archive: Report 2010/133 (2010)
2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
3. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg (2011)
4. Acar, T., Nguyen, L.: Revocation for Delegatable Anonymous Credentials. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 423–440. Springer, Heidelberg (2011)
5. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive: Report 2005/385 (2005)
6. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
7. Ateniese, G., Song, D., Tsudik, G.: Quasi-Efficient Revocation in Group Signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
8. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
9. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press (1993)

10. Bellare, M., Shi, H., Zhang, C.: Foundations of Group Signatures: The Case of Dynamic Groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
11. Benaloh, J., de Mare, M.: One-Way Accumulators: A Decentralized Alternative to Digital Signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
12. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
13. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
14. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
15. Boneh, D., Franklin, M.: Identity based encryption from the Weil pairing. SIAM J. of Computing 32(3), 586–615 (2003)
16. Boneh, D., Gentry, C., Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
17. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM-CCS 2004, pp. 168–177. ACM Press (2004)
18. Boyen, X., Waters, B.: Compact Group Signatures Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
19. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
20. Bresson, E., Stern, J.: Efficient Revocation in Group Signatures. In: Kim, K.-C. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001)
21. Brickell, E.: An efficient protocol for anonymously providing assurance of the container of the private key. Submission to the Trusted Computing Group (April 2003)
22. Brickell, E., Camenisch, J., Chen, L.: Direct Anonymous Attestation. In: ACM-CCS 2004, pp. 132–145 (2004)
23. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
24. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
25. Camenisch, J., Kohlweiss, M., Soriente, C.: Solving Revocation with Efficient Update of Anonymous Credentials. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 454–471. Springer, Heidelberg (2010)
26. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
27. Catalano, D., Fiore, D.: Concise Vector Commitments and their Applications to Zero-Knowledge Elementary Databases. Cryptology ePrint Archive: Report 2011/495 (2011)
28. Cathalo, J., Libert, B., Yung, M.: Group Encryption: Non-interactive Realization in the Standard Model. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 179–196. Springer, Heidelberg (2009)

29. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) *EUROCRYPT 1991*. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
30. Delerablée, C., Pointcheval, D.: Dynamic Fully Anonymous Short Group Signatures. In: Nguyén, P.Q. (ed.) *VIETCRYPT 2006*. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006)
31. Dodis, Y., Fazio, N.: Public Key Broadcast Encryption for Stateless Receivers. In: Feigenbaum, J. (ed.) *DRM 2002*. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
32. Fuchsbauer, G.: Automorphic Signatures in Bilinear Groups and an Application to Round-Optimal Blind Signatures. Cryptology ePrint Archive: Report 2009/320 (2009)
33. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) *ASIACRYPT 2002*. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
34. Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: Lai, X., Chen, K. (eds.) *ASIACRYPT 2006*. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
35. Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) *ASIACRYPT 2007*. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
36. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) *EUROCRYPT 2008*. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
37. Halevy, D., Shamir, A.: The LSD Broadcast Encryption Scheme. In: Yung, M. (ed.) *CRYPTO 2002*. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
38. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) *EUROCRYPT 2002*. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
39. Izabachène, M., Libert, B., Vergnaud, D.: Block-Wise P-Signatures and Non-interactive Anonymous Credentials with Efficient Attributes. In: Chen, L. (ed.) *IMACC 2011*. LNCS, vol. 7089, pp. 431–450. Springer, Heidelberg (2011)
40. Kiayias, A., Yung, M.: Secure scalable group signature with dynamic joins and separable authorities. International Journal of Security and Networks (IJSN) 1(1/2), 24–45 (2006)
41. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) *TCC 2006*. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
42. Libert, B., Vergnaud, D.: Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) *CANS 2009*. LNCS, vol. 5888, pp. 498–517. Springer, Heidelberg (2009)
43. Libert, B., Yung, M.: Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs. In: Micciancio, D. (ed.) *TCC 2010*. LNCS, vol. 5978, pp. 499–517. Springer, Heidelberg (2010)
44. Libert, B., Peters, T., Yung, M.: Scalable Group Signatures with Revocation. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 609–627. Springer, Heidelberg (2012)
45. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In: Jarecki, S., Tsudik, G. (eds.) *PKC 2009*. LNCS, vol. 5443, pp. 463–480. Springer, Heidelberg (2009)
46. Nakanishi, T., Funabiki, N.: Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In: Roy, B. (ed.) *ASIACRYPT 2005*. LNCS, vol. 3788, pp. 533–548. Springer, Heidelberg (2005)

47. Naor, D., Naor, M., Lotspiech, J.: Revocation and Tracing Schemes for Stateless Receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
48. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
49. Nguyen, L., Safavi-Naini, R.: Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 372–386. Springer, Heidelberg (2004)
50. Song, D.: Practical forward secure group signature schemes. In: ACM-CCS 2001, pp. 225–234 (2001)
51. Tsang, P., Au, M.-H., Kapadia, A., Smith, S.: Blacklistable anonymous credentials: blocking misbehaving users without TTPs. In: ACM-CCS 2007, pp. 72–81 (2007)
52. Tsudik, G., Xu, S.: Accumulating Composites and Improved Group Signing. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (2003)
53. Zhou, S., Lin, D.: Shorter Verifier-Local Revocation Group Signatures from Bilinear Maps. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 126–143. Springer, Heidelberg (2006)

Tightly Secure Signatures and Public-Key Encryption

Dennis Hofheinz and Tibor Jager

Karlsruhe Institute of Technology, Germany
`{dennis.hofheinz,tibor.jager}@kit.edu`

Abstract. We construct the first public-key encryption scheme whose chosen-ciphertext (i.e., IND-CCA) security can be proved under a standard assumption *and* does not degrade in either the number of users or the number of ciphertexts. In particular, our scheme can be safely deployed in unknown settings in which no a-priori bound on the number of encryptions and/or users is known.

As a central technical building block, we construct the first structure-preserving signature scheme with a tight security reduction. (This signature scheme may be of independent interest.) Combining this scheme with Groth-Sahai proofs yields a tightly simulation-sound non-interactive zero-knowledge proof system for group equations. If we use this proof system in the Naor-Yung double encryption scheme, we obtain a tightly IND-CCA secure public-key encryption scheme from the Decision Linear assumption.

We point out that our techniques are not specific to public-key encryption security. Rather, we view our signature scheme and proof system as general building blocks that can help to achieve a tight security reduction.

Keywords: Tight security proofs, structure-preserving signatures, public-key encryption, Groth-Sahai proofs.

1 Introduction

Many interesting cryptographic primitives (such as public-key encryption and signature schemes) cannot be proven secure with current techniques, as their security would imply $P \neq NP$. Instead, we usually provide a proof of security under a suitable (computational) assumption (such as the hardness of factoring large integers). Concretely, a *security reduction* shows that any successful adversary \mathcal{A} on the scheme's security can be converted into a successful solver \mathcal{B} of the underlying computational problem. Naturally, we would desire that \mathcal{B} 's success $\epsilon_{\mathcal{B}}$ is at least as large as \mathcal{A} 's success $\epsilon_{\mathcal{A}}$ in attacking the system. However, security reductions often suffer from a nontrivial multiplicative *security loss* L (such that only $\epsilon_{\mathcal{A}} \leq L \cdot \epsilon_{\mathcal{B}}$ can be guaranteed).

The issue of a nontrivial security loss becomes particularly problematic, e.g., in the case of a realistic public-key encryption (PKE) scenario with many users who encrypt and send many ciphertexts. Standard security notions for PKE

schemes (such as IND-CCA security [44, 20]) only consider one user and one ciphertext. In particular, with very few exceptions ([9, 32]), most security proofs of encryption schemes only prove security in this simplified scenario. This can be justified with general results ([8, 9]) that show that one-user, one-ciphertext PKE security implies security in the much more realistic multi-user, multi-ciphertext case. However, this generic reduction suffers from a reduction loss of $L = n_U \cdot n_C$, where n_U is the number of users, and n_C is the number of ciphertexts per user.

That is, even if a PKE scheme reaches a certain level of security in the commonly considered one-user, one-ciphertext setting, its security level may be significantly lower in a realistic setting. (In fact, Bellare et al. [7] give a concrete example of such a scheme in the symmetric-key setting.) This is particularly problematic, since it may not be clear at deployment time for how many users and encryptions a PKE scheme will be used. We thus note that the analysis of cryptographic primitives in the multi-user setting is necessary to derive concrete security guarantees for realistic settings.

Let us say that a security reduction (in the multi-user setting) is *tight* if the corresponding reduction loss L is a (preferably small) constant. In particular, the security of a tightly secure scheme does not deteriorate in the number of users (or encryptions). For some security notions and constructions, tightly secure reductions can be constructed relatively painlessly. For instance, the random self-reducibility of the Decisional Diffie-Hellman problem allows to tightly prove the IND-CPA security of ElGamal encryption [21] even with many users and ciphertexts [9]. However, for other security notions, it seems inherently difficult to derive a tight security reduction.

For instance, there is no known PKE scheme with a tight (IND-CCA) security reduction from a standard assumption.¹ Diving into the technical details for a moment, one reason for this apparent difficulty is that an IND-CCA security reduction must be able to decrypt all of \mathcal{A} 's decryption queries, but must not be able to decrypt its own IND-CCA challenge. One way to resolve this dilemma is to partition the set of ciphertexts into those that can be decrypted, and those that cannot. (For instance, one can set up the proof simulation for \mathcal{A} such that the reduction can decrypt all ciphertexts except for one single challenge ciphertext; examples of this approach are [11, 34, 35].) This proof technique can only argue about a small number of ciphertexts at a time, and a hybrid argument is required to show security in the multi-ciphertexts case. Such a hybrid argument results again in a reduction loss that is linear in the number of ciphertexts.

Another way to show IND-CCA security is to argue with the information the adversary has about the secret key. (Examples of this approach are [18, 19, 38].)

¹ Bellare, Boldyreva, and Micali [9] show that the security loss of Cramer-Shoup encryption [18] does not depend on the number of users; however, their reduction loss still grows linearly in the number of ciphertexts per user. The identity-based encryption schemes [25, 26] enjoy a tight reduction, and can be generically converted into tightly IND-CCA secure PKE schemes [14]; however, they rely on a non-standard multi-challenge assumption. A similar argument holds for the tightly IND-SO-CCA secure PKE scheme of Hofheinz [32].

Since the size of the secret key is limited, its entropy can only be used to argue about the security of a limited number of ciphertexts at a time. Again a hybrid argument (entailing a linear reduction loss) is required to argue about the security of many ciphertexts. One could hope that the described inherent hybrid arguments of partitioning and entropy-based strategies to show IND-CCA security can be circumvented using dual system (identity-based) encryption techniques [47, 39]. In a nutshell, dual system encryption provides a way to subtly and gradually randomize the distribution of challenge ciphertexts (and user secret keys in an IBE scheme) without explicitly partitioning the set of ciphertexts into decryptable and non-decryptable ones. However, while dual system techniques rely on re-randomizable computational problems (such as the Decision Linear problem), and thus in principle should not suffer from the described problems, all known dual systems schemes still have to use a hybrid argument and do not achieve a tight security reduction. Note that one can construct IND-CCA-secure public-key encryption schemes with tight reduction in the random oracle model [5], for instance by applying the Fujisaki-Okamoto transform [23] to the tightly IND-CPA-secure schemes from [9].

In this paper, we present a general technique to construct tightly secure cryptographic primitives. As an example, we construct the first PKE scheme that is tightly IND-CCA secure under a simple assumption. Concretely, all the constructions in this paper build on the Decision Linear (DLIN) assumption.²

Our main technical building block is a *structure-preserving* signature scheme with a tight security reduction.³ Loosely speaking, a structure-preserving signature scheme is one in which verification can be expressed as a sequence of group equations. In particular, structure-preserving schemes are amenable to Groth-Sahai (GS) proofs [31], which are efficient non-interactive proof systems for sets of equations over a group. Following a known paradigm [40, 30, 16], we then turn our signature scheme into a *simulation-sound* non-interactive zero-knowledge proof system for group equations.⁴ Since our signature is tightly secure, so is the proof system.

This tightly secure and simulation-sound proof system offers the technical means to achieve tight security. We exemplify this by implementing the Naor-Yung paradigm [42, 40, 13] with our proof system to obtain a tightly IND-CCA secure PKE scheme. This construction appears in Section 5.

² However, we expect that our constructions also naturally generalize to the — potentially weaker — K -Linear assumption and to suitable subgroup decision assumptions.

³ We construct tightly secure structure-preserving signatures. (In fact, our schemes can sign their own public key; such signature schemes are commonly also referred to as automorphic.) While there exist tightly secure signature schemes (e.g., [24, 12, 17, 10, 36, 46]), and structure-preserving signature schemes (e.g., [22, 3, 16]), our scheme seems to be the first to achieve both properties. This combination of properties is crucial for our applications.

⁴ By a simulation-sound zero-knowledge proof system, we mean one in which it is infeasible to generate valid proofs for false statements, even when already having observed *many* simulated proofs for possibly false statements.

Our signature scheme is tree-based and inspired by the scheme of Boneh, Mironov, and Shoup [12]. However, their scheme is not structure-preserving, as it uses the hash of group elements as an exponent. To avoid this kind of “domain translation,” we construct a one-time signature scheme in which signatures and messages are vectors of group elements. (Since we want to implement a tree-based many-time signature scheme, we require, however, that messages can be longer than public verification keys.) To describe our (one-time) scheme, assume groups \mathbb{G}, \mathbb{G}_T with pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Write $E : \mathbb{G}^3 \times \mathbb{G} \rightarrow \mathbb{G}_T^3$ for component-wise pairing. (That is, $E((u_1, u_2, u_3), v) = (e(u_1, v), e(u_2, v), e(u_3, v))$.) In a nutshell, a signature for a message $m = (m_i)_{i=1}^n \in \mathbb{G}^n$ is of the form $(s, t) \in \mathbb{G}^2$ and satisfies

$$\left(\prod_{i=1}^n E(U_i, m_i) \right) \cdot E(G, s) \cdot E(H, t) = E(X, z), \quad (1)$$

where the $G, H, X, U_1, \dots, U_n \in \mathbb{G}^3$, and $z \in \mathbb{G}$ are part of the public verification key.⁵ This means that m_i, s, t, z act as coefficients in a linear equation (in the \mathbb{G} -exponent) for the vectors U_i, G, H, X . Now if all the vectors U_i, G, H, X are DLIN-tuples of the form (g^x, h^y, k^{x+y}) (for fixed g, h, k), then any message can be signed when knowing all U_i, G, H, X -exponents. Now consider a setup in which one U_i (say, U_j) and X are non-DLIN-tuples, and the other U_i and G, H are DLIN-tuples. Then, only messages with a specific m_j -component can be signed. (This is easiest seen by thinking of DLIN-elements as linearly dependent vectors in the exponent; with this view, X and U_j are the only vectors outside the vector space generated by DLIN-tuples. We note that this idea of “unique signatures” already appears in the ROM-based signature scheme of Katz and Wang [37].) In the proof of (non-adaptive) one-time security, this m_j will be set up as the message signed for the adversary \mathcal{A} . Furthermore, if the adversary forges a message, we know that this message must have m_j in its j -th component. Since a forged signature must refer to a message M^* that is different from the message M signed for \mathcal{A} , there must be an j with $m_j^* \neq m_j$. A small hybrid argument over $j \in [n]$ thus shows security. (We stress that we employ a hybrid argument only over a small set $[n]$ that will not depend on the number of users or ciphertexts. Specifically, our scheme can be implemented with $n = 8$.) From this one-time secure scheme, we will construct a tree-based many-time secure scheme following ideas from [12]; in particular, we will re-use the U_i for many instances of the one-time scheme. (Such a re-use of public key parts has also been used and made explicit [6].) This will finally yield an adaptively secure structure-preserving signature scheme with a tight security reduction. The remaining steps that lead to a tightly simulation-sound proof system and tightly IND-CCA secure public-key encryption follow existing ideas [30, 13], so we will not outline them here. (Details follow inside.)

We note that plugging our tightly simulation-sound proof system into the construction of [13] yields a PKE scheme that is tightly chosen-ciphertext secure

⁵ We highlight that (1) actually consists of three pairing product equations. This can in part be justified by [3, Theorem 2], which states that already any secure structure-preserving two-time signature scheme must have at least two verification equations.

even under *key-dependent* message attacks. Similarly, we expect that proofs of chosen-ciphertext security for *identity-based* encryption schemes can be made independent of the number of challenge ciphertexts. (However, here we do not expect to obtain independence from the number of users, i.e., identities.) Besides, structure-preserving signatures have found applications in several areas (e.g., [15, 22, 29]). We expect that *tightly secure* structure-preserving signature schemes lead to tighter security proofs in these applications.

In [2, Appendix C] (the full version of [1]), Abe et al. describe a DLIN-based structure-preserving one-time signature scheme that is more efficient than ours and has subsequently also been proven compatible with our tree-based approach [4]. In particular, together with our work, their scheme yields a more efficient tightly IND-CCA-secure encryption scheme. (We were not aware of their one-time signature scheme when designing ours.)

2 Preliminaries

Notation. If A is a set, then $a \xleftarrow{\$} A$ denotes that a is distributed uniformly over A . If A is a probabilistic algorithm, then $a \xleftarrow{\$} A$ denotes that a is computed by A using fresh random coins. For $n \in \mathbb{N}$ we write $[n]$ to denote the set $[n] = \{1, \dots, n\}$. For $j \in [n]$ we write $[n \setminus j]$ to denote the set $\{1, \dots, n\} \setminus \{j\}$.

Digital Signatures. Generally, we assume a parameter generation algorithm Sig.Param which takes as input the security parameter κ and generates public parameters $\Pi \xleftarrow{\$} \text{Sig.Param}(\kappa)$.

A digital signature scheme $\text{Sig} = (\text{Sig.Gen}, \text{Sig.Sign}, \text{Sig.Vfy})$ consists of three algorithms. Key generation algorithm Sig.Gen generates, on input parameters Π , a keypair $(vk, sk) \xleftarrow{\$} \text{Sig.Gen}(\Pi)$ consisting of a secret signing key sk and a public verification key vk . The signing algorithm Sig.Sign inputs a message and the secret signing key, and returns a signature $\sigma \xleftarrow{\$} \text{Sig.Sign}(sk, m)$ of the message. The verification algorithm Sig.Vfy takes a verification key and a message with corresponding signature as input, and returns $b \leftarrow \text{Sig.Vfy}(vk, m, \sigma)$ where $b \in \{0, 1\}$. We require the usual correctness properties.

Let us recall the *existential unforgeability against chosen message attacks* (EUF-CMA) security experiment [28], played between a challenger and a forger \mathcal{A} .

1. The forger, on input public parameters Π , may ask a *non-adaptive* chosen-message query. To this end, it submits a list of messages $M^{(1)}, \dots, M^{(q_0)}$ to the challenger.
2. The challenger runs $\text{Sig.Gen}(\Pi)$ to generate a keypair (vk, sk) . The forger receives vk and a signature $\sigma^{(i)}$ for each chosen message $M^{(i)}$, $i \in [q_0]$.
3. Now the forger may ask *adaptive* chosen-message queries, by submitting messages $M^{(q_0+1)}, \dots, M^{(q)}$ to the challenger. The challenger returns a signature $\sigma^{(i)}$ under sk for each message $M^{(i)}$, $i \in [q_0 + 1, q]$.
4. Finally the forger outputs a message M^* and signature σ^* .

Definition 1. An adversary is adaptive, if it asks at least one adaptive chosen-message query. Otherwise it is non-adaptive. Let \mathcal{A} be an adversary (adaptive or non-adaptive) that runs in time t , makes q chosen-message queries (in total), and outputs (M^*, σ^*) . We say that $\mathcal{A}(\epsilon, t, q)$ -breaks the EUF-CMA security of Sig if

$$\Pr[\text{Sig.Vfy}(vk, M^*, \sigma^*) = 1 \wedge M^* \notin \{M^{(1)}, \dots, M^{(q)}\}] \geq \epsilon.$$

We say that $\mathcal{A}(\epsilon, t, q)$ -breaks the strong EUF-CMA security of Sig if

$$\Pr[\text{Sig.Vfy}(vk, M^*, \sigma^*) = 1 \wedge (M^*, \sigma^*) \notin \{(M^{(1)}, \sigma^{(1)}), \dots, (M^{(q)}, \sigma^{(q)})\}] \geq \epsilon.$$

Accordingly, a signature scheme Sig is (ϵ, t, q) -secure against adaptive (non-adaptive) EUF-CMA attacks, if there exists no adaptive (non-adaptive) adversary that (ϵ, t, q) -breaks Sig .

Complexity Assumptions. In the following, let \mathbb{G} be a group of prime order p . For a generator $g \in \mathbb{G}$ and arbitrary $h \in \mathbb{G}$, let $\log_g(h) \in \mathbb{Z}_p$ be the discrete logarithm of h (to base g), such that $g^{\log_g(h)} = h$.

Definition 2. Let $g, h \in \mathbb{G}$ be random generators of \mathbb{G} . We say that an adversary $\mathcal{A}(\epsilon, t)$ -breaks the Discrete Logarithm (DLOG) assumption in \mathbb{G} , if \mathcal{A} runs in time t and $\Pr[\mathcal{A}(g, h) = \log_g(h)] \geq \epsilon$.

Furthermore, for generators $g, h, k \in \mathbb{G}$ let $\text{DLIN}(g, h, k)$ denote the set

$$\text{DLIN}(g, h, k) = \{(g^u, h^v, k^{u+v}) : u, v \in \mathbb{Z}_p\}$$

Let $G = (g, 1, k) \in \mathbb{G}^3$ and $H = (1, h, k) \in \mathbb{G}^3$. For two vectors $V = (v_1, v_2, v_3)$ and $W = (w_1, w_2, w_3)$ in \mathbb{G}^3 and $u \in \mathbb{Z}_p$ let $V \cdot W := (v_1 \cdot w_1, v_2 \cdot w_2, v_3 \cdot w_3)$ and let $V^u := (v_1^u, v_2^u, v_3^u)$. Then we can write the set $\text{DLIN}(g, h, k)$ equivalently as

$$\text{DLIN}(g, h, k) = \{U : U = G^u \cdot H^v, u, v \in \mathbb{Z}_p\}.$$

Definition 3. Let $g, h, k \xleftarrow{\$} \mathbb{G}$ be random generators of group \mathbb{G} , and let $U \xleftarrow{\$} \text{DLIN}(g, h, k)$ and $V \xleftarrow{\$} \mathbb{G}^3$. We say that an adversary $\mathcal{B}(\epsilon, t)$ -breaks the Decision Linear (DLIN) assumption in \mathbb{G} , if \mathcal{B} runs in time t and

$$\Pr[\mathcal{B}(g, h, k, U) = 1] - \Pr[\mathcal{B}(g, h, k, V) = 1] \geq \epsilon.$$

3 Structure-Preserving Signatures

In the sequel let \mathbb{G}, \mathbb{G}_T be groups of prime order p with bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and let g, h, k be random generators of \mathbb{G} . For a vector $V = (v_0, v_1, v_2) \in \mathbb{G}^3$ and a group element $w \in \mathbb{G}$, we write $E(V, w)$ to denote the vector

$$E(V, w) = (e(v_0, w), e(v_1, w), e(v_2, w)).$$

For two vectors $V = (v_0, v_1, v_2)$ and $W = (w_0, w_1, w_2)$ we denote with $V \cdot W$ the component-wise product $V \cdot W = (v_0 \cdot w_0, v_1 \cdot w_1, v_2 \cdot w_2)$. For $w \in \mathbb{Z}_p$ we write V^w to denote $V^w = (v_0^w, v_1^w, v_2^w)$.

3.1 One-Time Signatures for Single Group Elements

Let $\text{OTSig} = (\text{OTSig.Gen}, \text{OTSig.Sign}, \text{OTSig.Vfy})$ be the following signature scheme.

OTSig.Gen(g, h, k): Given random generators $g, h, k \in \mathbb{G}$, choose a random generator $z \xleftarrow{\$} \mathbb{G}$ and integers $u, v, x, y \xleftarrow{\$} \mathbb{Z}_p$. Set $U := (g^u, h^v, k^{u+v})$ and $X := (g^x, h^y, k^{x+y})$. Set $vk := (g, h, k, U, X, z)$ and $sk := (u, v, x, y)$ and return (vk, sk) .

OTSig.Sign(sk, m): Given a message $m \in \mathbb{G}$ and a secret key $sk = (u, v, x, y)$, compute $s := z^x m^{-u}$ and $t := z^y m^{-v}$ and return $\sigma = (s, t)$.

OTSig.Vfy(vk, m, σ): Given a public key $vk = (g, h, k, U, X, z)$, message m , and signature $\sigma = (s, t)$, let $G := (g, 1, k)$ and $H = (1, h, k)$. Return 1 if equation

$$E(U, m) \cdot E(G, s) \cdot E(H, t) = E(X, z)$$

holds. Otherwise return 0.

Theorem 1. Suppose there exists a non-adaptive adversary \mathcal{A} that $(\epsilon, t, 1)$ -breaks the EUF-CMA security of OTSig. Then there exists an adversary \mathcal{B} that (ϵ', t') -breaks the DLIN assumption in \mathbb{G} , where t' is roughly the runtime of the EUF-CMA experiment with \mathcal{A} , and $\epsilon' \geq \epsilon - 1/p$.

Due to space limitations, we have to refer to the full version [33] for the proof.

3.2 One-Time Signatures for Vectors of Group Elements

In this section we extend the message space of OTSig from the previous section to vectors $M = (m_1, \dots, m_n)$ of n elements of \mathbb{G} . The scheme is very similar, except that now the public key and the verification equation contain n elements U_1, \dots, U_n instead of a single element U .

Scheme $\text{OTSig}^n = (\text{OTSig.Gen}^n, \text{OTSig.Sign}^n, \text{OTSig.Vfy}^n)$ works as follows.

OTSig.Gen n (g, h, k): Given a generators $g, h, k \in \mathbb{G}$, choose a random generator $z \xleftarrow{\$} \mathbb{G}$ and $2(n+1)$ integers $u_1, v_1, \dots, u_n, v_n, x, y \xleftarrow{\$} \mathbb{Z}_p$. Set $U_i := (g^{u_i}, h^{v_i}, k^{u_i+v_i})$ for $i \in [n]$ and $X := (g^x, h^y, k^{x+y})$. Define the keys as $vk := (g, h, k, U_1, \dots, U_n, X, z)$ and $sk := (u_1, v_1, \dots, u_n, v_n, x, y)$ and return (vk, sk) .

OTSig.Sign n (sk, M): Given a message vector $M = (m_1, \dots, m_n) \in \mathbb{G}^n$ and secret key sk , compute $s := z^x \prod_{i=1}^n m_i^{-u_i}$ and $t := z^y \prod_{i=1}^n m_i^{-v_i}$ and return $\sigma = (s, t)$.

OTSig.Vfy n (vk, M, σ): Given a public key vk , message vector $M = (m_1, \dots, m_n)$, and signature $\sigma = (s, t)$, let $G := (g, 1, k)$ and $H = (1, h, k)$. Return 1 if equation

$$\prod_{i=1}^n E(U_i, m_i) \cdot E(G, s) \cdot E(H, t) = E(X, z)$$

holds. Otherwise return 0.

Theorem 2. Suppose there exists a non-adaptive adversary \mathcal{A} that $(\epsilon, t, 1)$ -breaks the EUF-CMA security of OTSig^n . Then there exists an adversary \mathcal{B} that (ϵ', t') -breaks the DLIN assumption in \mathbb{G} , where t' is roughly with runtime of the EUF-CMA experiment with \mathcal{A} , and $\epsilon' \geq \epsilon/n - 1/p$.

Again we have to refer to the full version [33] for the proof.

3.3 Signatures Secure against Non-adaptive Adversaries

Now we construct a signature scheme based on a binary tree of depth d . The scheme has message space \mathbb{G}^8 , allows us to issue up to 2^d signatures (where d may be large enough such that 2^d is virtually unbounded, e.g. $d = 80$), and is provably secure against non-adaptive adversaries under the DLIN assumption.

Basic Idea. The construction is based on binary Merkle trees [41], instantiated such that all nodes except for the root can be generated “on the fly.” In particular, not the complete tree must be stored (which would clearly be infeasible for large d). Each node of the tree consists of a key pair (vk, sk) of our one-time signature scheme from Section 3.2. The two children of this node are authenticated by a signature over their respective public keys that verifies under vk . The key-pairs corresponding to tree leaves are used to sign actual messages.

Recall that a public key consists of a vector $(g, h, k, U_1, \dots, U_n, X, z)$, where n is the number of group elements to be signed. In order to obtain a tight security reduction, we re-use the public-key components $(g, h, k, U_1, \dots, U_n)$ for all nodes of the tree. Only the (X, z) -components are unique for each node. The tight reduction is inspired by the proof of the tree-based signature scheme of Boneh et al. [12]. Let us give some more details on an informal level.

- The tree is parametrized by $(g, h, k) \in \mathbb{G}^3$ and $U_1, \dots, U_8 \in \text{DLIN}(g, h, k)$, where for each $i \in [8]$ we have $U_i = (g^{u_i}, h^{v_i}, k^{u_i+v_i})$ for random $u_i, v_i \xleftarrow{\$} \mathbb{Z}_p$. (It will later become clear that we will sign vectors of group elements, where each consists of 8 group elements. This is the reason why we choose $n = 8$ here).
- Each tree node N is identified by a four-tuple of group elements $N = (X, z) \in \mathbb{G}^4$, where $z \xleftarrow{\$} \mathbb{G}$ is random and $X = (g^x, h^y, k^{x+y})$ for random $x, y \xleftarrow{\$} \mathbb{Z}_p$.
- To each node $N = (X, z)$ of the tree we assign the public key

$$vk = (g, h, k, U_1, \dots, U_8, X, z)$$

with corresponding secret key $sk_N = (u_1, v_1, \dots, u_8, v_8, x, y)$. Note that this is a valid key pair for the one-time signature scheme from Section 3.2, instantiated such that vectors of 8 group elements can be signed. Note also that each node is identified by $(X, z) \in \mathbb{G}^4$, so that we can sign two child nodes with each public key.

- The tree is constructed — on the fly — as follows. Let $N_L = (X_L, z_L)$ and $N_R = (X_R, z_R)$ be the two children of node $N = (X, z)$. Then a signature of the message $M = (N_L, N_R) = (X_L, z_L, X_R, z_R) \in \mathbb{G}^8$ under secret key sk_N authenticates N_L and N_R as children of N . (This is why we chose $n = 8$).

– This gives the following signature scheme, which can be used to sign 8-tuples of elements of \mathbb{G} :

- The public key of the signature scheme consists of $(g, h, k, U_1, \dots, U_8)$ and the root node $N_0 = (X_0, z_0)$, the secret key consists of the discrete logarithms $(u_1, v_1, \dots, u_8, v_8, x_0, y_0)$.
- In order to sign a message $M = M_d \in \mathbb{G}^8$, select a leaf node N_d which has not been used before. Let N_{d-1}, \dots, N_0 denote the path from N_d to the root N_0 , and for all N_i ($i \in \{1, \dots, d-1\}$), let N_i^{co} denote the sibling of N_i . Let

$$M_{i-1} := \begin{cases} (N_i, N_i^{\text{co}}), & \text{if } N_i^{\text{co}} \text{ is the right-hand sibling of } N_i, \\ (N_i^{\text{co}}, N_i), & \text{if } N_i^{\text{co}} \text{ is the left-hand sibling of } N_i. \end{cases} \quad (2)$$

A signature for M_d consists of all pairs M_{d-1}, \dots, M_0 and signatures $(\sigma_d, \dots, \sigma_0)$ such that each signature σ_i authenticates M_i as child of node N_{i-1} .

We note that, strictly speaking, the described scheme is not structure-preserving. (The reason is the case distinction in (2).) We will show later how to implement our scheme in a structure-preserving way. A more precise description as well as a graphical illustration of the scheme can be found in the full version [33].

Theorem 3. *Suppose there exists a non-adaptive adversary \mathcal{A} that (ϵ, t, q) -breaks the EUF-CMA security of TSig . Then there exists an adversary \mathcal{B} that (ϵ', t') -breaks the DLIN assumption in \mathbb{G} , where t' is roughly the runtime of the EUF-CMA experiment with \mathcal{A} , and $\epsilon' = \epsilon/8$.*

Note that the success probability ϵ' of the DLIN-breaker \mathcal{B} is independent of the number q of chosen-message queries issued by \mathcal{A} . In the full version [33] we also describe how to construct an adaptively secure scheme with tight reduction.

4 Tightly Simulation-Sound NIZK Proofs for Pairing Product Equations

In this section we use the signature scheme from Section 3.3 to construct a NIZK proof for satisfiability of pairing product equations whose security reduces tightly to the DLIN assumption. “Tight” means here that the success probability of the reduction is independent of the number of simulated proofs the adversary sees. The construction is a special case of Groth-Sahai (GS) proofs [31], and uses a trick from [30, Section 4] to express the disjunction of two sets of pairing product equations as one set.

Non-Interactive Zero-Knowledge Proofs. Let R be a binary relation and let $\mathcal{L} := \{x : \exists w \text{ s.t. } R(x, w) = 1\}$ be the language defined by R . A non-interactive zero-knowledge proof system $\text{NIZK} = (\text{NIZK.Gen}, \text{NIZK.Prove}, \text{NIZK.Vfy})$ for \mathcal{L} consists of three algorithms. The common reference string generation algorithm

$crs \xleftarrow{\$} \text{NIZK.Gen}(\kappa)$ takes as input a security parameter κ and outputs a common reference string crs . Algorithm $\pi \xleftarrow{\$} \text{NIZK.Prove}(crs, x, w)$ takes as input crs , statement x , and a witness w that $x \in \mathcal{L}$, and outputs a proof π . The verification algorithm $\text{NIZK.Vfy}(crs, \pi, x) \in \{0, 1\}$ takes as input proof π and statement x . We say that NIZK.Vfy accepts if $\text{NIZK.Vfy}(crs, \pi, x) = 1$. We say that NIZK.Vfy rejects if $\text{NIZK.Vfy}(crs, \pi, x) = 0$.

NIZK is $(\epsilon_{\text{ZK}}, \epsilon_{\text{snd}}, \epsilon_{\text{simsnd}}, t, Q)$ -secure, if the following holds.

Perfect completeness. For each $(x, w) \in R$, each parameter κ , and each $crs \xleftarrow{\$} \text{NIZK.Gen}(\kappa)$ holds that

$$\Pr[\text{NIZK.Vfy}(crs, \pi, x) = 1 : \pi \xleftarrow{\$} \text{NIZK.Prove}(crs, x, w)] = 1.$$

Soundness. For all adversaries \mathcal{A} running in time t holds that

$$\Pr[\mathcal{A}(crs) = (x, \pi) : x \notin \mathcal{L} \wedge \text{NIZK.Vfy}(crs, \pi, x) = 1] \leq \epsilon_{\text{snd}}$$

Zero knowledge. There exists a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, such that $(crs, td) \xleftarrow{\$} \mathcal{S}_0(\kappa)$ generates a common reference string and trapdoor information td , and $\pi \xleftarrow{\$} \mathcal{S}_1(crssim, td, x)$ generates a simulated proof π for statement x (where not necessarily $x \in \mathcal{L}$).

Let $crs_{\text{real}} \xleftarrow{\$} \text{NIZK.Gen}(\kappa)$ and let $\mathcal{O}_{\text{real}}$ denote an oracle that takes as input $(x, w) \in R$ and returns $\text{NIZK.Prove}(crs_{\text{real}}, x, w)$. Let $(crssim, td) \xleftarrow{\$} \mathcal{S}_0(\kappa)$ and let \mathcal{O}_{sim} return $\mathcal{S}_1(crssim, td, x)$ on input $(x, w) \in R$. We require that

$$\Pr[\mathcal{A}^{\mathcal{O}_{\text{real}}}(crs_{\text{real}}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\text{sim}}}(crssim) = 1] \leq \epsilon_{\text{ZK}}$$

for all \mathcal{A} running in time at most t that issue at most Q oracle queries.

Simulation soundness. For $crs \xleftarrow{\$} \mathcal{S}_0(\kappa)$ and for all adversaries \mathcal{A} running in time t that may query \mathcal{S}_1 at most Q times for simulated proofs π_1, \dots, π_Q of arbitrary statements x_1, \dots, x_Q (where possibly $x_i \notin \mathcal{L}$ for some or all $i \in [Q]$) holds that

$$\Pr \left[\mathcal{A}^{\mathcal{S}_1}(crs) = (x, \pi) : \begin{array}{l} x \notin \mathcal{L} \wedge (x, \pi) \neq (x_i, \pi_i) \forall i \in [Q] \\ \text{and } \text{NIZK.Vfy}(crs, \pi, x) = 1 \end{array} \right] \leq \epsilon_{\text{simsnd}}$$

We will also use a variant of NIZK proof systems as a technical building block. Namely, a **(perfectly) non-interactive witness-indistinguishable (NIWI) proof system** is defined like a NIZK proof system above, with the following difference. Instead of the zero-knowledge and simulation-soundness properties, we require (perfect) witness-indistinguishability: for all crs in the image of NIZK.Gen , and all $(x, w_1), (x, w_2) \in R$ (for the same x), we require that the distributions induced by $\text{NIZK.Prove}(crs, x, w_1)$ and $\text{NIZK.Prove}(crs, x, w_2)$ are identical.

4.1 Building Blocks

Pairing Product Equations. Following [30, 31], a pairing product equation (PPE) s of length ℓ over \mathbb{G} is an equation of the form

$$\prod_{j=1}^{\ell} e(Q_{j,0}, Q_{j,1}) = 1 \quad \text{with} \quad Q_{j,b} = A_{j,b} \cdot \prod_{i=1}^{\nu} X_i^{\alpha_{j,b,i}} \tag{3}$$

where the $A_i \in \mathbb{G}$ and $\alpha_{j,b,i} \in \mathbb{Z}_p$ are constants, and the $X_i \in \mathbb{G}$ are variables. We say that a vector $\vec{x} = (x_1, \dots, x_\nu) \in \mathbb{G}^\nu$ satisfies the equation, if Equation 3 holds when setting $X_i = x_i$. A set S of pairing product equations is *satisfiable*, if there exists a vector \vec{x} that satisfies all equations $s \in S$ simultaneously. In the following, we will consider sets of satisfiable PPEs as languages for NIZK proof systems.

Disjunctions of Pairing Product Equations. Groth [30, Section 4.8] shows how to express the disjunction of several sets of PPEs through one set. Concretely, given n sets S_1, \dots, S_n of PPEs, he constructs a set $S := OR(S_1, \dots, S_n)$ of PPEs such that

- every solution \vec{x} that satisfies S allows to efficiently derive a solution \vec{x}_i of at least one S_i ,
- every solution \vec{x}_i of some S_i allows to efficiently derive a solution \vec{x} of S ,
- if S_i has ν_i variables X_i and consists of equations of total length ℓ_i , then S has total length $2\ell + 1$ for $\ell = \sum_{i=1}^n \ell_i$, and $(\sum_{i=1}^n \nu_i) + n + \ell$ variables.

NIWI Proofs for a Set of Pairing Product Equations. Groth and Sahai [31] present an efficient non-interactive witness-indistinguishable proof system for arbitrary sets of PPEs. Their system features a CRS crs that can be chosen either to be *hiding* or to be *binding*. If crs is hiding, then the resulting proofs are perfectly witness-indistinguishable. If crs is binding, the resulting proofs enjoy perfect soundness, and become extractable: a special trapdoor to crs allows to extract a witness \vec{x} from a valid proof. Hiding and binding CRSs are computationally indistinguishable under the DLIN assumption in the underlying group \mathbb{G} . When implemented over a DLIN-group (as will be the case in our setting), their system has the following efficiency properties (cf. Figure 2 in [31]):

- the CRS contains 6 \mathbb{G} -elements,
- each used variable X_i results in 3 \mathbb{G} -elements in the proof,
- each PPE incurs 9 \mathbb{G} -elements in the proof.

TSig-Verification as a Set of Pairing Product Equations. The verification algorithm of our weakly-secure DLIN-based signature scheme **TSig** from Section 3 can be expressed as a set of PPEs. Concretely, assume a verification key $vk = (g, h, k, U_1, \dots, U_8, X_0, z_0)$ for **TSig**, and a message $M = (m_{d,1}, \dots, m_{d,8}) \in \mathbb{G}^8$. Recall that a **TSig**-signature $\Sigma \in \mathbb{G}^{10d+2}$ determines OTSig-verification-keys vk_i , messages M_i , and signatures $\Sigma^{(i)}$ such that Σ is valid iff $\Sigma^{(i)}$ is a valid OTSig-signature of $M_i = (m_{i,j})_{j=1}^8$ under vk_{i-1} for all $i \in [d]$. Hence, verification amounts to checking a set $S_{vk, M}^{\text{TSig}} = \{OR(S_{L,i}, S_{R,i})\}_{i \in [d]}$, where $S_{D,i}$ (for $i \in [d]$ and $D \in \{L, R\}$) is given by

$$S_{D,i} = \left\{ \left(\prod_{j=1}^8 E(U_i, m_{i,j}) \right) \cdot E(G, s_i) \cdot E(H, t_i) = E(X_{D,i}, z_{D,i}) \right\}$$

of PPEs, where $G, H, U_1, \dots, U_8 \in \mathbb{G}^3$ and the $m_{d,j} \in \mathbb{G}$ are constants, and the $m_{i,j} \in \mathbb{G}$ (for $i \in [d-1]$), $X_{L,i}, X_{R,i} \in \mathbb{G}^3$, and $z_{L,i}, z_{R,i}, s_i, t_i \in \mathbb{G}$ (for $i \in [d]$) are variables.

Tightly Secure One-Time Signatures. As a final preparation, we require a means to secure proofs from tampering. Typically, this is done via a one-time signature scheme (as, e.g., in [40]). For our purposes, however, we require *tightly secure* (but not necessarily structure-preserving) one-time signatures. To enable a tight reduction, we will consider signature schemes with an algorithm TOTS.Param that outputs common system parameters $\text{pars}_{\text{tots}}$.

Definition 4. *The security experiment for strong n -fold one-time EUF-CMA security is identical to the strong one-time EUF-CMA experiment (see Section 2), except that the adversary \mathcal{A} gets the scheme’s public parameters and n verification keys pk_i ($i \in [n]$) as input. \mathcal{A} may request (up to) one signature for each pk_i , and may finally output a forged signature under exactly one pk_i . We say that a signature scheme TOTS is strongly n -fold one-time (ϵ, t, q) -secure if there is no \mathcal{A} that (ϵ, t, q) -breaks the strong EUF-CMA security of TOTS .*

We now construct a signature scheme TOTS whose n -fold one-time EUF-CMA security experiment reduces to the discrete logarithm problem in \mathbb{G} . The corresponding reduction loses only a factor of 2, independently of n .

$\text{TOTS.Param}(\kappa)$: The common parameters $\text{pars}_{\text{tots}}$ are a two generators g, h_0 and a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$.

$\text{TOTS.Gen}(\text{pars}_{\text{tots}}, \text{pars}_{\text{tots}})$: Uniformly choose exponents $\omega_1, s_1 \in \mathbb{Z}_p$ and output $vk_{\text{tots}} := (h_1, c_1) := (g^{\omega_1}, g^{s_1})$ and $sk_{\text{tots}} := (\omega_1, s_1)$.

$\text{TOTS.Sign}(\text{pars}_{\text{tots}}, sk_{\text{tots}}, m)$: Uniformly choose $r_0 \in \mathbb{Z}_p$ and compute $c_0 := g^{H(m)} h_0^{r_0}$ and $r_1 = (s_1 - H(c_0)) / \omega_1 \bmod p$, such that $c_1 = g^{H(c_0)} h_1^{r_1}$. Output $\sigma = (r_0, r_1)$.

$\text{TOTS.Vfy}(\text{pars}_{\text{tots}}, vk_{\text{tots}}, m, \sigma)$: Parse $\sigma =: (r_0, r_1)$, and set $c_0 := g^{H(m)} h_0^{r_0}$. If $c_1 = g^{H(c_0)} h_1^{r_1}$, output 1, else 0.

Note that TOTS essentially consists of a two-fold application of Pedersen commitments [43], interpreted as one-time signatures.

Lemma 1. *Let $n \in \mathbb{N}$. Then scheme TOTS above is strongly n -fold one-time EUF-CMA secure assuming H is collision-resistant and the discrete logarithm assumption in \mathbb{G} holds. Concretely, $\epsilon_{n\text{-cma}} \leq 2\epsilon_{\text{dlog}} + \epsilon_{\text{crhf}}$ for the advantage $\epsilon_{n\text{-cma}}$ of an arbitrary n -fold one-time EUF-CMA adversary \mathcal{A} , and the advantages ϵ_{dlog} of a corresponding DLOG-solver \mathcal{B} and ϵ_{crhf} of a H -collision-finder \mathcal{C} .*

The proof is fairly standard, see [33] for a sketch.

4.2 Our Simulation-Sound NIZK Proof System

We are now ready to describe our proof system for a set S of PPEs. Intuitively, we will prove, using GS NIWI proofs, that either S is satisfiable, or that we know a TSig-signature for a “suitably unique” value (or both). The “suitably unique”

value will be a verification key for a strongly (and tightly) secure one-time signature scheme. Simulated proofs prove the “or” branch of the statement, using TSig ’s signing key. Simulation-soundness (and also soundness) follows from the existential unforgeability of TSig , and from the soundness of GS proofs. A bit more formally, consider the following non-interactive proof system:

$\text{NIZK}.\text{Gen}(\mathbb{G})$ outputs a CRS $crs = (crs_{\text{GS}}, vk, pars_{\text{tots}})$, where (a) crs_{GS} is a binding CRS for DLIN-based GS proofs over \mathbb{G} , (b) vk is a verification key for TSig , (c) $pars_{\text{tots}}$ are parameters for a strongly n -fold EUF-CMA secure one-time signature scheme TOTS , whose security reduction does not depend on n .

$\text{NIZK}.\text{Prove}(crs, S, \vec{x})$ takes as input a CRS $crs = (crs_{\text{GS}}, vk)$, a set S of PPEs, and a satisfying assignment $\vec{x} = (x_1, \dots, x_\nu) \in \mathbb{Z}_p^\nu$. Then, $\text{NIZK}.\text{Prove}$ samples a TOTS keypair $(vk_{\text{tots}}, sk_{\text{tots}})$ and outputs $\pi = (\pi_{\text{GS}}, vk_{\text{tots}}, \sigma_{\text{tots}})$. Here, π_{GS} is a GS proof (using CRS crs_{GS}) for the set $OR(S, S_{vk, vk_{\text{tots}}}^{\text{TSig}})$ of PPEs (as described above), and σ_{tots} is a TOTS -signature under vk_{tots} of π_{GS} .

$\text{NIZK}.\text{Vfy}(crs, S, \pi)$ takes a CRS $crs = (crs_{\text{GS}}, vk)$, a set S of PPEs, and a proof π as above, verifies σ_{tots} , and then checks π as a GS proof for the set $OR(S, S_{vk, vk_{\text{tots}}}^{\text{TSig}})$ of PPEs.

Theorem 4. *The proof system NIZK just described is $(\epsilon_{\text{ZK}}, \epsilon_{\text{snd}}, \epsilon_{\text{simsnd}}, t, Q)$ -secure, where $\epsilon_{\text{ZK}} \leq 2|\epsilon_{\text{GS}}|$ and $\epsilon_{\text{snd}}, \epsilon_{\text{simsnd}} \leq \epsilon_{\text{tots}} + \epsilon_{\text{tsig}}$ for the advantages of suitable adversaries on the indistinguishability of hiding and binding GS CRSSs, the strong Q -fold one-time EUF-CMA security of TOTS , and the weak EUF-CMA security of TSig . All constructed adversaries have roughly the same runtime as the zero-knowledge, soundness, resp. simulation-soundness experiments (with adversaries of runtime t).*

5 Tight IND-CCA Security in the Multi-user Setting

A public-key encryption scheme consists of four algorithms $\text{PKE} = (\text{PKE}.\text{Param}, \text{PKE}.\text{Gen}, \text{PKE}.\text{Enc}, \text{PKE}.\text{Dec})$. Parameter generation $\Pi \xleftarrow{\$} \text{PKE}.\text{Param}(\kappa)$ takes as input a security parameter κ and outputs parameters Π . The key generation algorithm $(pk, sk) \xleftarrow{\$} \text{PKE}.\text{Gen}(\Pi)$ generates, on input Π , a public encryption key pk and a secret decryption key sk . The probabilistic encryption algorithm $c \xleftarrow{\$} \text{PKE}.\text{Enc}(pk, m)$ takes as input a public key pk and a message m , and outputs a ciphertext c . The deterministic decryption algorithm $\text{PKE}.\text{Dec}(sk, c) \in \{m, \perp\}$ takes as input a secret key sk and a ciphertext c , and outputs a message m or an error symbol \perp . We require the usual correctness properties.

The following security experiment, played between a challenger and an adversary \mathcal{A} , is based on the multi-user security definition from [9]. The experiment is parametrized by two integers $\mu, q \in \mathbb{N}$.

1. The challenger runs $\Pi \xleftarrow{\$} \text{PKE}.\text{Param}(\kappa)$ once and then $\text{PKE}.\text{Gen}(\Pi)$ μ times to generate μ key pairs $(pk^{(i)}, sk^{(i)})$, $i \in [\mu]$. Then it tosses a coin $b \xleftarrow{\$} \{0, 1\}$, initializes a list $Clist := \emptyset$ to the empty list, and defines a counter $j_i := 0$ for each $i \in [\mu]$.

2. The adversary receives the public keys $pk^{(1)}, \dots, pk^{(\mu)}$ as input. It may query the challenger for two types of operations.

Encryption queries. The adversary submits two messages m_0, m_1 and an index $i \in [\mu]$. If $j_i \geq q$ then the challenger returns \perp . Otherwise it encrypts m_b under public key $pk^{(i)}$ by computing $c = \text{PKE}.\text{Enc}(pk^{(i)}, m_b)$. Then it appends (c, i) to Clist , updates counter j_i as $j_i := j_i + 1$, and returns c .

Decryption queries. The adversary submits a ciphertext c and an index $i \in [\mu]$. If $(c, i) \in \text{Clist}$ then the challenger returns \perp . Otherwise it returns whatever $\text{PKE}.\text{Dec}(sk^{(i)}, c)$ returns.

3. Eventually the adversary \mathcal{A} outputs a bit b' . We say that the adversary *wins* the game, if $b = b'$.

Definition 5. Let \mathcal{A} be an adversary that runs in time t and wins with probability $1/2 + \epsilon$. Then \mathcal{A} (ϵ, t) -breaks the (μ, q) -IND-CCA security of PKE. If \mathcal{A} never asks any decryption query, then \mathcal{A} (ϵ, t) -breaks the (μ, q) -IND-CPA security of PKE. For $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$ we say that PKE is (ϵ, t, μ, q) -IND-ATK secure, if there exists no adversary that (ϵ, t) -breaks the (μ, q) -IND-ATK security of PKE.

Note that for $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$ the classical definitions of IND-ATK security [27, 44] are identical to $(1, 1)$ -IND-ATK security in the above sense. Moreover, the generic reduction from [9] shows that an adversary \mathcal{A} that (ϵ, t) -breaks the (μ, q) -IND-ATK security of public-key encryption scheme PKE implies an adversary \mathcal{A}' that (ϵ', t') -breaks the $(1, 1)$ -IND-ATK security of PKE with $t' \approx t$ and $\epsilon' \geq \epsilon/(q\mu)$. Thus, the generic reduction loses a factor of $q\mu$.

Generic Construction. The construction of the public-key encryption scheme Enc_{CCA} with tight security reduction follows the Naor-Yung paradigm [42, 45, 40]. It uses as building blocks an (IND-CPA secure) public-key encryption scheme $\text{Enc}_{\text{CPA}} = (\text{CPA}.\text{Param}, \text{CPA}.\text{Gen}, \text{CPA}.\text{Enc}, \text{CPA}.\text{Dec})$ and a (simulation-sound) non-interactive zero-knowledge proof system $\text{NIZK} = (\text{NIZK}.\text{Gen}, \text{NIZK}.\text{Prove}, \text{NIZK}.\text{Vfy})$. We let scheme $\text{Enc}_{\text{CCA}} = (\text{CCA}.\text{Param}, \text{CCA}.\text{Gen}, \text{CCA}.\text{Enc}, \text{CCA}.\text{Dec})$ be as follows.

$\text{CCA}.\text{Param}(\kappa)$ generates a common reference string for the NIZK proof system $crs \xleftarrow{\$} \text{NIZK}.\text{Gen}(\kappa)$ for the language $\mathcal{L} := \{(pk_0, pk_1, c_0, c_1)\}$ such that $(pk_0, pk_1, c_0, c_1) \in \mathcal{L}$ if and only if

$$c_0 = \text{CPA}.\text{Enc}(pk_0, m) \wedge c_1 = \text{CPA}.\text{Enc}(pk_1, m).$$

That is, we have $(pk_0, pk_1, vk_{\text{ots}}, c_0, c_1, c_{\Pi}) \in \mathcal{L}$ iff c_0 and c_1 encrypt the same message m .

$\text{CCA}.\text{Gen}(\Pi)$ generates two key pairs $(pk_0, sk_0), (pk_1, sk_1) \xleftarrow{\$} \text{CPA}.\text{Gen}$ of the public-key encryption scheme. The resulting public key is $pk = (pk_0, pk_1, \Pi)$, the secret key is $sk = sk_0$.

$\text{CCA}.\text{Enc}(pk, m)$ encrypts a message m by computing $c_0 = \text{CPA}.\text{Enc}(pk_0, m)$, $c_1 = \text{CPA}.\text{Enc}(pk_1, m)$, and a proof π that $(pk_0, pk_1, c_0, c_1) \in \mathcal{L}$, using the

encryption randomness of c_0 and c_1 as witness. The resulting ciphertext is $c = (c_0, c_1, \pi)$.

$\text{CCA}.\text{Dec}(sk, c)$ decrypts a given ciphertext as follows. First it checks whether $(pk_0, pk_1, c_0, c_1) \in \mathcal{L}$ by verifying the proof π . If false, then it returns \perp . Otherwise it computes and returns $m = \text{CPA}.\text{Dec}(sk_0, c_0)$.

It is a classical result [45] that the above encryption scheme is $(1, 1)$ -IND-CCA secure, if Enc_{CPA} is $(1, 1)$ -IND-CPA secure and NIZK is one-time simulation sound. In the sequel we generalize this to showing that the (μ, q) -IND-CCA security of Enc_{CCA} reduces *tightly* (i.e., independent of μ and q) to the (μ, q) -IND-CPA security of Enc_{CPA} and the μq -security of NIZK .

We remark that our NIZK proof system from Section 4 inherits the (witness)-extractability of Groth-Sahai proofs. Hence, one could think of treating the NIZK system NIZK as one instance of an IND-CPA secure PKE scheme (with the extraction trapdoor as decryption key). It would seem natural to expect that a variant of scheme Enc_{CCA} above with only one Enc_{CPA} instance might be IND-CCA secure. We do not know if this holds, however: concretely, to show witness-indistinguishability of our proof system NIZK , we will at some point need to switch NIZK into hiding mode. In this mode, no extraction trapdoor exists, and it is unclear how to go answer decryption queries for Enc_{CCA} .

Theorem 5. *Let Enc_{CPA} be $(\epsilon_{\text{CPA}}, t_{\text{CPA}}, \mu, q)$ -IND-CPA secure, and let NIZK be $(\epsilon_{\text{ZK}}, \epsilon_{\text{snd}}, \epsilon_{\text{simsnd}}, t_{\text{NIZK}}, \mu q)$ -secure. Then Enc_{CCA} is (ϵ, t, μ, q) -IND-CCA secure, where t_{NIZK} and t_{CPA} are roughly the runtime of the IND-CCA experiment with an adversary of runtime t , and $\epsilon \leq 2 \cdot (\epsilon_{\text{CPA}} + \epsilon_{\text{ZK}}) + \epsilon_{\text{snd}} + \epsilon_{\text{simsnd}}$.*

In order to instantiate the CCA-secure scheme from the previous section, we finally need an IND-CPA secure encryption scheme with tight security reduction. A well-known construction can be found in [33].

References

- [1] Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
- [2] Abe, M., Haralambiev, K., Ohkubo, M.: Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133 (2010), <http://eprint.iacr.org/>
- [3] Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg (2011)
- [4] Abe, M., Chase, M., David, B., Kohlweiss, M., Nishimaki, R., Ohkubo, M.: Constant-size structure-preserving signatures: Generic constructions and simple assumptions. Cryptology ePrint Archive, Report 2012/285 (2012), <http://eprint.iacr.org/>
- [5] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press (November 1993)

- [6] Bellare, M., Shoup, S.: Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir Without Random Oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (2007)
- [7] Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS, pp. 394–403. IEEE Computer Society Press (October 1997)
- [8] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
- [9] Bellare, M., Boldyreva, A., Micali, S.: Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
- [10] Bernstein, D.J.: Proving Tight Security for Rabin-Williams Signatures. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 70–87. Springer, Heidelberg (2008)
- [11] Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
- [12] Boneh, D., Mironov, I., Shoup, V.: A Secure Signature Scheme from Bilinear Maps. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 98–110. Springer, Heidelberg (2003)
- [13] Camenisch, J., Chandran, N., Shoup, V.: A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
- [14] Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. *Journal of Cryptology* 20(3), 265–294 (2007)
- [15] Cathalo, J., Libert, B., Yung, M.: Group Encryption: Non-interactive Realization in the Standard Model. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 179–196. Springer, Heidelberg (2009)
- [16] Chase, M., Kohlweiss, M.: A domain transformation for structure-preserving signatures on group elements. *Cryptology ePrint Archive*, Report 2011/342 (2011), <http://eprint.iacr.org/>
- [17] Chevallier-Mames, B., Joye, M.: A Practical and Tightly Secure Signature Scheme Without Hash Function. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 339–356. Springer, Heidelberg (2007)
- [18] Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
- [19] Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
- [20] Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. *SIAM Journal on Computing* 30(2), 391–437 (2000)
- [21] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 469–472 (1985)
- [22] Fuchsbauer, G.: Automorphic Signatures and Applications. PhD thesis, ENS, Paris (2010)
- [23] Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

- [24] Gennaro, R., Halevi, S., Rabin, T.: Secure Hash-and-Sign Signatures without the Random Oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
- [25] Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
- [26] Gentry, C., Halevi, S.: Hierarchical Identity Based Encryption with Polynomially Many Levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)
- [27] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
- [28] Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
- [29] Green, M., Hohenberger, S.: Practical Adaptive Oblivious Transfer from Simple Assumptions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 347–363. Springer, Heidelberg (2011)
- [30] Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
- [31] Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
- [32] Hofheinz, D.: All-But-Many Lossy Trapdoor Functions. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 209–227. Springer, Heidelberg (2012)
- [33] Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. Cryptology ePrint Archive (2012), <http://eprint.iacr.org/>
- [34] Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
- [35] Hofheinz, D., Kiltz, E.: Practical Chosen Ciphertext Secure Encryption from Factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
- [36] Joye, M.: An Efficient On-Line/Off-Line Signature Scheme without Random Oracles. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 98–107. Springer, Heidelberg (2008)
- [37] Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM CCS 2003, pp. 155–164. ACM Press (October 2003)
- [38] Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
- [39] Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
- [40] Lindell, Y.: A Simpler Construction of CCA2-Secure Public-Key Encryption Under General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 241–254. Springer, Heidelberg (2003)
- [41] Merkle, R.C.: A Certified Digital Signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)

- [42] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. ACM Press (May 1990)
- [43] Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
- [44] Rackoff, C., Simon, D.R.: Non-interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
- [45] Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS, pp. 543–553. IEEE Computer Society Press (October 1999)
- [46] Schäge, S.: Tight Proofs for Signature Schemes without Random Oracles. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 189–206. Springer, Heidelberg (2011)
- [47] Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)

Efficient Padding Oracle Attacks on Cryptographic Hardware

Romain Bardou^{1,*}, Riccardo Focardi^{2,**}, Yusuke Kawamoto^{3,*},
Lorenzo Simionato^{2,***}, Graham Steel^{1,*}, and Joe-Kai Tsay^{4,*}

¹ INRIA Project ProSecCo, Paris, France

² DAIS, Università Ca' Foscari, Venezia, Italy

³ School of Computer Science, University of Birmingham, UK

⁴ Department of Telematics, NTNU, Norway

Abstract. We show how to exploit the encrypted key import functions of a variety of different cryptographic devices to reveal the imported key. The attacks are padding oracle attacks, where error messages resulting from incorrectly padded plaintexts are used as a side channel. In the asymmetric encryption case, we modify and improve Bleichenbacher's attack on RSA PKCS#1v1.5 padding, giving new cryptanalysis that allows us to carry out the 'million message attack' in a mean of 49 000 and median of 14 500 oracle calls in the case of cracking an unknown valid ciphertext under a 1024 bit key (the original algorithm takes a mean of 215 000 and a median of 163 000 in the same case). We show how implementation details of certain devices admit an attack that requires only 9 400 operations on average (3 800 median). For the symmetric case, we adapt Vaudenay's CBC attack, which is already highly efficient. We demonstrate the vulnerabilities on a number of commercially available cryptographic devices, including security tokens, smartcards and the Estonian electronic ID card. The attacks are efficient enough to be practical: we give timing details for all the devices found to be vulnerable, showing how our optimisations make a qualitative difference to the practicality of the attack. We give mathematical analysis of the effectiveness of the attacks, extensive empirical results, and a discussion of countermeasures.

1 Introduction

Tamper-resistant cryptographic security devices such as smartcards, USB keys, and Hardware Security Modules (HSMs) are an increasingly common component of distributed systems deployed in insecure environments. Such a device must offer an API to the outside world that allows the keys stored on the device to be used for cryptographic functions and permits key management operations,

* Work partially carried out while at LSV, CNRS & INRIA & ENS-Cachan, France.

** Work partially supported by the RAS Project *TESLA: Techniques for Enforcing Security in Languages and Applications*.

*** Now at Google Inc.

but without compromising security. The most commonly used standard for designing cryptographic device interfaces, RSA PKCS#11 [25], is known to have vulnerabilities if the attacker is assumed to have access to the full API, and can therefore make attacks by combining commands in unexpected ways [5, 6, 8]. In this paper, we describe a different way to attack keys stored on the device using only decryption queries performed by a single function, usually the `C_UnwrapKey` function for encrypted key import. These attacks are cryptanalytic rather than purely logical, and hence require multiple command calls to the interface, but the attacker only needs access to one seemingly innocuous command, subverting the typical countermeasure of introducing access control policies permitting only limited access to the interface.

We will show how the `C_UnwrapKey` command from the PKCS#11 API is often implemented on commercially available devices in such a way that it offers a ‘padding oracle’, i.e. a side channel allowing him to see whether a decryption has succeeded or not. We give two varieties of the attack: the first for when the imported key is encrypted under a public key using RSA PKCS#1 v1.5 padding, which is still by far the most common and often the only available mechanism on the devices we obtained, and the second for when the key is encrypted under a symmetric key using CBC and PKCS#5 padding. The first attack is based on Bleichenbacher’s well-known attack [3]. Although commonly known as the ‘million message attack’, in practice Bleichenbacher’s attack requires only about 215 000 oracle calls on average against a 1024 bit modulus when the ciphertext under attack is known to be a valid PKCS#1 v1.5 block. This is however not efficient enough to be practical on low power devices such as smartcards which perform RSA operations rather slowly. We give a modified algorithm which results in an attack which is 4 times faster on average than the original, with a median attack time over 10 times faster. We also show how the implementation details of some devices can be exploited to create stronger oracles, where our algorithm requires only 9400 mean (3800 median) calls to the oracle. At the heart of our techniques is a small but significant theorem that allows not just multiplication (as in the original attack) but also division to be used to manipulate a PKCS#1 v1.5 ciphertext and learn about the plaintext. In the second attack we use Vaudenay’s technique [27] which is already highly efficient. Countermeasures to such chosen ciphertext attacks are well known: one should use an encryption scheme proven to be secure against them. We discuss the availability of such modes in current cryptographic hardware and examine what other countermeasures could be used while such modes are still not available.

In summary, our contributions are the following: i) new results on PKCS#1 v1.5 cryptanalysis that, when combined with the ‘parallel threads’ technique of Klima-Pokorny-Rosa [26] (which on its own contributes a 38% improvement on mean and 52% on median) results in an improved version of Bleichenbacher’s algorithm giving a fourfold (respectively tenfold) improvement in mean (respectively median) attack time compared to the original algorithm (measured over 1000 runs with randomly generated 1024 bit RSA keys and randomly generated conforming plaintexts); ii) demonstration of the attacks on a variety of

cryptographic hardware including USB security tokens, smartcards and the Estonian electronic ID card, where we found various implementations of the oracle, and adapted our algorithm to each one, resulting in attacks with as few as 9400 mean (3800 median) oracle calls on the most vulnerable devices; iii) analysis of the complexity of the attacks, empirical data, and countermeasures. In the full version available online [1], we discuss manufacturer reaction.

In the next section, we describe the padding attacks relevant to this work and describe our modifications to Bleichenbacher's algorithm. The results on commercial devices are described in section 3. We discuss countermeasures in section 4. Finally we conclude with a discussion of future work in section 5.

2 Padding Oracle Attacks

A padding oracle attack is a particular type of side channel attack where the attacker is assumed to have access to an oracle which returns true just when a chosen ciphertext corresponds to a correctly padded plaintext under a given scheme.

2.1 Bleichenbacher's Attack

Bleichenbacher's padding oracle attack, published in 1998, applies to RSA encryption with PKCS#1 v1.5 padding [3]. Let n, e be an RSA public key and d be the corresponding private key, i.e. $n = pq$ and $ed \equiv 1 \pmod{\phi(n)}$. Let k be the byte length of n , so $2^{8(k-1)} \leq n < 2^{8k}$. Suppose we want to encrypt a plaintext block P where P is l bytes long. Under PKCS#1 v1.5 we first generate a pseudorandom non-zero padding string PS which is $k - 3 - l$ bytes long. We allow l to be at most $k - 11$, so there will be at least 8 bytes of padding. The block for encryption is now created as

$$0x00, 0x02, PS, 0x00, P$$

We call a correctly padded plaintext and a ciphertext that encrypts a correctly padded plaintext *PKCS conforming* or just *conforming*. For the attack, imagine, as above, that the attacker has access to an oracle that tells him just when an encrypted block decrypts to give a conforming plaintext, and assume he is trying to obtain the message $m = c^d \pmod{n}$, where c is an arbitrary integer. He is going to choose integers s , calculate $c' = c \cdot s^e \pmod{n}$ and then send c' to the padding oracle. If c' is conforming then he learns that the first two bytes of $m \cdot s$ are $0x00, 0x02$. Hence, if we let $B = 2^{8(k-2)}$, $2B \leq m \cdot s \pmod{n} < 3B$. The idea is to repeat the process for many values of s until only a single plaintext is possible.

2.2 Improving the Bleichenbacher Attack

Let us first review in a little more detail the original attack algorithm. We are trying to obtain message $m = c^d \pmod{n}$ from ciphertext c . In step 1 (Blinding),

we search for a random integer value s_0 such that $c(s_0)^e \bmod n$ is conforming, by accessing the padding oracle. We let $c_0 = c(s_0)^e \bmod n$ and $m_0 = (c_0)^d \bmod n$. Note that $m_0 = ms_0 \bmod n$. Thus, if we recover m_0 we can compute the target m as $m_0(s_0)^{-1} \bmod n$. If the target ciphertext is already conforming, we can set s_0 to 1 and skip this step.

We let $B = 2^{8(k-2)}$. If c_0 is conforming, $2B \leq m_0 < 3B$. Thus, we set the initial set M_0 of possible intervals for the plaintext as $\{[2B, 3B - 1]\}$. In step 2, we search for s_i such that $c(s_i)^e \bmod n$ is conforming. In step 3, we apply the s_i we found to narrow the set of possible intervals M_i containing the value of the plaintext, and in step 4 we either compute the solution or jump back to step 2.

We are interested in improving step 2, i.e. the search for s_i . We give step 2 of the original algorithm below, and omit the other steps.

Step 2a. If $i = 1$ (i.e. we are searching for s_1), search for the smallest positive integer $s_1 \geq n/(3B)$ such that $c_0(s_1)^e \bmod n$ is conforming. It can be shown that smaller values of s_1 never give a conforming ciphertext.

Step 2b. If $i > 1$ and $|M_{i-1}| > 1$, search for the smallest positive integer $s_i > s_{i-1}$ such that $c_0(s_i)^e \bmod n$ is conforming.

Step 2c. If $i > 1$ and $|M_{i-1}| = 1$, i.e. $M_{i-1} = \{[a, b]\}$, choose small r_i, s_i such that

$$r_i \geq 2^{\frac{bs_{i-1}-2B}{n}} \quad \text{and} \quad \frac{2B+r_in}{b} \leq s_i < \frac{3B+r_in}{a}$$

until $c_0(s_i)^e \bmod n$ is conforming. Intuitively, the bounds for s_i derive from the fact that we want $c_0(s_i)^e \bmod n$ conforming, i.e. $2B \leq m_0s_i - r_in < 3B$, for some r_i , and from the assumption $a \leq m_0 \leq b$. As explained in the original paper, the constraint on r_i aims at dividing the remaining interval in half so to maximize search performance.

Some features of the algorithm's behaviour were already known from the original paper. For example, step 2a/b will in general be executed only very few times (in roughly 90% of our trials, step 2b was executed a maximum of once, and in 32% of cases not at all). However, a lot of the expected calls are here, since each time we just search naively for the next s_i , which takes an expected $1/Pr(P)$ calls where $Pr(P)$ is the probability of a random ciphertext decrypting to give a conforming block. Step 2c, meanwhile, is highly efficient, but is only applicable if there is only one interval left. Furthermore it cannot be directly applied to the original interval $\{2B, 3B - 1\}$ (since the bound on r_i, s_i collapses and we end up with the same search as in step 2a). Based on this observation, we devised a new method for narrowing down the initial interval so that 'step 2c-like' reasoning could be applied to speed up the search for s_1 .

Trimming M_0 . First observe that as well as multiplying the value of the decrypted plaintext ($\bmod n$) by some integer s , we can also divide it by an integer t by multiplying the original ciphertext by $t^{-e} \bmod n$. Multiplication modulo n is a group operation on $(\mathbb{Z}_n)^*$, so inverses are unique. If the original plaintext

was divisible by t , the result $m_0 \cdot t^{-1} \bmod n$ will just be m_0/t , otherwise it will be some other value in the group that we in general cannot predict without knowing m_0 . The following holds.

Proposition 1. *Let u and t be two coprime positive integers such that $u < \frac{3}{2}t$ and $t < \frac{2n}{9B}$. If m_0 and $m_0 \cdot ut^{-1} \bmod n$ are PKCS conforming, then m_0 is divisible by t .*

Proof. We have $m_0 u < m_0 \frac{3}{2}t < 3B\frac{3}{2}t < n$. Thus, $m_0 u \bmod n = m_0 u$. Let $x = m_0 \cdot ut^{-1} \bmod n$. We know $x < 3B$ since it is conforming. Thus $xt < 3Bt < n$ and $xt \bmod n = xt$. Now, $xt = xt \bmod n = m_0 u \bmod n = m_0 u$ which implies t divides m_0 .

By Proposition 1, if we find coprime positive integers u and t , $u < \frac{3}{2}t$ and $t < \frac{2n}{9B}$ such that for a PKCS conforming m_0 , $m_0 \cdot ut^{-1} \bmod n$ is also conforming, then we know that m_0 is divisible by t and $m_0 \cdot ut^{-1} \bmod n = m_0 \frac{u}{t}$. As a consequence

$$2B \cdot t/u \leq m_0 < 3B \cdot t/u.$$

Note that since we already know $2B \leq m_0 < 3B$ we can restrict our search to t and u such that $2/3 < u/t < 3/2$. We apply this by constructing a list of suitable fractions u/t that we call ‘trimmers’. In practice, we use a few thousand trimmers and take $t \leq 2^{12}$ as the implementations typically satisfy $n \geq 2^{8k-1}$. For each trimmer u/t , we submit $c_0 u^e t^{-e}$ to the padding oracle. If the oracle succeeds, we can trim the bounds of M_0 .

A large denominator t allows for a more efficient trimming. The trimming process can be thus optimised by taking successful trimming fractions $u_1/t_1, \dots, u_n/t_n$, computing the lowest common multiple t' of t_1, \dots, t_n , using this value as a denominator and then searching for the highest and lowest numerators u_h, u_l that imply a valid padding, giving $2B \cdot t'/u_l \leq m < 3B \cdot t'/u_h$.

Skipping Holes. In the original algorithm step 2a, the search for the first s_1 starts at the value $\lceil n/3B \rceil$. However, note that to be conforming we require in fact that $m \cdot s \geq n + 2B$. Since $3B - 1 \geq m$ we get $(3B - 1)s \geq n + 2B$. So we can start with $s = \lceil (n + 2B)/(3B - 1) \rceil$. On its own this does not save us much: about 8000 queries depending on the exact value of the modulus. However, when we have already applied the trimming rule above to reduce the upper bound on M_0 to some b , this translates immediately into a better start bound for s_1 of $(n + 2B)/b$.

Observe that in general for a successful s we must have $2B \leq ms - jn < 3B$ for some natural number j . Given that we have trimmed the first interval M_0 to the range $[a, b]$, this gives us a series of bounds

$$\frac{2B + jn}{b} \leq s < \frac{3B + jn}{a}$$

Observe further that when

$$\frac{3B + jn}{a} < \frac{2B + (j + 1)n}{b}$$

we have a ‘hole’ of values where a suitable s cannot possibly be. When used in combination with the trimming rule, we found that we frequently obtain a list of many such holes. We use this list to skip out the holes during the search for the s_1 . Note that this is similar to the reasoning used to calculate s values in step 2c, except that here we are concerned with finding the smallest possible s_1 in order to have the fewest possible intervals remaining when searching for s_2 . As we show in the results below, the combination of the trimming and hole skipping techniques is highly effective, in particular against more permissive oracles than a strict PKCS padding oracle.

2.3 Existing Optimisations

In addition to our original modifications, we also implemented changes proposed by Klima, Pokorny and Rosa (KPR) [26]. These are mainly aimed at improving performance in step 2b, because they were concerned with attacking a weaker oracle where most time was spent in step 2b (see below). They are therefore naturally complementary to our optimisation of step 2a.

Parallel Thread Method. The parallel thread method consists of omitting step 2b in the case where there are several intervals in M_{i-1} , and instead forking a separate thread for each interval and using the method of step 2c to search for s_i . As soon as one thread finds a hit, all threads are halted and the new intervals are calculated. If there is still more than one interval remaining, new threads are launched. In practice, since access to the oracle may not be parallelisable, the actions of each thread can be executed stepwise. This heuristic is quite powerful in practice, as we will see below.

Tighter Bounds and Beta Method. KPR were concerned with attacking the weaker ‘bad version’ oracle found in implementations of SSL patched against the original vulnerability. This meant that when the oracle succeeds, they could be sure of the length of the unpadded plaintext, since it must be the right length for the SSL ‘pre-master secret’. This allowed them to tighten the $2B$ and $3B - 1$ bounds. We also implemented this optimisation where possible, since it has no significant cost, but its effects are not significant. We implemented a further proposal of KPR, the so-called ‘Beta Method’ that we do not have space to describe here, but again found that it caused little improvement in practice.

2.4 Stronger and Weaker Oracles

In order to capture behaviour found in real devices (see section 3), we define stronger and weaker Bleichenbacher oracles, i.e. oracles which return true for a greater or smaller proportion of values x such that $2B \leq x < 3B$. We characterise them by three Booleans specifying the tests they apply or skip on the decrypted plaintext. The first Boolean corresponds to the test for a 0 somewhere after the first ten bytes. The second Boolean corresponds to the check for 0s in

the non-zero padding. The third Boolean corresponds to a check of the plain-text length against some specific value (e.g. 16 bytes for an encrypted AES-128 key). More precisely, we say an oracle is FFF if it returns true only on correctly padded plaintexts of a specific fixed length, like the the KPR ‘bad version’ oracle found in some old versions of SSL. An oracle is FFT if it returns true on a correctly padded plaintext of any length. This is the standard PKCS oracle used by Bleichenbacher. An oracle is FTT if it returns true on a correctly padded plaintext of any length and additionally on an otherwise correctly padded plain-text containing a zero in the eight byte padding. An oracle is TFT if it returns true on a correctly padded plaintext of any length and on plaintexts containing no 0s after the first byte. The most permissive oracle, TTT, returns true on any plaintext starting with 0x00, 0x02. We will see in the next section how all these oracles arise in practice.

In Table 1, we show performance of the standard Bleichenbacher algorithm on these oracles, apart from FFF for which it is far too slow to obtain meaningful statistics. Attacking the strongest oracles TTT and TFT is substantially easier than the standard oracle. We can explain this by observing that for the original oracle, on a 1024 bit block, the probability $Pr(P)$ of a random ciphertext decrypting to give a conforming block is equal to the probability that the first two blocks are 0x00, 0x02, the next 8 bytes are non-zero, and there is a zero somewhere after that. We let $Pr(A)$ be the probability that the first two bytes are 0x00, 0x02, i.e $Pr(A) \approx 2^{-16}$. We identify $Pr(P|A)$, the probability of a ciphertext giving a valid plaintext provided the first two bytes are 0x00, 0x02, i.e

$$\left(\frac{255}{256}\right)^8 \cdot \left(1 - \left(\frac{255}{256}\right)^{118}\right) \approx 0.358$$

$Pr(P)$ is therefore $0.358 \cdot 2^{-16}$. Bleichenbacher estimates that, if no blinding phase is required, the attack on a 128 byte plaintext will take

$$2/Pr(P) + 16 \cdot 128/Pr(P|A)$$

oracle calls. So we have

$$(2 \cdot 2^{16} + 16 \cdot 128)/Pr(P|A) = 371843$$

In the case of, say, the TTT oracle, $Pr(P|A)$ is 1, since any block starting 0x00, 0x02 will be accepted. Hence we have

$$2^{17} + 16 \cdot 128 = 133120$$

oracle queries. This is higher than what we were able to achieve in practice in both cases, but the discrepancy is not surprising since the analysis Bleichenbacher uses is a heuristic approximation of the upper bound rather than the mean. However, it gives an explanation of why the powerful oracle gives such a big improvement in run times: improvements in the oracle $Pr(P|A)$ make a

multiplicative difference to the run time. Additionally, the expected number of intervals at the end of step 2a is $\lceil s_1 \cdot B/n \rceil$ [3, p. 7], so if s_1 is less than 2^{16} , the expected number of intervals is one. For the FFT oracle, the expected value of s_1 (calculated as $1/2 \cdot 1/\Pr(P)$) is about 91 500, between 2^{16} and 2^{17} , whereas for TTT it is 2^{15} . That means that in the TTT case we can often jump step 2b and go straight to step 2c, giving a total of

$$2^{16} + 16 \cdot 128 = 34816$$

i.e. the TTT oracle is about 10 times more powerful than the FFT oracle, which is fairly close to what we see in practice (our mean for FFT is about 5.5 times that for TTT).

In comparison, if the modulus is 2048 bit long, then $\Pr(P|A) \approx 0.599$. Because the modulus is longer, the probability that 0x00 appears after the 8 non-zero bytes is higher than in the 1024 bit case. Furthermore, following the same argument as above, we obtain that the attack on a 2048 bit plaintext will take about 335 065 calls to the FFT oracle, fewer than in the 1024 bit case. Note however that RSA private key operations slow down by roughly a factor of four when key length is doubled.

Table 1. Performance of the original and modified algorithms

Oracle	Original algorithm		Modified algorithm			
	Mean	Median	Mean	Median	Trimmers	Mean skipped
FFF	-	-	18 040 221	12 525 835	50 000	7 321
FFT	215 982	163 183	49 001	14 501	1 500	65 944
FTT	159 334	111 984	39 649	11 276	2 000	61 552
TFT	39 536	24 926	10 295	4 014	600	20 192
TTT	38 625	22 641	9 374	3 768	500	18 467

2.5 Performance of the Modified Algorithm

Referring again to Table 1, we give a summary of our experiments with our modified algorithm. As well as mean and median, we give the number of trimming fractions tried and the average number of oracle calls saved by the hole skipping modification we presented in section 2.2. Observe that as the oracles become stronger, the contribution of the KPR ‘parallel threads’ method becomes less significant and our hole skipping technique more significant. This is to be expected, since as discussed above, for the stronger oracles, fewer runs need to use step 2b. Similarly, when trimming the first interval M_0 , we find that more fractions can be used because of the more permissive oracle, hence we find more holes to skip. For the most restrictive oracle, FFF, the addition of our trimming method slightly improves on the results of KPR (which were 20 835 297 mean and 13 331 256 median). Note also that the trimming technique contributes more

than just the oracle calls saved by the hole skipping, it also slightly improves performance on all subsequent stages of the algorithm. We know this because we can compare performance using only the parallel threads optimisation, where we obtain a mean of 113 667 and a median of 78 674 (on the FFT oracle). In Figure 1, we give the density distribution for 1000 runs of the original algorithm and our optimised algorithm on the classical FFT oracle, with medians marked. Notice the change in shape: we have a much thinner tail.

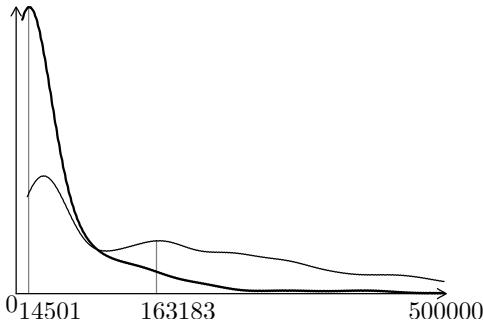


Fig. 1. Graph comparing distribution of oracle calls for original (lower peak, thinner line) and optimised version of the algorithm on the FFT oracle. Median is marked for each.

2.6 Vaudenay’s Attack

Vaudenay’s attack on CBC mode symmetric-key encryption [27] is somewhat simpler and highly efficient. Recall first the operation of CBC mode [9]: given some block cipher with encryption, decryption functions $E(\cdot), D(\cdot)$ and a fixed block size of b bytes, suppose we want to encrypt a message P of length $l = j \cdot b$ for some integer j , i.e. $P = P_1, \dots, P_j$. In CBC mode, we first choose a fresh *initialisation vector* IV . The first encrypted block is defined as $C_1 = E(IV \oplus P_1)$, and subsequent blocks as $C_i = E(C_{i-1} \oplus P_i)$. The need for padding arises because l is not always a multiple of b . Suppose $l = j \cdot b + r$. Then we need to encrypt the last r bytes of the message in a b bytes block in such a way that on decryption, we can recognise that only the first r bytes are to be considered part of the plaintext. One way to do this is the so-called RC5 padding, also known as PKCS padding and described in RFC 5652 [12]. The r bytes are encoded into the leftmost bytes of the final block, and then the final $b - r$ bytes are filled with the value $b - r$. Under this padding scheme, if the plaintext length should happen to be an exact multiple of the block size, then we add a whole block of padding bytes b .

To effect Vaudenay’s attack, suppose that the attacker has some ciphertext C_1, \dots, C_n and access to an oracle that returns true just when a ciphertext decrypts with valid padding. To attack a given block C_i , we first prepend a random block $R = r_1, \dots, r_b$. We then ask the padding oracle to decrypt $R \mid C_i$.

If the padding is valid most probably the final byte is 1, hence the final byte p_m of the plaintext P_i satisfies $p_b = r_b \oplus 1$. If the padding is not accepted, we iterate over i setting $r'_b = r_b \oplus i$ and retrying the oracle until eventually it is accepted. There is a small chance that the final byte of an accepted block is not 1, but this is easily detected. Having discovered the last byte, it is easy to extend the attack to obtain p_{b-1} by tweaking r_{b-1} , and so on for the whole block. Given this ‘block decryption oracle’ we can then apply it to all the blocks of the message. Overall, the attack requires $O(nb)$ steps, and hence is highly efficient.

Since the original attack appeared, many variations have been found on other padding schemes and block cipher modes [2, 7, 14, 17, 20, 22]. Bond and French recently showed that the attack could be applied to the `C_UnwrapKey` command as implemented on a hardware security module (HSM) [4]. We will show in the next section that many cryptographic devices are indeed vulnerable to variants of the attack.

Table 2. Attack Results on Tokens

Device	PKCS#11 version	PKCS#1 v1.5 Attack		CBC-PAD Attack	
		Token	Session	Token	Session
Aladdin eTokenPro	2.01	✓	✓	✓	✓
Feitian ePass 2000	2.11	✗	✗	N/A	N/A
Feitian ePass 3003	2.20	✗	✗	N/A	N/A
Gemalto Cyberflex	2.01	✓	N/A	N/A	N/A
RSA Securid 800	2.20	✓	N/A	N/A	N/A
Safenet Ikey 2032	2.01	✓	✓	N/A	N/A
SATA DKey	2.11	✗	✗	✗	✗
Siemens CardOS	2.11	✓	✓	N/A	N/A

3 Attacking Real Devices

We applied the optimised versions of the attacks of Bleichenbacher and Vaudenay presented in section 2 to the unwrap functionality of PKCS#11 devices. RSA PKCS#11, which describes the ‘Cryptoki’ API for cryptographic hardware, was first published in 1995 (v1.0). The latest official version is v2.20 (2004) which runs to just under 400 pages [25]. Adoption of the standard is almost ubiquitous in commercial cryptographic tokens and smartcards, even if other additional interfaces are frequently offered. In a PKCS#11-based API, applications initiate a *session* with the cryptographic token, by supplying a PIN. Once a session is initiated, the application may access the *objects* stored on the token, such as keys and certificates. Objects are referenced in the API via *handles*, which can be thought of as pointers to or names for the objects. In general, the value of the handle, e.g. for a secret key, does not reveal any information about the actual value of the key. Objects have *attributes*, which may be bitstrings e.g. the value

of a key, or Boolean flags signalling properties of the object, e.g. whether the key may be used for encryption (`CKA_ENCRYPT`¹), or for encrypting other keys, for signing, verification, and other uses. New objects can be created by calling a key generation command, or by *unwrapping* an encrypted key packet using the `C_UnwrapKey` command, which takes a handle, a ciphertext and a *template* as input. A template is a partial description of the key to be imported, giving notably its length. The device attempts to decrypt the ciphertext using the key referred to by the handle. If it succeeds, it creates a new key on the device using the extracted plaintext and the template, and returns a new handle.

Observe that a padding check immediately following the decryption could give rise to an oracle that may be used to determine the value of the newly stored key. To test for such an oracle on a device, we create a key with the `CKA_UNWRAP` attribute set to allow the `C_UnwrapKey` operation, create encrypted key packets with deliberately placed padding errors, call the function on these ciphertexts and observe the return codes. For the case of asymmetric key unwrapping, constructing test ciphertexts is easy since the public key of the pair is always obtainable via a query to the PKCS#11 interface. For symmetric key unwrapping, it is not quite so trivial since the device may create unwrapping keys marked with the Boolean key attribute `CKA_SENSITIVE` which prevents them from being read via the PKCS#11 interface. In this case there are various tricks we can use: we can try to set the attribute `CKA_ENCRYPT` and then use the PKCS#11 function `C_Encrypt` to construct the test packets if a suitable mode is available, or if the device does not allow this, we can explicitly try to create a key with `CKA_SENSITIVE` set to false, assuming the same unwrap algorithm will be used as for sensitive keys. In the event, we were always able to find some way to do this with the devices under test.

3.1 Smartcards and Security Tokens

In Table 2 we give results from implementing the attacks on all the commercially available smartcards and USB tokens we were able to obtain that offer a PKCS#11 interface and support the unwrap operation. A tick means not only that we were able to construct a padding oracle, but that we were actually able to execute the attack and extract the correct encrypted key. A cross notes that the attack fails. We explain these failures below. Not applicable (N/A) means that the token did not support the cryptographic mechanisms and/or unwrap modes required for this attack. Note that relatively few devices support unwrap under symmetric key algorithms. We tested the attacks using both token keys and session keys for the unwrapping. The exact semantics of the difference between these key types is not completely clear from the standard: there is an attribute `CKA_TOKEN` which when set to true indicates a token key and when false

¹ Throughout the paper we will refer to commands, attributes, return codes and mechanisms by their names as defined in the PKCS#11 standard, so `C_` prefixes a (cryptoki) command, `CKA_` prefixes a cryptoki attribute, `CKR_` prefixes a cryptoki return code and `CKM_` prefixes a cryptoki mechanism.

indicates a session key. Session keys are destroyed when the session is ended, whereas token keys persist. However, we have noticed that devices often enforce very different policies for token keys and session keys, so it seemed pertinent to test both types.

In Table 3 we give the class of padding oracle found in each device in the PKCS#1 v1.5 case. To obtain this table we construct padded plaintexts with a single padding error and observed the return code from the token. Note that we give separate entries for token and session keys in this table only when there is a difference in the device’s behaviour in the two cases. We report median attack time, computed from the results of table 1 and from a measure of the unwrap rate of the hardware. Notice how the tenfold improvement in median attack time of our modified algorithm makes attacks even against FFT oracles on slow devices quite practical. Unwrap calls using session keys are often many times faster than token keys though it is not clear why, unless perhaps these devices are carrying out session key operations in the driver software rather than on the card.

We will briefly discuss each line of Table 2 in turn. The **Aladdin eToken Pro** supports both unwrapping modes required, though the **CBC_PAD** unwrap mode does not conform to the standard: a block containing a final byte of 0x00 is accepted. According to the standard, if the final byte of the plaintext is zero and it falls at the end of a block, then an entire block of padding should be added (see section 2). This causes a small problem for the attack since it gives us an extra possibility for the last byte, but we easily adapted the attack to take account of this. The PKCS#1 v1.5 padding implementation ignores zeros in the first 8 bytes of the padding and gives a separate error when the length of the extracted key does not match the requested one (**CKR_TEMPLATE_INCONSISTENT**). Based on this we can build an FTT oracle. The **Feitian** tokens do not support **CBC_PAD** modes. They also do not implement PKCS#1 v1.5 padding correctly: in our tests, any block with 0x02 in the second byte was accepted, except for very large values (e.g. for one key, anything between 0x00 and 0xE2 in the first byte was accepted). The result is that the attack does not succeed. The **Gemalto Cyberflex** smartcard does not allow unwrapping under symmetric keys. However, it seems to implement standard PKCS#1 v1.5 padding correctly, and the Bleichenbacher attack succeeds (FFT oracle, since the length is ignored). The **RSA SecurID** device does not support unwrapping using symmetric keys, hence the Vaudenay attack is not possible. However, the Bleichenbacher attack works perfectly. In fact, the RSA token implements a perfect TTT oracle. The device also supports OAEP, but not in a way that prevents the attack (see next paragraph). The **Safenet ikey2032** implements an asymmetric key unwrapping. The padding oracle derived is more accepting than the Bleichenbacher oracle since the 0s in the first 8 bytes of the padding string are ignored (FTT oracle). The **SATA DKey** does not implement standard padding checks. In **CBC_PAD** mode, only the last byte is checked: it seems that as long as the last byte n is less than the number of bytes in a block, the padding is accepted and the final n bytes discarded. This means we cannot use the attack to recover the whole

key, just the final byte. In PKCS#1 v1.5 mode, many incorrectly padded blocks were accepted, and we were unable to deduce the rationale. For example, any block with the *first* byte equal to 0x02 is accepted. The wide range of accepted blocks prevents the attack. The **Siemens CardOS** supports only unwrapping under asymmetric keys. The Bleichenbacher attack works perfectly: with token keys the oracle is TTT, while with session keys it is FFT.

Table 3. Oracle Details and Median Attack Times

Device	Token		Session	
	Oracle	Time	Oracle	Time
Aladdin eTokenPro	FTT	21m	FTT	17m
Gemalto Cyberflex	FFT	92m	N/A	N/A
RSA Securid 800	TTT	13m	N/A	N/A
Safenet Ikey 2032	FTT	88m	FTT	17m
Siemens CardOS	TTT	21m	FFT	89s

Attacking OAEP Mode Unwrapping. A solution to the Bleichenbacher attack is to use OAEP mode encryption, which was first added to PKCS#1 in v2.0 (1998) and is recommended for all new applications since v2.1 (2002). RSA OAEP was included as a mechanism in PKCS#11 in version 2.10 (1999). However, out of the tokens tested (all of which are currently available products), only one, the RSA SecureID, supports OAEP encryption. The standard PKCS#1 v2.1 notes that it is dangerous to allow two mechanisms to be enabled on the same key [24, p. 14], since “an opponent might be able to exploit a weakness in the implementation of RSAES-PKCS1-v1_5 to recover messages encrypted with either scheme.”. An examination of the developer’s manual for the RSA SecurID reveals that for private keys generated by the token, the relevant attribute “CKA_ALLOWED_MECHANISMS is always set to the following mechanism list : CKM_RSA_PKCS, CKM_RSA_PKCS_OAEP, and CKM_RSA_X_509.”. We created a key wrapped under OAEP and then performed Bleichenbacher’s attack on it using a PKCS#1 v1.5 unwrap oracle. The attack is only slightly complicated by the fact that the initial encrypted block does not yield a valid block when decrypted, requiring us to use the ‘blinding phase’ where many ciphertexts are derived from the original to obtain one that passes the padding oracle. In our tests this added only a few hundred seconds to the attack.

3.2 HSMs

Hardware Security Modules are widely used in banking and similar sectors where a large amount of cryptographic processing has to be done securely at high speed (verifying PIN numbers, signing transactions, etc.). A typical HSM retails for around 20 000 Euros hence is unfortunately too expensive for our laboratory

budget. HSMs process RSA operations at considerable speed: over 1000 decryptions per second for 1024 bit keys. Even in the case of the FFF oracle, which requires 12 000 000 queries, this would result in a median attack time of 12 000 seconds, or just over three hours.

We hope to be able to give details of HSM testing soon in the online version of the paper [1].

3.3 Estonian ID Card

Estonia's Citizenship and Migration Board completed the issuing of more than 1 million national electronic ID (eID) cards in 2006 [16]. The eID is the primary national identification document in Estonia and it is mandatory for all Estonian citizens and alien residents 15 years and older to have one [10]. The card contains two RSA key pairs [13]. One key pair is intended to be mainly used for authentication (e.g., for mutual authentication with TLS/SSL) but can also be used for encrypting and signing email (e.g., with S/MIME). The other key pair is attributed only to be used for digital signatures. Only this latter key pair can be used for legally binding digital signatures [16]. Since January 1, 2011, the eID cards contain 2048 bit RSA keys, therefore these cards comply with NIST's recommendation [18]. However, cards issued before January 1, 2011 continue to use 1024 bit keys.

Attack Vector. Unlike the cryptographic devices discussed above, the Estonian eID card does not allow the import of keys, so our attack here does not rely on the unwrap operation. Instead we consider attacks using the padding oracle provided by the decryption function of the DigiDoc software, part of the official ID software package developed by the Estonian Certification Center, Estonia's only CA [11]. We note that the attack succeeds with any application that returns whether decryption with the eID card succeeds. Our experiments were conducted using the Java library of DigiDoc, called *JDigidoc*. DigiDoc encrypts data using a hybrid encryption scheme, where a 128-bit AES key is encrypted under a public key. First we tested the Estonian ID card's decryption function using raw PKCS#11 calls and confirmed that it checks padding correctly. We then observed that with the default configuration, when attempting to decrypt, e.g., an encrypted email, JDigiDoc writes a log file of debug information that includes the padding errors for the 128-bit AES key that is encrypted under the public key. Any application built on JDigiDoc, that reveals whether decryption succeeds, e.g., by leaking the contents of the log file, provides an attacker with a suitable padding oracle. The information in JDigiDoc's log file gives an attacker access to essentially an FFT oracle but with additional length information. The length information allows us to adjust the $2B$ and $3B - 1$ bounds used in the attack, though in our experiments this made little difference.

In tests, the Estonian ID card, using 2048 bit keys, was able to perform 100 decryptions in 340 seconds. This means that for our optimised attack, where 28 300 decryptions are required, we would need about 96 200 seconds, or about 27 hours to decrypt an arbitrary valid ciphertext. For ID cards using 1024 bit

keys, each decryption should be four times faster, while 49 000 decryptions are required; therefore we estimate a time of about 41 700 seconds, or about 11 hours and 30 minutes to decrypt an arbitrary valid ciphertext. To forge a signature, we require, due to the extra blinding step, a mean of 109 000 oracle calls and a median of 69 000 oracle calls to get a valid signature on an arbitrary message, giving an expected time of 103 hours on a 2048 bit Estonian eID. On a card using 1024 bit keys, we require a mean of 203 000 calls and a median of 126 000 calls; therefore expect to sign an arbitrary message in around 48 hours.

4 Countermeasures

A general countermeasure to the Bleichenbacher and Vaudenay attacks has been well known for years: use authenticated encryption. There are no such modes for symmetric key encryption in the current version of PKCS#11, but version 2.30, which is still at the draft stage, includes GCM and CCM (mechanisms CKM_AES_GCM and CKM_AES_CCM). While these modes have their critics [23], they do in theory provide secure authenticated encryption and hence could form the basis of secure symmetric key unwrap mechanisms. Unfortunately, in the current draft (v7), they are given only as modes for `C_Encrypt`. Adoption of these modes for `C_UnwrapKey` would provide a great opportunity to give the option of specifying authenticated data along with the encrypted key to allow secure transfer of attributes between devices. This would greatly enhance the flexibility of secure configurations of PKCS#11. To prevent the Bleichenbacher attack one must simply switch to OAEP, which is already in the standard. PKCS#11 should follow PKCS#1's long-held position of recommending OAEP exclusively for all new applications. Care must also be taken to remind developers not to allow the two modes to be used on the same key, as is the case in RSA's own SecureID device. In fact, the minutes of the 2003 PKCS workshop suggest that there was a consensus to include the single mechanism recommendation in version 2.20 [21], but it does not appear in the final draft. Note that care must be taken when implementing OAEP as otherwise there may also be a padding oracle attack which is even more efficient than our modified Bleichenbacher attack [15], though we are yet to find such an oracle on a PKCS#11 device.

If unauthenticated unwrap modes need to be maintained for backwards compatibility reasons, there are various options available. For the CBC case, Black and Urtubia note that the 10^* padding, where the plaintext is followed by a single 1 bit and then only 0 bits until the end of the block, leaks no information from failed padding checks while still allowing length of the plaintext to be determined unambiguously [2]. Paterson and Watson suggest a refinement that additionally preserves a notion of indistinguishability, by ensuring that no padded blocks are invalid [19]. They also give appropriate security proofs for the two schemes. If PKCS#1 v1.5 needs to be maintained, we have seen that an implementation of the padding check that rejects anything other than a conforming plaintext containing a key of the correct length with a single error code gives the weakest possible (FFF) oracle. This may be enough for some applications, but

one is well advised to remember the maxim that attacks only get better, never worse. An alternative approach would be to adopt ‘SSL style’ countermeasures, proceeding to import a randomly generated key in the case where a block contains invalid padding. However, this may not fix the hole: if an attacker is able to replay the same block and detect that two different keys have been imported, he knows there is a padding error. One could also decide to ignore padding errors completely and always import just the number of bytes corresponding to the size of the key required, but this looks dangerous: if the same block can be passed off as several different kinds of key, this might open the possibility of attacking weaker algorithms to obtain keys for stronger ones. Thus it seems clear that authenticated encryption is by far the superior solution.

We give manufacturer response in the full version of the paper [1]. There is a broad spectrum: while some manufacturers offer mitigations and state a clear need to get authenticated encryption into the standard and adopted as soon as possible, others see their responsibility as ending as soon as they conform to the PKCS#11 standard, however vulnerable it might be.

5 Conclusions

We have demonstrated a modified version of the Bleichenbacher RSA PKCS#1 v1.5 attack that allows the ‘million message attack’ to be carried out in a few tens of thousands of messages in many cases. We have implemented and tested this and the Vaudenay CBC attack on a variety of contemporary cryptographic hardware, enabling us to determine the value of encrypted keys under import. We have shown that the way the `C_UnwrapKey` command from the PKCS#11 standard is implemented on many devices gives rise to an especially powerful error oracle that further reduces the complexity of the Bleichenbacher attack. In the worst case, we found devices for which our algorithm requires a median of only 3 800 oracle calls to determine the value of the imported key. Vulnerable devices include eID cards, smartcards and USB tokens.

While some theoreticians find the lack of a security proof sufficient grounds for rejecting a scheme, some practitioners find the absence of practical attacks sufficient grounds for continuing to use it. We hope that the new results with our modified algorithm will prompt editors to reconsider the inclusion of PKCS#1 v1.5 in contemporary standards such as PKCS#11.

References

1. Bardou, R., Focardi, R., Kawamoto, Y., Simionato, L., Steel, G., Joe-Kai-Tsay: The million message attack in 15 000 messages, or efficient padding oracle attacks on cryptographic hardware. Cryptology ePrint Archive (to appear, 2012), <http://eprint.iacr.org/>
2. Black, J., Urtubia, H.: Side-channel attacks on symmetric encryption schemes: The case for authenticated encryption. In: Boneh, D. (ed.) USENIX Security Symposium, pp. 327–338. USENIX (2002)

3. Bleichenbacher, D.: Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
4. Bond, M., French, G.: Hidden semantics: why? how? and what to do? Presentation at Fourth Analysis of Security APIs Workshop, ASA-4 (July 2010)
5. Bortolozzo, M., Centenaro, M., Focardi, R., Steel, G.: Attacking and fixing PKCS#11 security tokens. In: Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010), Chicago, Illinois, USA. ACM Press (October 2010)
6. Clulow, J.: On the Security of PKCS #11. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 411–425. Springer, Heidelberg (2003)
7. Degabriele, J.P., Paterson, K.G.: On the (in)security of ipsec in mac-then-encrypt configurations. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 493–504. ACM (2010)
8. Delaune, S., Kremer, S., Steel, G.: Formal analysis of PKCS#11. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF 2008), Pittsburgh, PA, USA, pp. 331–344. IEEE Computer Society Press (June 2008)
9. Dworkin, M.: Recommendation for block cipher modes of operation: Modes and techniques. NIST Special Publication 800-38A (December 2001)
10. Estonian Certification Center. The estonian ID card and digital signature concept, principles and solutions (March 2003), http://www.id.ee/public/The_Estonian_ID_Card_and_Digital_Signature_Concept.pdf
11. Estonian Informatics Center. Estonian ID-software,
<https://installer.id.ee/?lang=eng>
12. Housley, R.: Cryptographic Message Syntax (CMS). RFC 5652 (Standard) (September 2009)
13. ID Süsteemide AS. EstEID specification v2.01,
http://www.id.ee/public/EstEID_Spetsifikatsioon_v2.01.pdf
14. Jager, T., Somorovsky, J.: How to break XML encryption. In: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), pp. 413–422 (2011)
15. Manger, J.: A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 230–238. Springer, Heidelberg (2001)
16. Martens, T.: eID interoperability for PEGS, national profile estonia, European Commission's IDABC programme (November 2007),
<http://ec.europa.eu/idabc/en/document/6485/5938>
17. Mitchell, C.J.: Error Oracle Attacks on CBC Mode: Is There a Future for CBC Mode Encryption? In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 244–258. Springer, Heidelberg (2005)
18. National Institute of Standards and Technology. NIST special publication 800-57, recommendation for key management (March 2007),
<http://csrc.nist.gov/publications/PubsSPs.html>
19. Paterson, K.G., Watson, G.J.: Immunising CBC Mode Against Padding Oracle Attacks: A Formal Security Treatment. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 340–357. Springer, Heidelberg (2008)
20. Paterson, K.G., Yau, A.: Padding Oracle Attacks on the ISO CBC Mode Encryption Standard. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 305–323. Springer, Heidelberg (2004)
21. Minutes from the April, 2003 PKCS workshop (2003),
<ftp://ftp.rsa.com/pub/pkcs/03workshop/minutes.txt>

22. Rizzo, J., Duong, T.: Practical padding oracle attacks. In: Proceedings of the 4th USENIX Conference on Offensive Technologies, WOOT 2010, pp. 1–8. USENIX Association, Berkeley (2010)
23. Rogaway, P.: Evaluation of some blockcipher modes of operation (February 2011), <http://www.cs.ucdavis.edu/~rogaway>; Evaluation carried out for the Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan
24. RSA Security Inc., v2.1. PKCS #1: RSA Cryptography Standard (June 2002)
25. RSA Security Inc., v2.20. PKCS #11: Cryptographic Token Interface Standard (June 2004)
26. Klíma, V., Pokorný, O., Rosa, T.: Attacking RSA-Based Sessions in SSL/TLS. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 426–440. Springer, Heidelberg (2003)
27. Vaudenay, S.: Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS... In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 534–545. Springer, Heidelberg (2002)

Public Keys

Arjen K. Lenstra¹, James P. Hughes², Maxime Augier¹, Joppe W. Bos¹,
Thorsten Kleinjung¹, and Christophe Wachter¹

¹ EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland
² Self, Palo Alto, CA, USA

Abstract. We performed a sanity check of public keys collected on the web and found that the vast majority works as intended. Our main goal was to test the validity of the assumption that different random choices are made each time keys are generated. We found that this is not always the case, resulting in public keys that offer no security. Our conclusion is that generating secure public keys in the real world is challenging. We did *not* study usage of public keys.

Keywords: Sanity check, public keys, (batch) factoring, discrete logarithm, Euclidean algorithm, seeding random number generators.

1 Introduction

Various studies have been conducted to assess the state of the current public key infrastructure, with a focus on X.509 certificates (cf. [3]). Key generation standards for RSA (cf. [23]) have been analysed and found to be satisfactory in [19]. In [11] and [26] (and the references therein) several problems have been identified that are mostly related to the way certificates are used. In this paper we complement previous studies by concentrating on computational and randomness properties of actual public keys, issues that are usually taken for granted.

Compared to the collection of certificates considered in [11], where shared public keys are “not very frequent”, we found a much higher fraction of duplicates. We also found public keys that are not related to the Debian OpenSSL vulnerability but that offer no security at all. The existence of such keys may be crypto-folklore, but it was new to us (but see [12]). This is not a disappearing trend, as may be seen by comparing the results in this paper to those reported in [15]. Vulnerabilities of this sort could affect the expectation of security that the public key infrastructure is intended to achieve. We limited our study to collections of public keys and did not consider issues arising while using them.

We summarize our findings, referring to later sections for details. We collected as many openly accessible public keys as possible from the web, while avoiding activities that our system administrators may have frowned upon. In particular we did not capture or analyse any encrypted traffic of digitally signed documents. The set of 11.7 million public keys that we collected contains 6.4 million distinct RSA moduli. The remainder is almost evenly split between ElGamal keys (cf. [8]) and DSA keys (cf. [25]), plus a single ECDSA key (cf. [25]). The frequency of

keys blacklisted due to the Debian OpenSSL vulnerability (cf. [28]) is comparable to [11]; the findings presented below are not related to this vulnerability. All keys were checked for consistency such as compositeness, primality, and (sub)group membership tests. As the sheer number of keys and their provenance precluded extensive cryptanalysis and the sensibility thereof, per key a modest search for obvious weaknesses was carried out as well. These efforts resulted in a small number of inconsistent or weak keys.

A tacit and crucial assumption underlying the security of the public key infrastructure is that during key setup previous random choices are not repeated. In [11,19] public key properties are considered but this issue is not addressed, with [19] nevertheless concluding that

The entropy of the output distribution [of standardized RSA key generation] is always almost maximal, ... and the outputs are hard to factor if factoring in general is hard.

We do not question the validity of this conclusion, but found that it can only be valid if each output is considered in isolation. When combining outputs the above assumption sometimes fails. Among all types of public keys collected (except ECDSA), we found duplicates with unrelated owners. This is a concern because, if these owners find out, they may breach each other's security. Duplication of keys is more frequent in our collection than in the one from [11].

We also stumbled upon RSA moduli, not affected by the Debian OpenSSL vulnerability, that offer no security. Their secret keys are accessible to anyone who redoing our work. Assuming access to the public key collection, this is straightforward compared to more traditional ways to retrieve RSA secret keys (cf. [4,16]). Figure 1 depicts a simplified sketch of the situation and how it may evolve. Our findings, of which we do not and will not publicly present any evidence, are confirmed by independent similar work (cf. [10]). As shown in Section 3 we used a computational method different from the one used in [10].

Section 2 presents our data collection efforts. Sections 3 and 4 describe the counts and calculations performed for the RSA-related data and for the ElGamal, DSA, and ECDSA data, respectively. Section 5 summarizes our findings.

2 Data Collection

Before the data from [7] was generally available, we started collecting public keys from a wide variety of sources, assisted by colleagues and students. We collected *only* public keys, no encrypted data or digitally signed documents (other than digital certificates). This resulted in almost 5.5 million PGP keys and fewer than 0.1 million X.509 certificates. The latter got a boost with [7] and, to a smaller extent, the data from [11]. We did not engage in web crawling, extensive ssh-session monitoring, or other data collection activities that may be perceived as intrusive, aggressive, or unethical. Thus, far more data can be collected than we did (see also [15]).

Skipping a description of our attempts to agree on a sufficiently uniform and accessible representation of the data, by November 2011 the counts had settled

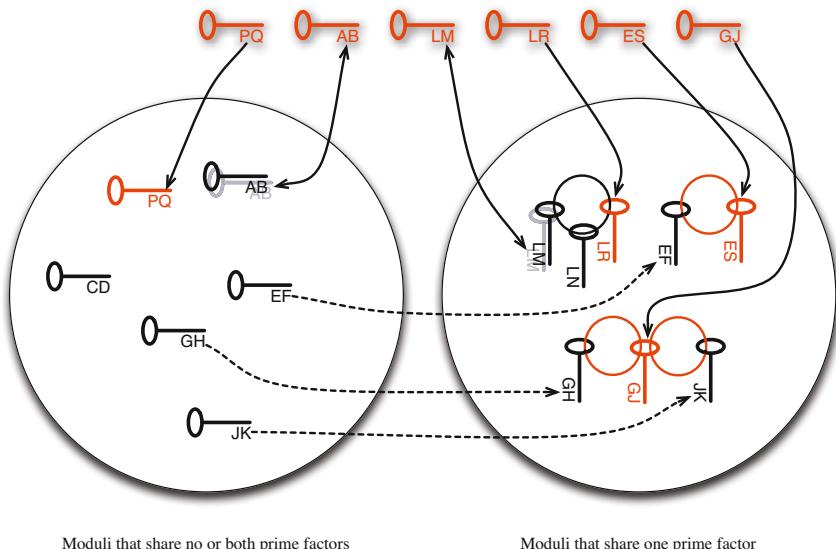


Fig. 1. An existing collection of seven (black) keys is extended with six (red) new keys, where capital letters play the role of (matching) large primes. Initially, keys AB, CD, EF, GH, and JK on the left are secure and keys LM and LN on the right are openly insecure in the same keyring due to the common factor L. New key PQ is secure and appended to the secure list on the left. New key AB duplicates key AB on the left, making both insecure to each other but not to anyone else. New key LM duplicates a key already known to be in the openly insecure group, while key LR results in a new openly insecure modulus on that keyring. Key ES removes known good key EF from the secure keys on the left, resulting in a new openly insecure group on the right consisting of keys EF and ES. Even if the owner of ES now knows that he is insecure and destroys the key, this information can be used by any owners involved to determine the factors of key EF. Key GJ removes two known good keys, GH and JK, from the list of secure keys on the left to form an insecure double keyring on the right (cf. Figure 5 in Section 3). All example keyrings, and many more, occur in the real world. Note that a key that has been dragged from left to right will never be able to return.

as follows: 6 185 372 distinct X.509 certificates (most from the EFF SSL repository, 43 from other sources), and 5 481 332 PGP keys, for a total of at most 11 666 704 public keys. Of the X.509 certificates 6 185 230 are labeled to contain an RSA (modulus, exponent) pair with 141 DSA public keys and a single ECDSA point on the NIST standardized curve `secp384r1` (cf. [2, Section 2.8], [25]). Of the certificates 47.6% have an expiration date later than 2011. About 77.7% of the certifying signatures use SHA1 or better ($5287 \times$ SHA256, $24 \times$ SHA384, $525 \times$ SHA512), 22.3% use MD5 (with $122 \times$ MD2, $30 \times$ GOST, $14 \times$ MD4, and $9 \times$ RIPEMD160). Both requirements, expiration later than 2011 and usage of SHA1-or-2, are met by 33.4% of the certificates.

Table 1. Most frequently occurring RSA public exponents

X.509		PGP		Combined	
e	%	e	%	e	%
65537	98.4921	65537	48.8501	65537	95.4933
17	0.7633	17	39.5027	17	3.1035
3	0.3772	41	7.5727	41	0.4574
35	0.1410	19	2.4774	3	0.3578
5	0.1176	257	0.3872	19	0.1506
7	0.0631	23	0.2212	35	0.1339
11	0.0220	11	0.1755	5	0.1111
47	0.0101	3	0.0565	7	0.0596
13	0.0042	21	0.0512	11	0.0313
65535	0.0011	$2^{127} + 3$	0.0248	257	0.0241
other	0.0083	other	0.6807	other	0.0774

Of the PGP keys 2 546 752 (46.5%) are labeled as ElGamal public keys, 2 536 959 (46.3%) as DSA public keys, the other 397 621 (7.3%) as RSA public keys. PGP keys have no expiration dates or hashes. All public keys were further analysed as described below.

3 RSA

In this section we present the results of various counts and tests that we conducted on the data labeled as RSA public keys. An RSA public key is a pair (n, e) of a supposedly hard to factor RSA modulus n and a public exponent e . The corresponding secret key is the integer d such that $de \equiv 1 \pmod{\varphi(n)}$ or, equivalently, the factorization of n .

Public Exponents. Table 1 lists the ten most frequent public exponents along with their percentage of occurrence for the RSA keys in the X.509 certificates, the PGP keys, and when combined. Except for eight times $e = 1$ and two even e -values among the PGP RSA keys, there is no reason to suspect that the e -values are not functional. Two e -values were found that, due to their size and random appearance, may correspond to a short secret exponent (we have not investigated this). The public exponents are not further regarded below.

Debian Moduli. Two of the n -values, a 1024 and a 2048-bit one each occurring once, were discarded because they could be fully factored using the data from [20] (cf. Debian OpenSSL vulnerability in [28]). A further 30097 n -values (0.48%, with 21459 distinct ones) were found to be blacklisted (cf. [24]), but as their factors were not easily available they were kept.

Shared Moduli. We partition the set of 6 185 228 X.509 certificates into *clusters*, where certificates in the same cluster contain the same RSA modulus. There is a considerable number of clusters containing two or more certificates, each of which could be a security issue; clusters consisting of one certificate, on the

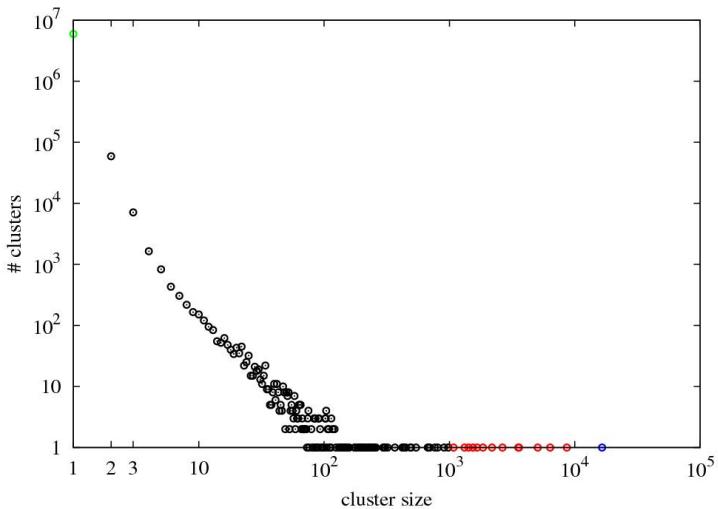


Fig. 2. Number of certificate clusters as a function of the cluster-size

other hand, are the good cases. As depicted in Figure 2, there is one cluster with 16489 certificates (the blue circle on the x -axis), followed by clusters of sizes 8366, 6351, 5055, 3586, 3538, 2645, for a total of 14 clusters with more than a thousand certificates (the red and blue circles on the x -axis; the 5055 share a blacklisted modulus, with no other blacklisted modulus occurring more than seven times). On the other side of the scale the number of good cases is 5 918 499 (the single green circle on the y -axis), with 58913 and 7108 clusters consisting of two and three certificates, respectively. It follows that $6\,185\,228 - 5\,918\,499 = 266\,729$ X.509 certificates (4.3%) contain an RSA modulus that is shared with another X.509 certificate. With 71024 clusters containing two or more certificates it follows that there are $5\,918\,499 + 71024 = 5\,989\,523$ different n -values.

Looking at the owners with shared n -values among the relevant set of 266 729 X.509 certificates, many of the duplications are re-certifications or other types of unsuspicious recycling of the same key material by its supposedly legal owner. It also becomes clear that any single owner may come in many different guises. On the other hand, there are also many instances where an n -value is shared among seemingly unrelated owners. Distinguishing intentionally shared keys from other duplications (which are prone to fraud) is not straightforward, and is not facilitated by the volume of data we are dealing with (as 266 729 cases have to be considered). We leave it as a subject for further investigation into this “fuzzy” recognition problem to come up with good insights, useful information, and reliable counts.

The 397621 PGP RSA keys share their moduli to a much smaller extent: one n -value occurs five times and 27 occur twice. Overall, 28 n -values occur more than once, for a total of 59 occurrences. The n -value that occurs in five PGP keys

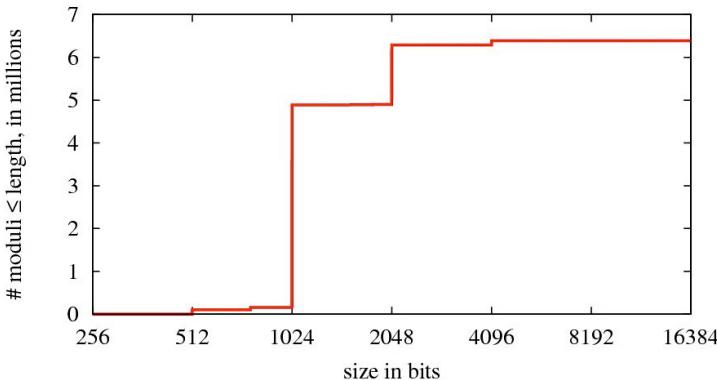


Fig. 3. Cumulative number of modulus sizes for RSA

also occurs twice among the X.509 certificates, and all seven occurrences refer to the same owner. For some of the other 27 multiple occurrences of n -values unique ownership of the RSA keys was harder to assess.

Distinct Moduli. As seen above, we extracted 5 989 523 different n -values from the X.509 certificates. Similarly, $397\,621 - 59 + 28 = 397\,590$ of the PGP n -values are unique. Joining the two sets resulted in 6 386 984 distinct values, with the 129 n -values contained in both sets occurring in 204 X.509 certificates and in 137 PGP keys: as mentioned, some PGP keys are X.509-certified as well (though we have not tried to establish unique or conflicting ownerships, as this already proved to be infeasible for keys shared just among X.509 certificates). In order not to make it easier to re-derive our results, most information below refers to the joined set of unique values, not distinguishing between X.509 and PGP ones.

Modulus Sizes. The cumulative sizes of the moduli in the set of 6 386 984 n -values are depicted in Figure 3. Although 512-bit and 768-bit RSA moduli were factored in 1999 (cf. [1]) and 2009 (cf. [13]), respectively, 1.6% of the n -values have 512 bits (with 0.01% of size 384 and smallest size 374 occurring once) and 0.8% of size 768. Those moduli are weak, but still offer marginal security. A large number of the 512-bit ones were certified after the year 2000 and even until a few years ago. With 73.9% the most common size is 1024 bits, followed by 2048 bits with 21.7%. Sizes 3072, 4096, and 8192 contribute 0.04%, 1.5%, and 0.01%, respectively. The largest size is 16384 bits, of which there are 181 (0.003%).

Primality, Small Factors, and Other Tests. Two of the unique n -values are prime, 171 have a factor $< 2^{24}$ (with 68 even n -values) after removal of which six cofactors are prime. About 25% of the remaining 165 composites were fully factored after a modest search for small factors using the implementation from [29] of the elliptic curve method (ECM, cf. [17]), some of the others may indeed be hard to factor and could, in principle, serve as RSA modulus. Nevertheless, these 173 n -values do not comply with the standards for the generation

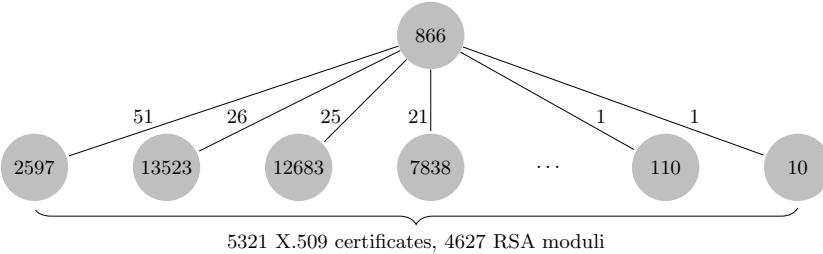


Fig. 4. The largest depth one tree found, on 4628 vertices, with the 512-bit prime p_{866} as root and leaves $p_{2597}, p_{13523}, \dots, p_{10}$. The edges correspond to 4627 distinct 1024-bit RSA moduli, with labels indicating the number of distinct X.509 certificates and PGP keys containing the RSA modulus corresponding to the edge, for a total of 5321 certificates. All certificates have expired and use SHA1 as hash function.

of RSA moduli (cf. [19]) and they were discarded. Nine cases are probably due to copy-and-paste errors, as eight proper moduli were found that differed from the wrong ones in a few hexadecimal positions (two distinct wrong moduli match up with the same correct one).

Fermat’s factorization method, which works well if two factors are close together, did not produce any factors. In particular we found no moduli as reported by Mike Wiener [27] ($n = pq$ with p prime and q the least prime greater than p , and thus with p the largest prime $\leq [\sqrt{n}]$). Neither were there any non-trivial powers.

Moduli with Shared Factors. Moduli that share one prime factor result in complete loss of security for all moduli involved. We discuss the results based on the graphs spawned by the moduli and shared factors. If different users make different choices during key setup, the graph associated to c distinct n -values (cf. Introduction) would consist of c connected components¹, each consisting of a single edge connecting two unidentified – and supposedly unidentifiable – primes. This turned out not to be the case: it took a matter of hours on a single core to find 1995 connected components that each consist of at least two edges. Much larger datasets can be handled without trouble. Our calculation uses a simple-minded binary tree, forming a parent node $\text{lcm}(a, b)$ for leaves a, b while taking appropriate action if $\gcd(a, b) > 1$ and using the subquadratic multiplication and greatest common divisor implementations from [9]. It scales well and is marginally slower than the more contrived gcd-computation described in [10] but uses less memory. On a 1.8GHz i7 processor the straightforward approach would require about ten core-years and would not scale well. Inclusion of the p and q -values from Section 4 and the primes from [20] related to the Debian OpenSSL vulnerability [28] did not produce additional results.

¹ Two distinct vertices are in the same connected component if and only if they are connected by a path consisting of edges in the graph.

Table 2. The s -column indicates the number of depth one trees with ℓ leaves for which all edge multiplicities are equal to one, the m -column the number of trees for which at least one edge occurs at least twice, and $T = s + m$ the total. The bold entry corresponds to the depth one tree depicted in Figure 4.

ℓ	s	m	T	ℓ	s	m	T	ℓ	s	m	T
2	1009	191	1200	13	3	3	6	24	1	1	2
3	259	86	345	14	2	3	5	26	0	1	1
4	95	44	139	15	1	4	5	27	0	1	1
5	43	32	75	16	2	1	3	32	0	1	1
6	23	29	52	17	1	2	3	33	0	1	1
7	20	19	39	18	1	1	2	35	0	2	2
8	13	17	30	19	1	2	3	36	1	1	2
9	4	11	15	20	1	2	3	37	0	1	1
10	3	8	11	21	0	3	3	42	0	1	1
11	3	9	12	22	1	2	3	44	0	2	2
12	3	3	6	23	0	1	1	46	0	1	1

Of the 1995 connected components, 1988 are depth one trees². Of those 1200 have two leaves (i.e., 1200 pairs of n -values, each with a distinct prime factor in common), 345 three leaves, etc., up to a single one with 4627 leaves (i.e., 4627 n -values all with the same prime factor in common). It is not uncommon for an n -value corresponding to an edge of these depth one trees to occur more than once as an RSA modulus: 497 of the 1988 depth one trees have at least one edge that corresponds to an RSA modulus that occurs in at least two X.509 certificates or PGP keys. In the other 1491 depth one trees all edge multiplicities are one. Table 2 lists for each number of leaves ℓ how often each type occurs, with the s -column the number of trees for which all n -values occur once as RSA modulus in an X.509 certificate or PGP key, the m -column the number of trees for which at least one n -value occurs as RSA modulus in at least two X.509 certificates or PGP keys, and the total $T = s + m$. For smaller tree-sizes s is larger, for larger trees multiple occurrence of moduli is more common.

Six of the other seven connected components contain four vertices and three edges, but are not depth one trees. Each of these six components thus consists of a “central” n -value that has a factor in common with each of two other, co-prime n -values. The remaining connected component is the most intriguing – or suspicious – as it is a complete graph on nine vertices (K_9): nine primes, each of whose $\binom{9}{2} = 36$ pairwise products occurs as n -value.

Denoting the primes identified with the vertices of the graph by p_1, p_2, \dots (using an ordering naturally implied by our representation), Figures 4, 5, and 6 depict the largest depth one tree, the six four-vertex components, and the K_9 , respectively, with the edge labels indicating the number of X.509 certificates and PGP keys containing the corresponding n -value as RSA modulus. Note that all moduli in the K_9 occur quite frequently.

² A depth one tree has no cycles and contains one *root* vertex with edges leading to all other vertices, as in Figure 4.

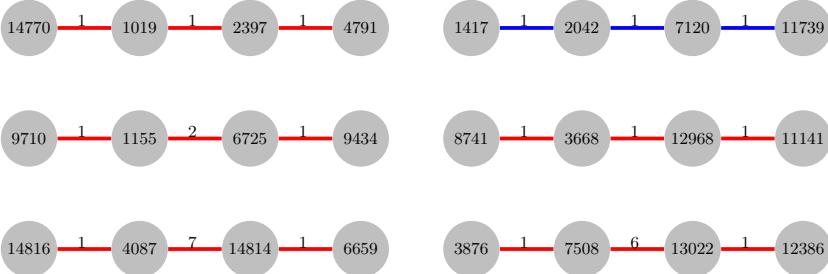


Fig. 5. Six connected components consisting of four vertices, with labels as in Figure 4. The eight primes in the top two components are 512 bits long, the other 16 are 256-bit primes). The red edges correspond to RSA moduli contained in certificates that will not expire anytime soon and that use SHA1 as hash function. The blue ones will expire soon and use MD5.

Any two n -values associated to edges in the same depth one tree can be factored. Two n -values associated to other edges can be factored if the edges are adjacent (i.e., share a vertex), or one finds a path connecting them. For non-adjacent edges in the same connected component from Figure 5 that is the unique central edge, for edges in the K_9 many paths are possible. All required edges are in our set of n -values.

Affected RSA Moduli and Certificates. The 1995 components contain 14901 vertices and 12934 edges: 14901 distinct primes fully factoring 12934 distinct n -values (0.2% of the total), 11699 of which each occurs as RSA modulus in a single X.509 certificate or PGP key, and 1235 occurring more than once in, in total, 9720 certificates and keys. Thus, $11699 + 9720 = 21419$ X.509 certificates and PGP keys are affected. Note that affected moduli are much more frequently shared than non-affected ones. None of the affected moduli are blacklisted.

Of the primes, 14592 are 512 bits long, 307 are 256-bit, and the remaining two have 257 bits. Of the n -values, 214 are 512 bits long, and there are 12720 of 1024 bits. Of the 512-bit n -values thus factored, 47 occur as RSA moduli in 188 X.509 certificates that have not expired and use SHA1. Of the factored 1024-bit n -values, 3201 occur as RSA moduli in 5250 certificates that have not expired and that use SHA1, of which 617 are regular non-self-signed end-user certificates with “CA=false” (with 390 distinct RSA moduli). The majority (4633, with 2811 moduli) has “CA=true” and is self-signed, of which 727 (304 moduli) have been used to certify other RSA moduli (none among our collection of affected moduli). These 727 certificates share their “issuer”-field and the 304 RSA moduli occur in depth one trees not containing moduli owned by others. So, this security issue is reasonably contained. But 4445 of the 5250 certificates (and 537 of the 617 end-user ones) have no relation to that `issuer` and have a wide variety of “issuer” and “subject”-fields. This could be a security concern. We do not and will not reveal what types of users or devices are affected. We note, however,

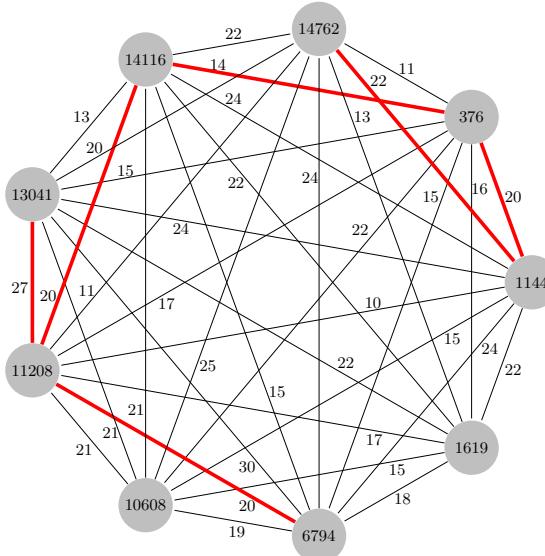


Fig. 6. Connected component consisting of nine vertices, corresponding to primes p_{376} , p_{1144} , ..., p_{14762} (all 512-bit). With labels as in Figure 4, in total 687 X.509 certificates are involved. Six of those certificates have not expired yet, use SHA1 as hash function (as opposed to MD5), and have “CA=false”; the red edges correspond to the RSA moduli contained in those six certificates.

that our data give us more reason for concern than reported elsewhere (cf. [10]) and that affected “flesh and blood” users that we talked to were not pleased³.

Discussion. Generation of a regular RSA modulus consists of finding two random prime numbers. This must be done in such a way that these primes were not selected by anyone else before. The probability not to regenerate a prime is commensurate with the security level if NIST’s recommendation [25, page 53] is followed to use a random seed of bit-length twice the intended security level. Clearly, this recommendation is not always followed.

Irrespective of the way primes are selected (additive/sieving methods or methods using fresh random bits for each attempted prime selection), a variety of obvious scenarios is conceivable where poor initial seeding may lead to mishaps, with duplicate keys a consequence if no “fresh” local entropy is used at all. If the latter is used, the outcome may be worse: for instance, a not-properly-random first choice may be prime right away (the probability that this happens is inversely proportional to the length of the prime, and thus non-negligible) and miss its chance to profit from local entropy to become unique. But local entropy may lead to a second prime that is unique, and thus a vulnerable modulus.

The above may, to some extent, explain the occurrence of duplicate RSA moduli and depth one trees. But we cannot explain the relative frequencies

³ “Donnerwetter!”

and appearance of these mishaps. Neither do we understand how connected components in Figures 5 and 6 may be expected to appear other than by very poor seeding or intentional malfeasance. Also, the great variation of issuers and subjects of the affected X.509 certificates (including the K_9) is disconcerting. No correlation between certification time and vulnerability of keys was detected. Vague, hand-waving arguments suggest that some of the devices involved may have used about 32 bits of entropy.

Avoiding two random choices during RSA modulus generation is straightforward (cf. [14]). But the resulting moduli may have other, as yet unpublished weaknesses (we are not aware of serious ones). It is better to make sure that cryptographic keys are generated only after proper initialization of the source of randomness.

4 ElGamal, DSA, and ECDSA

In this section we present the results of various counts and tests that we conducted on the data labeled as ElGamal, DSA, or ECDSA public keys. In neither collection did we find any of the numbers from [20] (cf. Debian OpenSSL vulnerability [28]).

4.1 ElGamal

An ElGamal public key consists of a triple (p, g, y) where p is prime, g is a generator of the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$ or a subgroup thereof of small index, and y is an element of $\langle g \rangle$. The secret key is an integer $x \in \{0, 1, \dots, p-2\}$ with $g^x = y$.

Correct ElGamal Keys. Among the PGP keys, 2 546 752 are labeled as ElGamal public keys. Three are incomplete and were discarded. Of the remaining triples 82 contain a composite p -value, resulting in 2 546 667 triples with correct p -values. Almost half (38) of the wrong p -values share a pattern with 65.6% of the p -values in the correct ElGamal keys, cf. below.

Restricting to the triples (p, g, y) with prime p -values, a triple is a correct ElGamal public key if $y \in \langle g \rangle$. To verify this the order of g , and thus the factorization of $p - 1$, is needed. This is easy for *safe primes* (i.e., primes p for which $(p - 1)/2$ is prime), but may be hard otherwise. The order of g could be established for 70.8% of the triples (65.6% with safe primes, 5.2% with primes p for which $(p - 1)/(2m)$ is prime and $m > 1$ has only small factors) and could reasonably be guessed for the other 29.2% (almost all with primes p for which $(p - 1)/2$ is composite but has no small factors). For at least 16.4% of the ElGamal keys the g -values do not generate $(\mathbf{Z}/p\mathbf{Z})^*$. This led to 33 failed membership tests $y \in \langle g \rangle$, i.e., an insignificant 0.001% of the triples. Note that if $y \in \langle g \rangle$ a secret key exists; it does not follow that the owner knows it. A handful of triples were identified with peculiar y -values for which it is doubtful if a secret key is known to anyone.

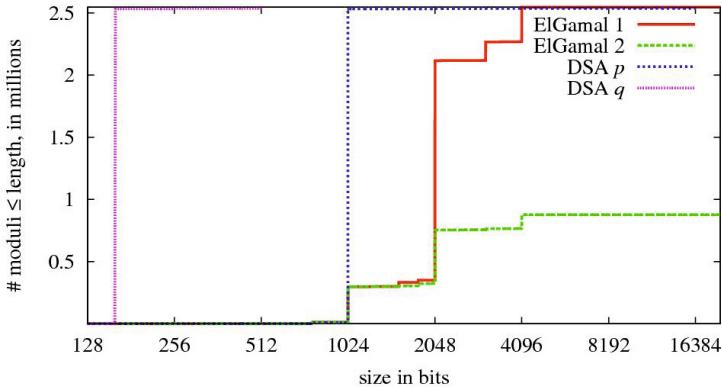


Fig. 7. Cumulative numbers of p and q -sizes in ElGamal and DSA public keys, indicated by “ElGamal 1”, “DSA p ”, and “DSA q ”; cumulative sizes of the distinct ElGamal p -values is indicated by “ElGamal 2”

Shared ElGamal Keys. Six of the ElGamal keys occur twice: two keys with two unrelated owners each, and four keys occurring twice but with the same owner.

ElGamal Key Sizes. Figure 7 depicts the cumulative p -sizes in the set of 2546 628 correct ElGamal keys. There are 1437 different p -sizes, ranging from thrice 256 bits to nine times 16384 and once 20000. Most frequent are 2048 bits (69.3%), 1024 (11.2%), 4096 (10.8%) and 3072 (5.8%) followed by 1536 (1.3%), 1792 (0.7%), 768 (0.4%), and 1025 (0.04%).

Shared Primes, Generators. Primes and generators may be shared. Among the 2546 628 distinct ElGamal keys 876 202 distinct p -values (and distinct (p, g) -pairs) occur. Despite this high duplication rate, only 93 distinct p -values occur more than once. The four most frequent p -values are “similar”. Let $p(x, L)$ denote the least safe prime $\geq x \bmod 2^L$. There is an integer v such that $p(v, L)$ for L -values 2048, 4096, 3072, 1536 occurs as p -value in 52.4%, 6.5%, 5.6%, and 1% of the ElGamal keys, respectively ($p(v, 1024)$ occurs twice, $p(v + 2^{510}, 512)$ once, and as noted above parts of v also occur in incorrect ElGamal keys). We suspect that these p -values, of different sizes, were generated using similar software and identical random seeding (if any), and from the least significant bit up to the most significant one.

All $p(., L)$ -values use $g = 2$, which for $L = 2048$ generates $(\mathbf{Z}/p\mathbf{Z})^*$, but for the others an index two subgroup thereof. Overall, $g = 2$ occurs most frequently (70.8%), followed by $g = 5$ (19.5%), 6 (4.7%), 7 (1.9%), 11 (1.3%), and 13 (0.9%), with a total of 76 distinct g -values. No g -values were found that do not have a large order, but for at least 9.6% of the distinct (p, g) -pairs the g -values do not generate $(\mathbf{Z}/p\mathbf{Z})^*$. We can only give a lower bound because, as pointed out above, we failed to find *any* prime factor of 29.2% of the $(p-1)/2$ -values in the ElGamal

keys (which turns out to be 87.7% of the distinct $(p - 1)/2$ -values in the set of distinct p -values). Thus, we cannot be certain that the corresponding generators were properly chosen; consistent failure of all ECM factoring attempts of these numbers suggests, however, that they were well chosen.

Among the distinct ElGamal keys, all y -values are distinct, which is as expected because distinct (p, g) -pairs have negligible probability to lead to the same y -value (and identical (p, g) -pairs with identical y -values have already been identified and removed). The secret keys, however, may still be the same. But as there is no way to tell if that is the case (for distinct (p, g) -pairs) there is no serious security risk even if they are.

4.2 DSA

A DSA public key is a four-tuple (p, q, g, y) where p and q are primes with q dividing $p - 1$, the element g generates an order q subgroup of the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$, and y is an element of $\langle g \rangle$. The secret key is the integer $x \in \{0, 1, \dots, q - 1\}$ with $g^x = y$.

Correct DSA Keys. Among the PGP keys and X.509 certificates, 2536959 and 141 four-tuples, respectively, are labeled as DSA keys. All four-tuples were first checked for correctness, casting them aside at the first test they failed. The tests were conducted in the following order: T1: primality of p ; T2: primality of q ; T3: divisibility of $p - 1$ by q ; T4: order of g equals q ; and T5: order of y equals q . An insignificant 0.002% (66) of the PGP four-tuples are incorrect with failures $12 \times T1$, $2 \times T2$, $10 \times T4$, and $42 \times T5$ (where T2 failed twice for the same q -value, as it occurred twice). The X.509 DSA four-tuples passed all tests. Some of the failures may be due to transcription errors, as they occur in four-tuples that differ from correct ones in a few hexadecimal positions.

Shared DSA Keys. The remaining 2536893 PGP DSA keys contain very few duplicates: one key occurs thrice (with possibly double ownership) and two keys occur twice each (each with single ownership), resulting in a total of 2536889 distinct PGP DSA keys. Although all 141 X.509 DSA keys are distinct, 95 of them are also among the PGP DSA keys, resulting in a total of $2536889 + 141 - 95 = 2536935$ DSA keys. We have not checked ownerships of these 95 duplicate DSA keys.

DSA Key Sizes. The cumulative p and q -sizes in the set of 2536935 DSA keys are depicted in Figure 7. There are nine different q -sizes: all except 0.2% (5012) are 160, with 256, 224, and 232 the most frequent exceptions occurring 4016, 702, and 249 times, respectively. The smallest and largest q -sizes are 160 and 512, the latter with seven occurrences. With 78 different sizes the variation among p -sizes is larger, though nowhere close to the variation among ElGamal p -sizes. All except 0.6% (15457) of the p -sizes are 1024, with 768, 2048, 3072 and 512 the most frequent exceptions with 9733, 3529, 1468 and 519 occurrences, respectively. The smallest and largest p -sizes are 512 and 16384, the latter with a single occurrence.

Shared Primes, Generators. Distinct DSA keys may contain identical p , q , or g values. In total 2 535 074 distinct p -values occur, with 2 535 037 distinct primes occurring once, 22 occurring twice, five occurring thrice, etc., up to a prime occurring 969 times (note the difference with ElGamal). Not surprisingly (but not necessarily, as the same q -value may give rise to many different p -values), the overall counts and numbers of occurrences are the same for the distinct q -values and the distinct (p, q) -pairs. Although the generator also allows considerable variation, the number of distinct (p, q, g) -triples is the same too. For all except 265 of the unique (p, q, g) -triples, the generator g equals $2^{(p-1)/q}$. We have not been able to determine how the other 265 generators were chosen.

The y -values are all distinct among the distinct DSA keys – given that shared keys were already removed, identical y -values would have been odd indeed. The same remark as above applies concerning identical secret keys.

4.3 ECDSA

The only interesting fact we can report about ECDSA is the surprisingly small number of certificates encountered that contain an ECDSA key (namely, just one), and the small number of certificates signed by ECDSA (one self-signed and a handful of RSA keys). As long as one subscribes to the notion of a standardized curve over a finite field of prime cardinality of a special form, as opposed to a randomly but properly chosen curve over a non-special prime field (cf. [18]), there is nothing wrong with the curve parameters `secp384r1`. It offers (in “theory”) about 192 bits of security which makes it, security-wise, comparable to 8000-bit RSA moduli (n) and ElGamal or DSA finite field sizes (p), and 384-bit DSA subgroup sizes (q).

4.4 ElGamal and (EC)DSA

Not surprisingly, the intersection of the sets of p -values for ElGamal and for DSA is empty. We have not tried hard to retrieve any of the secret exponents, i.e., (for ElGamal and DSA) x -values such that $g^x = y$, but have checked that none is less than 2^{12} in absolute value.

Random Nonces in ElGamal and (EC)DSA. Unlike RSA, during signature generation ElGamal and (EC)DSA require a random nonce that should be entirely unpredictable (cf. [8,21,22]). We are not aware of any studies that verify whether or not the nonces are properly chosen (with the notable exception of [5]). Collecting data for such a study requires a much more intrusive type of data collection and may be considered unethical. Note, however, that a mishap in the form of a poorly chosen nonce affects *only* the party that makes the poor choice, but does not affect any other party. In particular the choice of two identical nonces for distinct ElGamal or DSA parameters does not affect anyone but the two users involved.

Discussion. Both for ElGamal and DSA a small number of keys were identified that are shared among unrelated parties. This may be a security concern. Furthermore, there were some ill-formatted keys that cannot be expected to work

and that should be of insignificant security concern. From the point of view of this paper, the main security concern for ElGamal and (EC)DSA is the generation of the random nonce; this is a key usage but not a key generation issue and therefore beyond the scope of this paper.

5 Conclusion

We checked the computational properties of millions of public keys that we collected on the web. The majority does not seem to suffer from obvious weaknesses and can be expected to provide the expected level of security. We found that on the order of 0.003% of public keys is incorrect, which does not seem to be unacceptable. We were surprised, however, by the extent to which public keys are shared among unrelated parties. For ElGamal and DSA sharing is rare, but for RSA the frequency of sharing may be a cause for concern. What surprised us most is that many thousands of 1024-bit RSA moduli, including thousands that are contained in still-valid X.509 certificates, offer no security at all. This may indicate that proper seeding of random number generators is still a problematic issue.

The lack of sophistication of our methods and findings make it hard for us to believe that what we have presented is new, in particular to agencies and parties that are known for their curiosity in such matters. It may shed new light on NIST's 1991 decision to adopt DSA as digital signature standard as opposed to RSA, back then a "public controversy" (cf. [6]); but note the well-known nonce-randomness concerns for ElGamal and (EC)DSA (cf. Section 4.4) and what happens if the nonce is not properly used (cf. [5]).

Factoring one 1024-bit RSA modulus would be historic. Factoring 12720 such moduli is a statistic. The former is still out of reach for the academic community (but anticipated). The latter comes as an unwelcome warning that underscores the difficulty of key generation in the real world.

Acknowledgements. We thank Peter Eckersley from EFF for his invaluable assistance and Don Johnson for pointing out [12] to us. We gratefully acknowledge key collection and other efforts by Benne de Weger, Philippe Joye, Paul Leyland, Yann Schoenberger, Christoph Schuba, Deian Stefan, Fabien Willemin, Maryam Zargari and others that need to remain anonymous. The first author appreciates the legal advice from Mrs. Chardonnens. James P. Hughes participated in this project in his individual capacity without sponsorship of his employer or any other party. This work was supported by the Swiss National Science Foundation under grant number 200020-132160.

References

1. Cavallar, S., Dodson, B., Lenstra, A.K., Lioen, W., Montgomery, P.L., Murphy, B., te Riele, H., Aardal, K., Gilchrist, J., Guillerm, G., Leyland, P., Marchand, J., Morain, F., Muffett, A., Putnam, C., Putnam, C., Zimmermann, P.: Factorization of a 512-Bit RSA Modulus. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 1–18. Springer, Heidelberg (2000)

2. Certicom Research. Standards for efficient cryptography 2: Recommended elliptic curve domain parameters. Standard SEC2, Certicom (2000)
3. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (2008)
4. Coppersmith, D.: Modifications to the number field sieve. *Journal of Cryptology* 6(3), 169–180 (1993)
5. Darkmirage. PS3 completely cracked (2011), <http://www.darkmirage.com/2011/01/06/ps3-completely-cracked/>
6. Desmedt, Y., Landrock, P., Lenstra, A.K., McCurley, K.S., Odlyzko, A.M., Rueppel, R.A., Smid, M.E.: The Eurocrypt'92 Controversial Issue: Trapdoor Primes and Moduli. In: Rueppel, R.A. (ed.) *EUROCRYPT 1992*. LNCS, vol. 658, pp. 194–199. Springer, Heidelberg (1993)
7. Electronic Frontier Foundation. EFF SSL Observatory (2010), <https://www.eff.org/observatory>
8. El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
9. Free Software Foundation, Inc. GMP: The GNU Multiple Precision Arithmetic Library (2011), <http://www.gmplib.org/>
10. Heninger, N.: New research: There's no need to panic over factorable keys—just mind your Ps and Qs (2012), <https://freedom-to-tinker.com/blog/nadia/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs>
11. Holz, R., Braun, L., Kammenhuber, N., Carle, G.: The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC 2011, pp. 427–444. ACM (2011)
12. Johnson, D.B.: ECC, future resiliency and high security systems. Certicom Whitepaper (1999), <http://www.comms.engg.susx.ac.uk/fft/crypto/ECCFut.pdf>
13. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., te Riele, H., Timofeev, A., Zimmermann, P.: Factorization of a 768-Bit RSA Modulus. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 333–350. Springer, Heidelberg (2010)
14. Lenstra, A.K.: Generating RSA Moduli with a Predetermined Portion. In: Ohta, K., Pei, D. (eds.) *ASIACRYPT 1998*. LNCS, vol. 1514, pp. 1–10. Springer, Heidelberg (1998)
15. Lenstra, A.K., Hughes, J.P., Augier, M., Bos, J.W., Kleinjung, T., Wachter, C.: Ron was wrong, Whit is right. *Cryptology ePrint Archive*, Report 2012/064 (2012), <http://eprint.iacr.org/>
16. Lenstra, A.K., Lenstra Jr., H.W. (eds.): *The development of the number field sieve*. Lecture Notes in Mathematics, vol. 1554. Springer, Berlin (1993)
17. Lenstra Jr., H.W.: Factoring integers with elliptic curves. *Annals of Mathematics* 126(3), 649–673 (1987)
18. Lochter, M., Merkle, J.: Elliptic curve cryptography (ECC) brainpool standard curves and curve generation. RFC 5639 (2010)
19. Loebenberger, D., Nüsken, M.: Analyzing Standards for RSA Integers. In: Nitaj, A., Pointcheval, D. (eds.) *AFRICACRYPT 2011*. LNCS, vol. 6737, pp. 260–277. Springer, Heidelberg (2011)

20. Moore, H.D.: Debian OpenSSL Predictable PRNG Toys (2008),
<http://digitaloffense.net/tools/debian-openssl/>
21. Nguyen, P.Q., Shparlinski, I.: The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology* 15(3), 151–176 (2002)
22. Nguyen, P.Q., Shparlinski, I.: The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Design, Codes Cryptography* 30(2), 201–217 (2003)
23. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 120–126 (1978)
24. Tomášek, J., et al.: Blacklisted moduli, <http://mirror.switch.ch/ftp/mirror/debian/pool/main/o/openssl-blacklist/>,
<http://pocitace.tomasek.cz/debian-randomness/index.html>
25. U.S. Department of Commerce/National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS-186-3 (2009),
http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf
26. Vratonjic, N., Freudiger, J., Bindschaedler, V., Hubaux, J.-P.: The inconvenient truth about web certificates. In: The Workshop on Economics of Information Security, WEIS (2011)
27. Wiener, M.J.: Personal communication (1992)
28. Yilek, S., Rescorla, E., Shacham, H., Enright, B., Savage, S.: When private keys are public: results from the 2008 debian OpenSSL vulnerability. In: Feldmann, A., Mathy, L. (eds.) Internet Measurement Conference, pp. 15–27. ACM (2009)
29. Zimmermann, P., et al.: GMP-ECM (elliptic curve method for integer factorization) (2012), <https://gforge.inria.fr/projects/ecm/>

Multiparty Computation from Somewhat Homomorphic Encryption

Ivan Damgård¹, Valerio Pastro¹, Nigel Smart², and Sarah Zakarias¹

¹ Department of Computer Science, Aarhus University

² Department of Computer Science, Bristol University

Abstract. We propose a general multiparty computation protocol secure against an active adversary corrupting up to $n - 1$ of the n players. The protocol may be used to compute securely arithmetic circuits over any finite field \mathbb{F}_{p^k} . Our protocol consists of a preprocessing phase that is both independent of the function to be computed and of the inputs, and a much more efficient online phase where the actual computation takes place. The online phase is unconditionally secure and has total computational (and communication) complexity linear in n , the number of players, where earlier work was quadratic in n . Moreover, the work done by each player is only a small constant factor larger than what one would need to compute the circuit in the clear. We show this is optimal for computation in large fields. In practice, for 3 players, a secure 64-bit multiplication can be done in 0.05 ms. Our preprocessing is based on a somewhat homomorphic cryptosystem. We extend a scheme by Brakerski et al., so that we can perform distributed decryption and handle many values in parallel in one ciphertext. The computational complexity of our preprocessing phase is dominated by the public-key operations, we need $O(n^2/s)$ operations per secure multiplication where s is a parameter that increases with the security parameter of the cryptosystem. Earlier work in this model needed $\Omega(n^2)$ operations. In practice, the preprocessing prepares a secure 64-bit multiplication for 3 players in about 13 ms.

1 Introduction

A central problem in theoretical cryptography is that of secure multiparty computation (MPC). In this problem n parties, holding private inputs x_1, \dots, x_n , wish to compute a given function $f(x_1, \dots, x_n)$. A protocol for doing this securely should be such that honest players get the correct result and this result is the only new information released, even if some subset of the players is controlled by an adversary.

In the case of *dishonest majority*, where more than half the players are corrupt, unconditionally secure protocols cannot exist. Under computational assumptions, it was shown in [6] how to construct UC-secure MPC protocols that handle the case where all but one of the parties are actively corrupted. The public-key machinery one needs for this is typically expensive so efficient solutions are hard to design for dishonest majority. Recently, however, a new approach has been proposed making such protocols more practical. This approach works as follows: one

first designs a general MPC protocol in the *preprocessing model*, where access to a “trusted dealer” is assumed. The dealer does not need to know the function to be computed, nor the inputs, he just supplies raw material for the computation before it starts. This allows the “online” protocol to use only cheap information theoretic primitives and hence be efficient. Finally, one implements the trusted dealer by a secure protocol using public-key techniques, this protocol can then be run in a preprocessing phase. The current state of the art in this respect are the protocols in Bendlin et al., Damgård/Orlandi and Nielsen et al. [3,9,16]. The “MPC-in-the-head” technique of Ishai et al. [13,12] has similar overall asymptotic complexity, but larger constants and a less efficient online phase.

Recently, another approach has become possible with the advent of Fully Homomorphic Encryption (FHE) by Gentry [10]. In this approach all parties first encrypt their input under the FHE scheme; then they evaluate the desired function on the ciphertexts using the homomorphic properties, and finally they perform a distributed decryption on the final ciphertexts to get the results. The advantage of the FHE-based approach is that interaction is only needed to supply inputs and get output. However, the low bandwidth consumption comes at a price; current FHE schemes are very slow and can only evaluate small circuits, i.e., they actually only provide what is known as somewhat homomorphic encryption (SHE). This can be circumvented in two ways; either by assuming circular security and implementing an expensive bootstrapping operation, or by extending the parameter sizes to enable a “levelled FHE” scheme which can evaluate circuits of large degree (exponential in the number of levels) [4]. The main cost, much like other approaches, is in terms of the number of multiplications in the arithmetic circuit. So whilst theoretically appealing the approach via FHE is not competitive in practice with the traditional MPC approach.

1.1 Contributions of This Paper

Optimal Online Phase. We propose an MPC protocol in the preprocessing model that computes securely an arithmetic circuit C over any finite field \mathbb{F}_{p^k} . The protocol is statistically UC-secure against active and adaptive corruption of up to $n - 1$ of the n players, and we assume synchronous communication and secure point-to-point channels. Measured in elementary operations in \mathbb{F}_{p^k} the total amount of work done is $O(n \cdot |C| + n^3)$ where $|C|$ is the size of C . All earlier work in this model had complexity $\Omega(n^2 \cdot |C|)$. A similar improvement applies to the communication complexity and the amount of data one needs to store from the preprocessing. Hence, the work done by each player in the online phase is essentially independent of n . Moreover, it is only a small constant factor larger than what one would need to compute the circuit in the clear. This is the first protocol in the preprocessing model with these properties¹.

¹ With dishonest majority, successful termination cannot be guaranteed, so our protocols simply abort if cheating is detected. We do not, however, identify *who* cheated, indeed the standard definition of secure function evaluation does not require this. Identification of cheaters is possible but we do not know how to do this while maintaining complexity linear in n .

Finally, we show a lower bound implying that w.r.t the amount of data required from the preprocessing, our protocol is optimal up to a constant factor. We also obtain a similar lower bound on the number of bit operations required, and hence the computational work done in our protocol is optimal up to polylogarithmic factors.

All results mentioned here hold for the case of large fields, i.e., where the desired error probability is $(1/p^k)^c$, for a small constant c . Note that many applications of MPC need integer arithmetic, modular reductions, conversion to binary, etc., which we can emulate by computing in \mathbb{F}_p with p large enough to avoid overflow. This naturally leads to computing with large fields. As mentioned, our protocol works for all fields, but like earlier work in this model it is less efficient for small fields by a factor of essentially $\lceil \frac{\sec}{\log p^k} \rceil$ for error probability $2^{-\Theta(\sec)}$, see the full version for details.

Obtaining our result requires new ideas compared to [3], which was previously state of the art and was based on additive secret sharing where each share in a secret is authenticated using an information theoretic Message Authentication Code (MAC). Since each player needs to have his own key, each of the n shares need to be authenticated with n MACs, so this approach is inherently quadratic in n . Our idea is to authenticate the secret value itself instead of the shares, using a single global key. This seems to lead to a “chicken and egg” problem since one cannot check a MAC without knowing the key, but if the key is known, MACs can be forged. Our solution to this involves secret sharing the key as well, carefully timing when values are revealed, and various tricks to reduce the amortized cost of checking a set of MACs.

Efficient use of FHE for MPC. As a conceptual contribution we propose what we believe is “the right” way to use FHE/SHE for *computationally* efficient MPC, namely to use it for implementing a preprocessing phase. The observation is that since such preprocessing is typically based on the classic circuit randomization technique of Beaver [2], it can be done by evaluating in parallel many small circuits of small multiplicative depth (in fact depth 1 in our case). Thus SHE suffices, we do not need bootstrapping, and we can use the SHE SIMD approach of [17] to handle many values in parallel in a single ciphertext.

To capitalize on this idea, we apply the SIMD approach to the cryptosystem from [5] (see also [11] where this technique is also used). To get the best performance, we need to do a non-trivial analysis of the parameter values we can use, and we prove some results on norms of embeddings of a cyclotomic field for this purpose. We also design a distributed decryption procedure for our cryptosystem. This protocol is only robust against passive attacks. Nevertheless, this is sufficient for the overall protocol to be actively secure. Intuitively, this is because the only damage the adversary can do is to add a known error term to the decryption result obtained. The effect of this for the online protocol is that certain shares of secret values may be incorrect, but this will be caught by the check involving the MACs. Finally we adapt a zero-knowledge proof of plaintext

knowledge from [3] for our purpose and in particular we improve the analysis of the soundness guarantees it offers. This influences the choice of parameters for the cryptosystem and therefore improves overall performance.

An Efficient Preprocessing Protocol. As a result of the above, we obtain a constant-round preprocessing protocol that is UC-secure against active and static corruption of $n - 1$ players assuming the underlying cryptosystem is semantically secure, which follows from the polynomial (PLWE) assumption. UC-security for dishonest majority cannot be obtained without a set-up assumption. In this paper we assume that a key pair for our cryptosystem has been generated and the secret key has been shared among the players.

Whereas previous work in the preprocessing/online model [3,9] use $\Omega(n^2)$ public-key operations per secure multiplication, we only need $O(n^2/s)$ operations, where s is a number that grows with the security parameter of the SHE scheme (we have $s \approx 12000$ in our concrete instantiation for computing in \mathbb{F}_p where $p \approx 2^{64}$). We stress that our adapted scheme is exactly as efficient as the basic version of [5] that does not allow this optimization, so the improvement is indeed “genuine”.

In comparison to the approach mentioned above where one uses FHE throughout the protocol, our combined preprocessing and online phase achieves a result that is incomparable from a theoretical point of view, but much more practical: we need more communication and rounds, but the computational overhead is much smaller – we need $O(n^2/s \cdot |C|)$ public key operations compared to $O(n \cdot |C|)$ for the FHE approach, where for realistic values of n and s , we have $n^2/s \ll n$. Furthermore, we only need a low depth SHE which is much more efficient in the first place. And finally, we can push all the work using SHE into a, function independent, preprocessing phase.

Performance in Practice. Both the preprocessing and online phase have been implemented and tested for 3 players on up-to-date machines connected on a LAN. The preprocessing takes about 13 ms amortized time to prepare one multiplication in \mathbb{F}_p for a 64-bit p , with security level corresponding roughly to 1024 bit RSA and an error probability of 2^{-40} for the zero-knowledge proofs (the error probability can be lowered to 2^{-80} by repeating the ZK proofs which will at most double the time). This is 2-3 orders of magnitude faster than preliminary estimates for the most efficient instantiation of [3]. The online phase executes a secure 64-bit multiplication in 0.05 ms amortized time. These rough orders of magnitude, and the ability to deal with a non-trivial number of players, are born out by a recent implementation of the protocols described in this paper [7].

Concurrent Related Work. In recent independent work [15,1,11], Meyers et al., Asharov et al. and Gentry et al. also use an FHE scheme for multiparty computation. They follow the pure FHE approach mentioned above, using a threshold decryption protocol tailored to the specific FHE scheme. They focus primarily on round complexity, while we want to minimize the computational overhead. We note that in [11], Gentry et al. obtain small overhead by showing a way to

use the FHE SIMD approach for computing any circuit homomorphically. However, this requires full FHE with bootstrapping (to work on arbitrary circuits) and does not (currently) lead to a practical protocol.

In [16], Nielsen et al. consider secure computing for Boolean Circuits. Their online phase is similar to that of [3], while the preprocessing is a clever and very efficient construction based on Oblivious Transfer. This result is complementary to ours in the sense that we target computations over large fields which is good for some applications whereas for other cases, Boolean Circuits are the most compact way to express the desired computation. Of course, one could use the preprocessing from [16] to set up data for our online phase, but current benchmarks indicate that our approach is faster for large fields, say of size 64 bits or more.

We end the introduction by covering some basic notation which will be used throughout this paper. For a vector $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ we denote by $\|\mathbf{x}\|_\infty := \max_{1 \leq i \leq n} |x_i|$, $\|\mathbf{x}\|_1 := \sum_{1 \leq i \leq n} |x_i|$ and $\|\mathbf{x}\|_2 := \sqrt{\sum |x_i|^2}$. We let $\epsilon(\kappa)$ denote an unspecified negligible function of κ . If S is a set we let $x \leftarrow S$ denote assignment to the variable x with respect to a uniform distribution on S ; we use $x \leftarrow s$ for a value s as shorthand for $x \leftarrow \{s\}$. If A is an algorithm $x \leftarrow A$ means assign to x the output of A , where the probability distribution is over the random coins of A . Finally $x := y$ means “ x is defined to be y ”.

2 Online Protocol

Our aim is to construct a protocol for arithmetic multiparty computation over \mathbb{F}_{p^k} for some prime p . More precisely, we wish to implement the ideal functionality $\mathcal{F}_{\text{AMPC}}$, presented in . Our MPC protocol is structured in a preprocessing (or offline) phase and an online phase. We start out in this section by presenting the online phase which assumes access to an ideal functionality $\mathcal{F}_{\text{PREP}}$. In Section 5 we show how to implement this functionality in an independent preprocessing phase.

In our specification of the online protocol, we assume for simplicity that a broadcast channel is available at unit cost, that each party has only one input, and only one public output value is to be computed. In the full version we explain how to implement the broadcasts we need from point-to-point channels and lift the restriction on the number of inputs and outputs without this affecting the overall complexity.

Before presenting the concrete online protocol we give the intuition and motivation behind the construction. We will use unconditionally secure MACs to protect secret values from being manipulated by an active adversary. However, rather than authenticating shares of secret values as in [3], we authenticate the shared value itself. More concretely, we will use a global key α chosen randomly in \mathbb{F}_{p^k} , and for each secret value a , we will share a additively among the players, and we also secret-share a MAC αa . This way to represent secret values is linear, just like the representation in [3], and we can therefore do secure multiplication based on multiplication triples à la Beaver [2] that we produce in the preprocessing.

An immediate problem is that opening a value reliably seems to require that we check the MAC, and this requires players know α . However, as soon as α is known, MACs on other values can be forged. We solve this problem by postponing the check on the MACs (of opened values) to the output phase (of course, this may mean that some of the opened values are incorrect). During the output phase players generate a random linear combination of both the opened values and their shares of the corresponding MACs; they commit to the results and only then open α (see Figure 1). The intuition is that, because of the commitments, when α is revealed it is too late for corrupt players to exploit knowledge of the key. Therefore, if the MAC checks out, all opened values were correct with high probability, so we can trust that the output values we computed are correct and can safely open them.

Representation of Values and MACs. In the online phase each shared value $a \in \mathbb{F}_{p^k}$ is represented as follows

$$\langle a \rangle := (\delta, (a_1, \dots, a_n), (\gamma(a)_1, \dots, \gamma(a)_n))$$

where $a = a_1 + \dots + a_n$ and $\gamma(a)_1 + \dots + \gamma(a)_n = \alpha(a + \delta)$. Player P_i holds $a_i, \gamma(a)_i$ and δ is public. The interpretation is that $\gamma(a) \leftarrow \gamma(a)_1 + \dots + \gamma(a)_n$ is the MAC authenticating a under the global key α .

Computations. Using the natural component-wise addition of representations, and suppressing the underlying choices of $a_i, \gamma(a)_i$ for readability, we clearly have for secret values a, b and public constant e that

$$\langle a \rangle + \langle b \rangle = \langle a + b \rangle \quad e \cdot \langle a \rangle = \langle ea \rangle, \quad \text{and} \quad e + \langle a \rangle = \langle e + a \rangle,$$

where $e + \langle a \rangle := (\delta - e, (a_1 + e, a_2, \dots, a_n), (\gamma(a)_1, \dots, \gamma(a)_n))$. This possibility to easily add a public value is the reason for the “public modifier” δ in the definition of $\langle \cdot \rangle$. It is now clear that we can do secure linear computations directly on values represented this way.

What remains is multiplications: here we use the preprocessing. We would like the preprocessing to output random triples $\langle a \rangle, \langle b \rangle, \langle c \rangle$, where $c = ab$. However, our preprocessing produces triples which satisfy $c = ab + \Delta$, where Δ is an error that can be introduced by the adversary. We therefore need to check the triple before we use it. The check can be done by “sacrificing” another triple $\langle f \rangle, \langle g \rangle, \langle h \rangle$, where the same multiplicative equality should hold (see the protocol for details). Given such a valid triple, we can do multiplications in the following standard way: To compute $\langle xy \rangle$ we first open $\langle x \rangle - \langle a \rangle$ to get ϵ , and $\langle y \rangle - \langle b \rangle$ to get δ . Then $xy = (a + \epsilon)(b + \delta) = c + \epsilon b + \delta a + \epsilon \delta$. Thus, the new representation can be computed as

$$\langle x \rangle \cdot \langle y \rangle = \langle c \rangle + \epsilon \langle b \rangle + \delta \langle a \rangle + \epsilon \delta.$$

An important note is that during our protocol we are actually not guaranteed that we are working with the correct results, since we do not immediately check

Protocol Π_{ONLINE}

Initialize: The parties first invoke the preprocessing to get the shared secret key $[\alpha]$, a sufficient number of multiplication triples $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$, and pairs of random values $\langle r \rangle, [\![r]\!]$, as well as single random values $[\![t]\!], [\![e]\!]$. Then the steps below are performed in sequence according to the structure of the circuit to compute.

Input: To share P_i 's input x_i , P_i takes an available pair $\langle r \rangle, [\![r]\!]$. Then, do the following:

1. $[\![r]\!]$ is opened to P_i (if it is known in advance that P_i will provide input, this step can be done already in the preprocessing stage).
2. P_i broadcasts $\epsilon \leftarrow x_i - r$.
3. The parties compute $\langle x_i \rangle \leftarrow \langle r \rangle + \epsilon$.

Add: To add two representations $\langle x \rangle, \langle y \rangle$, the parties locally compute $\langle x \rangle + \langle y \rangle$.

Multiply: To multiply $\langle x \rangle, \langle y \rangle$ the parties do the following:

1. They take two triples $(\langle a \rangle, \langle b \rangle, \langle c \rangle), (\langle f \rangle, \langle g \rangle, \langle h \rangle)$ from the set of the available ones and check that indeed $a \cdot b = c$.
 - Open a representation of a random value $[\![t]\!]$.
 - partially open $t \cdot \langle a \rangle - \langle f \rangle$ to get ρ and $\langle b \rangle - \langle g \rangle$ to get σ
 - evaluate $t \cdot \langle c \rangle - \langle h \rangle - \sigma \cdot \langle f \rangle - \rho \cdot \langle g \rangle - \sigma \cdot \rho$, and partially open the result.
 - If the result is not zero the players abort, otherwise go on with $\langle a \rangle, \langle b \rangle, \langle c \rangle$.

Note that this check could in fact be done as part of the preprocessing. Moreover, it can be done for all triples in parallel, and so we actually need only one random value t .

2. The parties partially open $\langle x \rangle - \langle a \rangle$ to get ϵ and $\langle y \rangle - \langle b \rangle$ to get δ and compute $\langle z \rangle \leftarrow \langle c \rangle + \epsilon \langle b \rangle + \delta \langle a \rangle + \epsilon \delta$

Output: We enter this stage when the players have $\langle y \rangle$ for the output value y , but this value has been not been opened (the output value is only correct if players have behaved honestly). We then do the following:

1. Let a_1, \dots, a_T be all values publicly opened so far, where $\langle a_j \rangle = (\delta_j, (a_{j,1}, \dots, a_{j,n}), (\gamma(a_j)_1, \dots, \gamma(a_j)_n))$. Now, a random value $[\![e]\!]$ is opened, and players set $e_i = e^i$ for $i = 1, \dots, T$. All players compute $a \leftarrow \sum_j e_j a_j$.
2. Each P_i calls \mathcal{F}_{COM} to commit to $\gamma_i \leftarrow \sum_j e_j \gamma(a_j)_i$. For the output value $\langle y \rangle$, P_i also commits to his share y_i , and his share $\gamma(y)_i$ in the corresponding MAC.
3. $[\![\alpha]\!]$ is opened.
4. Each P_i asks \mathcal{F}_{COM} to open γ_i , and all players check that $\alpha(a + \sum_j e_j \delta_j) = \sum_i \gamma_i$. If this is not OK, the protocol aborts. Otherwise the players conclude that the output value is correctly computed.
5. To get the output value y , the commitments to $y_i, \gamma(y)_i$ are opened. Now, y is defined as $y := \sum_i y_i$ and each player checks that $\alpha(y + \delta) = \sum_i \gamma(y)_i$, if so, y is the output.

Fig. 1. The online phase

the MACs of the opened values. During the first part of the protocol, parties will only do what we define as a *partial opening*, meaning that for a value $\langle a \rangle$, each party P_i sends a_i to P_1 , who computes $a = a_1 + \dots + a_n$ and broadcasts a to all players. We assume here for simplicity that we always go via P_1 , whereas in practice, one would balance the workload over the players.

As sketched earlier we postpone the checking to the end of the protocol in the output phase. To check the MACs we need the global key α . We get α from the preprocessing but in a slightly different representation:

$$\llbracket \alpha \rrbracket := ((\alpha_1, \dots, \alpha_n), (\beta_i, \gamma(\alpha)_1^i, \dots, \gamma(\alpha)_n^i)_{i=1,\dots,n}),$$

where $\alpha = \sum_i \alpha_i$ and $\sum_j \gamma(\alpha)_j^i = \alpha \beta_i$. Player P_i holds $\alpha_i, \beta_i, \gamma(\alpha)_1^i, \dots, \gamma(\alpha)_n^i$. The idea is that $\gamma(\alpha)_i \leftarrow \sum_j \gamma(\alpha)_j^i$ is the MAC authenticating α under P_i 's private key β_i . To open $\llbracket \alpha \rrbracket$ each P_j sends to each P_i his share α_j of α and his share $\gamma(\alpha)_i^j$ of the MAC on α made with P_i 's private key and then P_i checks that $\sum_j \gamma(\alpha)_j^i = \alpha \beta_i$. (To open the value to only one party P_i , the other parties will simply send their shares only to P_i , who will do the checking. Only shares of α and $\alpha \beta_i$ are needed.)

Finally, the preprocessing will also output n pairs of a random value r in both of the presented representations $\langle r \rangle, \llbracket r \rrbracket$. These pairs are used in the Input phase of the protocol.

The full protocol for the online phase is shown in Figure 1. It assumes access to a commitment functionality \mathcal{F}_{COM} that simply receives values to commit to from players, stores them and reveals a value to all players on request from the committer. Such a functionality could be implemented efficiently based, e.g., on Paillier encryption or the DDH assumption [8,14]. However, we show in the full version that we can do ideal commitments based only on $\mathcal{F}_{\text{PREP}}$ and with cost $O(n^2)$ computation and communication.

Complexity. The (amortized) cost of a secure multiplication is easily seen to be $O(n)$ local elementary operations in \mathbb{F}_{p^k} , and communication of $O(n)$ field elements. Linear operations have the same computational cost but require no communication. The input stage requires $O(n)$ communication and computation to open $\llbracket r \rrbracket$ to P_i and one broadcast. Doing the output stage requires opening $O(n)$ commitments. In fact, the total number of commitments used is also $O(n)$, so this adds an $O(n^3)$ term to the complexity. In total, we therefore get the complexity claimed in the introduction: $O(n \cdot |C| + n^3)$ elementary field operations and storage/communication complexity $O(n \cdot |C| + n^3)$ field elements.

We can now state the theorem on security of the online phase, and its proof can be found in the full version.

Theorem 1. *In the $\mathcal{F}_{\text{PREP}}, \mathcal{F}_{\text{COM}}$ -hybrid model, the protocol Π_{ONLINE} implements $\mathcal{F}_{\text{AMPC}}$ with statistical security against any static² active adversary corrupting up to $n - 1$ parties.*

² The protocol is in fact adaptively secure, here we only show static security since our preprocessing is anyway only statically secure.

Based on a result from [18], we can also show a lower bound on the amount of preprocessing data and work required for a protocol. The proof is in the full version.

Theorem 2. *Assume a protocol π is the preprocessing model can compute any circuit over \mathbb{F}_{p^k} of size at most S , with security against active corruption of at most $n - 1$ players. We assume that the players supply roughly the same number of inputs ($O(S/n)$ each), and that any player may receive output. Then the preprocessing must output $\Omega(S \log p^k)$ bits to each player, and for any player P_i , there exists a circuit C satisfying the conditions above, where secure computation of C requires P_i to execute an expected number of bit operations that is $\Omega(S \log p^k)$.*

It is easy to see that our protocol satisfies the conditions in the theorem and that it meets the first bound up to a constant factor and the second up to a poly-logarithmic factor (as a function of the security parameter).

3 The Abstract Somewhat Homomorphic Encryption Scheme

In this section we specify the abstract properties we need for our cryptosystem. A concrete instantiation is found in Section 6.

We first define the plaintext space M . This will be given by a direct product of finite fields $(\mathbb{F}_{p^k})^s$ of characteristic p . Componentwise addition and multiplication of elements in M will be denoted by $+$ and \cdot . We assume there is an injective encoding function `encode` which takes elements in $(\mathbb{F}_{p^k})^s$ to elements in a ring R which is equal \mathbb{Z}^N (as a \mathbb{Z} -module) for some integer N . We also assume a `decode` function which takes *arbitrary* elements in \mathbb{Z}^N and returns an element in $(\mathbb{F}_{p^k})^s$. We require that for all $\mathbf{m} \in M$ that $\text{decode}(\text{encode}(\mathbf{m})) = \mathbf{m}$ and that the decode operation is compatible with the characteristic of the field, i.e. for any $\mathbf{x} \in \mathbb{Z}^N$ we have $\text{decode}(\mathbf{x}) = \text{decode}(\mathbf{x} \pmod p)$. And finally that the encoding function produces “short” vectors. More precisely, that for all $\mathbf{m} \in (\mathbb{F}_{p^k})^s$ $\|\text{encode}(\mathbf{m})\|_\infty \leq \tau$ where $\tau = p/2$.

The two operations in R will be denoted by $+$ and \cdot . The addition operation in R is assumed to be componentwise addition, whereas we make no assumption on multiplication. All we require is that the following properties hold, for all elements $\mathbf{m}_1, \mathbf{m}_2 \in M$;

$$\begin{aligned}\text{decode}(\text{encode}(\mathbf{m}_1) + \text{encode}(\mathbf{m}_2)) &= \mathbf{m}_1 + \mathbf{m}_2, \\ \text{decode}(\text{encode}(\mathbf{m}_1) \cdot \text{encode}(\mathbf{m}_2)) &= \mathbf{m}_1 \cdot \mathbf{m}_2.\end{aligned}$$

From now on, when we discuss the plaintext space M we assume it comes implicitly with the `encode` and `decode` functions for some integer N . If an element in M has the same component in each of the s -slots, then we call it a “diagonal” element. We let $\text{Diag}(x)$ for $x \in \mathbb{F}_{p^k}$ denote the element $(x, x, \dots, x) \in (\mathbb{F}_{p^k})^s$.

Our cryptosystem consists of a tuple $(\text{ParamGen}, \text{KeyGen}, \text{KeyGen}^*, \text{Enc}, \text{Dec})$ of algorithms defined below, and parametrized by a security parameter κ .

ParamGen($1^\kappa, M$): This parameter generation algorithm outputs an integer N (as above), definitions of the `encode` and `decode` functions, and a description of a randomized algorithm D_ρ^d , which outputs vectors in \mathbb{Z}^d . We assume that D_ρ^d outputs \mathbf{r} with $\|\mathbf{r}\|_\infty \leq \rho$, except with negligible probability. The algorithm D_ρ^d is used by the encryption algorithm to select the random coins needed during encryption. The algorithm **ParamGen** also outputs an additive abelian group G . The group G also possesses a (not necessarily closed) multiplicative operator, which is commutative and distributes over the additive group of G . The group G is the group in which the ciphertexts will be assumed to lie. We write \boxplus and \boxtimes for the operations on G , and extend these in the natural way to vectors and matrices of elements of G . Finally **ParamGen** outputs a set C of allowable arithmetic SIMD circuits over $(\mathbb{F}_{p^k})^s$, these are the set of functions which our scheme will be able to evaluate ciphertexts over. We can think of C as a subset of $\mathbb{F}_{p^k}[X_1, X_2, \dots, X_n]$, where we evaluate a function $f \in \mathbb{F}_{p^k}[X_1, X_2, \dots, X_n]$ a total of s times in parallel on inputs from $(\mathbb{F}_{p^k})^n$. We assume that all other algorithms take as implicit input the output $P \leftarrow (1^\kappa, N, \text{encode}, \text{decode}, D_\rho^d, G, C)$ of **ParamGen**.

KeyGen(): This algorithm outputs a public key \mathbf{pk} and a secret key \mathbf{sk} .

Enc_{pk}(x, r): On input of $\mathbf{x} \in \mathbb{Z}^N$, $\mathbf{r} \in \mathbb{Z}^d$, this deterministic algorithm outputs a ciphertext $c \in G$. When applying this algorithm one would obtain \mathbf{x} from the application of the `encode` function, and \mathbf{r} by calling D_ρ^d . This is what we mean when we write **Enc_{pk}(m)**, where $\mathbf{m} \in M$. However, it is convenient for us to define **Enc** on the intermediate state, $\mathbf{x} = \text{encode}(\mathbf{m})$. To ease notation we write **Enc_{pk}(x)** if the value of the randomness \mathbf{r} is not important for our discussion. To make our zero-knowledge proofs below work, we will require that addition of V “clean” ciphertexts (for “small” values of V), of plaintext \mathbf{x}_i in \mathbb{Z}^N , using randomness \mathbf{r}_i , results in a ciphertext which could be obtained by adding the plaintexts and randomness, as integer vectors, and then applying **Enc_{pk}(x, r)**, i.e.

$$\text{Enc}_{\mathbf{pk}}(\mathbf{x}_1 + \cdots + \mathbf{x}_V, \mathbf{r}_1 + \cdots + \mathbf{r}_V) = \text{Enc}_{\mathbf{pk}}(\mathbf{x}_1, \mathbf{r}_1) \boxplus \cdots \boxplus \text{Enc}_{\mathbf{pk}}(\mathbf{x}_V, \mathbf{r}_V).$$

Dec_{sk}(c): On input the secret key and a ciphertext c it returns either an element $\mathbf{m} \in M$, or the symbol \perp .

We are now able to define various properties of the above abstract scheme that we will require. But first a bit of notation: For a function $f \in C$ we let $n(f)$ denote the number of variables in f , and we let \hat{f} denote the function on G induced by f . That is, given f , we replace every $+$ operation with a \boxplus , every \cdot operation is replaced with a \boxtimes and every constant c is replaced by $\text{Enc}_{\mathbf{pk}}(\text{encode}(c), \mathbf{0})$. Also, given a set of $n(f)$ vectors $\mathbf{x}_1, \dots, \mathbf{x}_{n(f)}$, we define $f(\mathbf{x}_1, \dots, \mathbf{x}_{n(f)})$ in the natural way by applying f in parallel on each coordinate.

Correctness: Intuitively correctness means that if one decrypts the result of a function $f \in C$ applied to $n(f)$ encrypted vectors of variables, then this should return the same value as applying the function to the $n(f)$ plaintexts. However, to apply the scheme in our protocol, we need to be a bit more liberal, namely the decryption result should be correct, even if the ciphertexts we start from were not necessarily generated by the normal encryption algorithm. They

only need to “contain” encodings and randomness that are not too large, such that the encodings decode to legal values. Formally, the scheme is said to be (B_{plain}, B_{rand}, C) -correct if

$$\Pr [P \leftarrow \text{ParamGen}(1^\kappa, M), (\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KeyGen}(), \text{ for any } f \in C, \\ \text{any } \mathbf{x}_i, \mathbf{r}_i, \text{ with } \|\mathbf{x}_i\|_\infty \leq B_{plain}, \|\mathbf{r}_i\|_\infty \leq B_{rand}, \text{ decode}(\mathbf{x}_i) \in (\mathbb{F}_{p^k})^s, \\ i = 1, \dots, n(f), \text{ and } c_i \leftarrow \text{Enc}_{\mathbf{pk}}(\mathbf{x}_i, \mathbf{r}_i), c \leftarrow \hat{f}(c_1, \dots, c_{n(f)}): \\ \text{Dec}_{\mathbf{sk}}(c) \neq f(\text{decode}(\mathbf{x}_1), \dots, \text{decode}(\mathbf{x}_{n(f)}))] < \epsilon(\kappa).$$

We will say that a ciphertext is (B_{plain}, B_{rand}, C) -admissible if it can be obtained as the ciphertext c in the above experiment, i.e., by applying a function from C to ciphertexts generated from (legal) encodings and randomness that are bounded by B_{plain} and B_{rand} .

KeyGen^{*}($\widetilde{\mathbf{pk}}$): This is a randomized algorithm that outputs a *meaningless public key* $\widetilde{\mathbf{pk}}$. We require that an encryption of any message $\text{Enc}_{\widetilde{\mathbf{pk}}}(\mathbf{x})$ is statistically indistinguishable from an encryption of 0. Furthermore, if we set $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KeyGen}()$ and $\widetilde{\mathbf{pk}} \leftarrow \text{KeyGen}^*(\mathbf{pk})$, then \mathbf{pk} and $\widetilde{\mathbf{pk}}$ are computationally indistinguishable. This implies the scheme is IND-CPA secure in the usual sense.

Distributed Decryption: We assume, as a set up assumption, that a common public key has been set up where the secret key has been secret-shared among the players in such a way that they can collaborate to decrypt a ciphertext. We assume throughout that only (B_{plain}, B_{rand}, C) -admissible ciphertexts are to be decrypted, this constraint is guaranteed by our main protocol.

We note that some set-up assumption is always required to show UC security which is our goal here. Concretely, we assume that a functionality $\mathcal{F}_{\text{KEYGEN}}$ is available, as specified in Figure 2. It basically generates a key pair and secret-shares the secret key among the players using a secret-sharing scheme that is assumed to be given as part of the specification of the cryptosystem. Since we want to allow corruption of all but one player, the maximal unqualified sets must be all sets of $n - 1$ players.

Functionality $\mathcal{F}_{\text{KEYGEN}}$

1. When receiving “start” from all honest players, run $P \leftarrow \text{ParamGen}(1^\kappa, M)$, and then, using the parameters generated, run $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KeyGen}()$ (recall P , and hence 1^κ , is an implicit input to all functions we specify). Send \mathbf{pk} to the adversary.
2. We assume a secret sharing scheme is given with which \mathbf{sk} can be secret-shared. Receive from the adversary a set of shares s_j for each corrupted player P_j .
3. Construct a complete set of shares (s_1, \dots, s_n) consistent with the adversary’s choices and \mathbf{sk} . Note that this is always possible since the corrupted players form an unqualified set. Send \mathbf{pk} to all players and s_i to each honest P_i .

Fig. 2. The Ideal Functionality for Distributed Key Generation

We note that it is possible to make a weaker set-up assumption, such as a common reference string (CRS), and using a general UC secure multiparty computation protocol for the CRS model to implement $\mathcal{F}_{\text{KEYGEN}}$. While this may not be very efficient, one only needs to run this protocol once in the life-time of the system.

We also want our cryptosystem to implement the functionality $\mathcal{F}_{\text{KEYGENDEC}}$ in Figure 3, which essentially specifies that players can cooperate to decrypt a $(B_{\text{plain}}, B_{\text{rand}}, C)$ -admissible ciphertext, but the protocol is only secure against a passive attack: the adversary gets the correct decryption result, but can decide which result the honest players should learn.

Functionality $\mathcal{F}_{\text{KEYGENDEC}}$

1. When receiving “start” from all honest players, run $\text{ParamGen}(1^\kappa, M)$, and then, using the parameters generated, run $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}()$. Send pk to the adversary and to all players, and store sk .
2. Hereafter on receiving “decrypt c ” for $(B_{\text{plain}}, B_{\text{rand}}, C)$ -admissible c from all honest players, send c and $m \leftarrow \text{Dec}_{\text{sk}}(c)$ to the adversary. On receiving m' from the adversary, send “Result m' ” to all players. Both m and m' may be a special symbol \perp indicating that decryption failed.
3. On receiving “decrypt c to P_j ” for admissible c , if P_j is corrupt, send $c, m \leftarrow \text{Dec}_{\text{sk}}(c)$ to the adversary. If P_j is honest, send c to the adversary. On receiving δ from the adversary, if $\delta \notin M$, send \perp to P_j , if $\delta \in M$, send $\text{Dec}_{\text{sk}}(c) + \delta$ to P_j .

Fig. 3. The Ideal Functionality for Distributed Key Generation and Decryption

We are now finally ready to define the basic set of properties that the underlying cryptosystem should satisfy, in order to be used in our protocol. Here we use an “information theoretic” security parameter sec that controls the errors in our ZK proofs below.

Definition 1. (Admissible Cryptosystem.) Let C contain formulas of form $(x_1 + \dots + x_n) \cdot (y_1 + \dots + y_n) + z_1 + \dots + z_n$, as well as all “smaller” formulas, i.e., with a smaller number of additions and possibly no multiplication. A cryptosystem is admissible if it is defined by algorithms $(\text{ParamGen}, \text{KeyGen}, \text{KeyGen}^*, \text{Enc}, \text{Dec})$ with properties as defined above, is $(B_{\text{plain}}, B_{\text{rand}}, C)$ -correct, where

$$B_{\text{plain}} = N \cdot \tau \cdot \text{sec}^2 \cdot 2^{(1/2+\nu)\text{sec}}, \quad B_{\text{rand}} = d \cdot \rho \cdot \text{sec}^2 \cdot 2^{(1/2+\nu)\text{sec}},$$

and where $\nu > 0$ can be an arbitrary constant. Finally there exist a secret sharing scheme as required in $\mathcal{F}_{\text{KEYGEN}}$ and a protocol $\Pi_{\text{KeyGenDec}}$ with the property that when composed with $\mathcal{F}_{\text{KEYGEN}}$ it securely implements the functionality $\mathcal{F}_{\text{KEYGENDEC}}$.

The set C is defined to contain all computations on ciphertext that we need in our main protocol. Throughout the paper we will assume that $B_{\text{plain}}, B_{\text{rand}}$ are

defined as here in terms of τ, ρ and sec . This is because these are the bounds we can force corrupt players to respect via our zero-knowledge protocol, as we shall see.

4 Zero-Knowledge Proof of Plaintext Knowledge

This section presents a zero-knowledge protocol that takes as input sec ciphertexts $c_1, \dots, c_{\text{sec}}$ generated by one of the players in our protocol, who will act as the prover. If the prover is honest then $c_i = \text{Enc}_{\text{pk}}(\mathbf{x}_i, \mathbf{r}_i)$, where \mathbf{x}_i has been obtained from the `encode` function, i.e. $\|\mathbf{x}_i\|_\infty \leq \tau$, and \mathbf{r}_i has been generated from D_ρ^d (so we may assume that $\|\mathbf{r}_i\|_\infty \leq \rho$). Our protocol is a zero-knowledge proof of plaintext knowledge (ZKPoPK) for the following relation:

$$\begin{aligned} R_{\text{PoPK}} = \{ & (x, w) \mid x = (\text{pk}, \mathbf{c}), w = ((\mathbf{x}_1, \mathbf{r}_1), \dots, (\mathbf{x}_{\text{sec}}, \mathbf{r}_{\text{sec}})) : \\ & \mathbf{c} = (c_1, \dots, c_{\text{sec}}), c_i \leftarrow \text{Enc}_{\text{pk}}(\mathbf{x}_i, \mathbf{r}_i), \\ & \|\mathbf{x}_i\|_\infty \leq B_{\text{plain}}, \text{decode}(\mathbf{x}_i) \in (\mathbb{F}_{p^k})^s, \|\mathbf{r}_i\|_\infty \leq B_{\text{rand}} \} . \end{aligned}$$

The zero-knowledge and completeness properties hold only if the ciphertexts c_i satisfy $\|\mathbf{x}_i\|_\infty \leq \tau$ and $\|\mathbf{r}_i\|_\infty \leq \rho$.

In our preprocessing protocol, players will be required to give such a ZKPoPK for all ciphertexts they provide. By admissibility of the cryptosystem, this will imply that every ciphertext occurring in the protocol will be $(B_{\text{plain}}, B_{\text{rand}}, C)$ -admissible and can therefore be decrypted correctly. The ZKPoPK can also be called with a flag `diag` which will modify the proof so that it additionally proves that `decode`(\mathbf{x}_i) is a diagonal element.

The protocol is not meant to implement an ideal functionality, but we can still use it and prove UC security for the main protocol, since we will always generate the challenge \mathbf{e} by calling the $\mathcal{F}_{\text{RAND}}$ ideal functionality (see the full version for more details).

The protocol and its proof of security are given in the full version and its computational complexity per ciphertext is essentially the cost of a constant number of encryptions. In the full version, we also give a variant of the ZK proof that allows even smaller values for $B_{\text{plain}}, B_{\text{rand}}$, namely $B_{\text{plain}} = N \cdot \tau \cdot \text{sec}^2 \cdot 2^{\text{sec}/2+8}$, $B_{\text{rand}} = d \cdot \rho \cdot \text{sec}^2 \cdot 2^{\text{sec}/2+8}$, and hence improves performance further. This variant is most efficient when executed using the Fiat-Shamir heuristic (although it can also work without random oracles), and we believe this variant is the best for a practical implementation.

5 The Preprocessing Phase

In this section we construct the protocol Π_{PREP} which securely implements the functionality $\mathcal{F}_{\text{PREP}}$ in the presence of functionalities $\mathcal{F}_{\text{KEYGENDEC}}$ (Figure 3) and $\mathcal{F}_{\text{RAND}}$. The preprocessing uses the above abstract cryptosystem with $M = (\mathbb{F}_{p^k})^s$, but the online phase is designed for messages in \mathbb{F}_{p^k} . Therefore, we extend the notation $\langle \cdot \rangle$ and $[\cdot]$ to messages in M : since addition and multiplication on

M are componentwise, for $\mathbf{m} = (m_1, \dots, m_s)$, we define $\langle \mathbf{m} \rangle = (\langle m_1 \rangle, \dots, \langle m_s \rangle)$ and similarly for $\llbracket \mathbf{m} \rrbracket$. Conversely, once a representation (or a pair, triple) on vectors is produced in the preprocessing, it will be disassembled into its coordinates, so that it can be used in the online phase. In Figures 4,5 and 6, we introduce subprotocols that are accessed by the main preprocessing protocol in several steps. Note that the subprotocols are not meant to implement ideal functionalities: their purpose is merely to summarize parts of the main protocol that are repeated in various occasions. Theorem 3 below is proved in the full version.

Theorem 3. *The protocol Π_{PREP} (Figure 7) implements $\mathcal{F}_{\text{PREP}}$ with computational security against any static, active adversary corrupting up to $n - 1$ parties, in the $\mathcal{F}_{\text{KEYGEN}}, \mathcal{F}_{\text{RAND}}$ -hybrid model when the underlying cryptosystem is admissible³.*

Protocol Reshare

Usage: Input is $e_{\mathbf{m}}$, where $e_{\mathbf{m}} = \text{Enc}_{\text{pk}}(\mathbf{m})$ is a public ciphertext and a parameter enc , where $enc = \text{NewCiphertext}$ or $enc = \text{NoNewCiphertext}$. Output is a share \mathbf{m}_i of \mathbf{m} to each player P_i ; and if $enc = \text{NewCiphertext}$, a ciphertext $e'_{\mathbf{m}}$. The idea is that $e_{\mathbf{m}}$ could be a product of two ciphertexts, which Reshare converts to a “fresh” ciphertext $e'_{\mathbf{m}}$. Since Reshare uses distributed decryption (that may return an incorrect result), it is not guaranteed that $e_{\mathbf{m}}$ and $e'_{\mathbf{m}}$ contain the same value, but it is guaranteed that $\sum_i \mathbf{m}_i$ is the value contained in $e'_{\mathbf{m}}$.

Reshare($e_{\mathbf{m}}$, enc) :

1. Each P_i samples a uniform $\mathbf{f}_i \in (\mathbb{F}_{p^k})^s$. Define $\mathbf{f} := \sum_{i=1}^n \mathbf{f}_i$.
2. Each P_i computes and broadcasts $e_{\mathbf{f}_i} \leftarrow \text{Enc}_{\text{pk}}(\mathbf{f}_i)$.
3. Each P_i runs Π_{ZKPoPK} as a prover on $e_{\mathbf{f}_i}$. The protocol aborts if any proof fails.
4. The players compute $e_{\mathbf{f}} \leftarrow e_{\mathbf{f}_1} \boxplus \dots \boxplus e_{\mathbf{f}_n}$, and $e_{\mathbf{m} + \mathbf{f}} \leftarrow e_{\mathbf{m}} \boxplus e_{\mathbf{f}}$.
5. The players invoke $\mathcal{F}_{\text{KEYGENDEC}}$ to decrypt $e_{\mathbf{m} + \mathbf{f}}$ and thereby obtain $\mathbf{m} + \mathbf{f}$.
6. P_1 sets $\mathbf{m}_1 \leftarrow \mathbf{m} + \mathbf{f} - \mathbf{f}_1$, and each player P_i ($i \neq 1$) sets $\mathbf{m}_i \leftarrow -\mathbf{f}_i$.
7. If $enc = \text{NewCiphertext}$, all players set $e'_{\mathbf{m}} \leftarrow \text{Enc}_{\text{pk}}(\mathbf{m} + \mathbf{f}) \boxminus e_{\mathbf{f}_1} \boxminus \dots \boxminus e_{\mathbf{f}_n}$, where a default value for the randomness is used when computing $\text{Enc}_{\text{pk}}(\mathbf{m} + \mathbf{f})$.

Fig. 4. The sub-protocol for additively secret sharing a plaintext $\mathbf{m} \in (\mathbb{F}_{p^k})^s$ on input a ciphertext $e_{\mathbf{m}} = \text{Enc}_{\text{pk}}(\mathbf{m})$

6 Concrete Instantiation of the Abstract Scheme Based on LWE

We now describe the concrete scheme, which is based on the somewhat homomorphic encryption scheme of Brakerski and Vaikuntanathan (BV) [5]. The main differences are that we are only interested in evaluation of circuits of multiplicative depth one, we are interested in performing operations in parallel on multiple

³ The definition of admissible cryptosystem demands a decryption protocol that implements $\mathcal{F}_{\text{KEYGENDEC}}$ based on $\mathcal{F}_{\text{KEYGEN}}$, hence the theorem only assumes $\mathcal{F}_{\text{KEYGEN}}$.

Protocol PBracket

Usage: On input shares $\mathbf{v}_1, \dots, \mathbf{v}_n$ privately held by the players and public ciphertext $e_{\mathbf{v}}$, this protocol generates $\llbracket \mathbf{v} \rrbracket$. It is assumed that $\sum_i \mathbf{v}_i$ is the plaintext contained in $e_{\mathbf{v}}$.

PBracket($\mathbf{v}_1, \dots, \mathbf{v}_n, e_{\mathbf{v}}$) :

1. For $i = 1, \dots, n$
 - (a) All players set $e_{\gamma_i} \leftarrow e_{\beta_i} \boxtimes e_{\mathbf{v}}$ (note that e_{β_i} is generated during the initialization process, and known by every player)
 - (b) Players generate $(\gamma_i^1, \dots, \gamma_i^n) \leftarrow \text{Reshare}(e_{\gamma_i}, \text{NoNewCiphertext})$, so each player P_j gets a share γ_i^j of $\mathbf{v} \cdot \beta_i$.
2. Output the representation $\llbracket \mathbf{v} \rrbracket = (\mathbf{v}_1, \dots, \mathbf{v}_n, (\beta_i, \gamma_i^1, \dots, \gamma_i^n)_{i=1, \dots, n})$.

Fig. 5. The sub-protocol for generating $\llbracket \mathbf{v} \rrbracket$

Protocol PAngle

Usage: On input shares $\mathbf{v}_1, \dots, \mathbf{v}_n$ privately held by the players and public ciphertext $e_{\mathbf{v}}$, this protocol generates $\langle \mathbf{v} \rangle$. It is assumed that $\sum_i \mathbf{v}_i$ is the plaintext contained in $e_{\mathbf{v}}$.

PAngle($\mathbf{v}_1, \dots, \mathbf{v}_n, e_{\mathbf{v}}$) :

1. All players set $e_{\mathbf{v} \cdot \alpha} \leftarrow e_{\mathbf{v}} \boxtimes e_{\alpha}$ (note that e_{α} is generated during the initialization process, and known by every player)
2. Players generate $(\gamma_1, \dots, \gamma_n) \leftarrow \text{Reshare}(e_{\mathbf{v} \cdot \alpha}, \text{NoNewCiphertext})$, so each player P_i gets a share γ_i of $\alpha \cdot \mathbf{v}$.
3. Output representation $\langle \mathbf{v} \rangle = (0, \mathbf{v}_1, \dots, \mathbf{v}_n, \gamma_1, \dots, \gamma_n)$.

Fig. 6. The sub-protocol for generating $\langle \mathbf{v} \rangle$

data items, and we require a distributed decryption procedure. In this section we detail the scheme and the distributed decryption procedure; in the full version we discuss security of the scheme, and present some sample parameter sizes and performance figures.

ParamGen($1^\kappa, M$): Recall the message space is given by $M = (\mathbb{F}_{p^k})^s$ for two integers k and s , and a prime p , i.e. the message space is s copies of the finite field \mathbb{F}_{p^k} . To map this to our scheme below, one first finds a cyclotomic polynomial $F(X) := \Phi_m(X)$ of degree $N := \phi(m)$, where N is lower bounded by some function of the security parameter κ . The polynomial $F(X)$ needs to be such that modulo p the polynomial $F(X)$ factors into l' irreducible factors of degree k' where $l' \geq s$ and k divides k' . We then define an algebra A_p as $A_p := \mathbb{F}_p[X]/F(X)$ and we have an embedding of M into A_p , $\phi : M \rightarrow A_p$. By “lifting” modulo p we see that there is a natural inclusion $\iota : A_p \rightarrow \mathbb{Z}^N$, which maps the polynomial of degree less than N with coefficients in \mathbb{F}_p into the integer vector of length N with coefficients in the range $(-p/2, \dots, p/2]$. The encode function is then defined by $\iota(\phi(\mathbf{m}))$ for $\mathbf{m} \in (\mathbb{F}_{p^k})^s$, with decode defined by $\phi^{-1}(\mathbf{x} \pmod p)$ for $\mathbf{x} \in \mathbb{Z}^N$. It is clear, by choice of the natural inclusion ι , that $\|\text{encode}(\mathbf{m})\|_\infty \leq p/2 = \tau$.

Protocol Π_{PREP}

Usage: The Triple-step is always executed \sec times in parallel. This ensures that when calling Π_{ZKPoPK} , we can always give it the \sec ciphertexts it requires as input. In addition both Π_{ZKPoPK} and Π_{PREP} can be executed in a SIMD fashion, i.e. they are data-oblivious bar when they detect an error. Thus we can execute Π_{ZKPoPK} and Π_{PREP} on the packed plaintext space $(\mathbb{F}_{p^k})^s$. Thereby, we generate $s \cdot \sec$ elements in one go and then buffer the generated triples, outputting the next unused one on demand.

Initialize: This step generates the global key α and “personal keys” β_i .

1. The players call “start” on $\mathcal{F}_{\text{KEYGENDEC}}$ to obtain the public key \mathbf{pk}
2. Each P_i generates a MAC-key $\beta_i \in \mathbb{F}_{p^k}$
3. Each P_i generates $\alpha_i \in \mathbb{F}_{p^k}$. Let $\alpha := \sum_{i=1}^n \alpha_i$
4. Each P_i computes and broadcasts
 $e_{\alpha_i} \leftarrow \text{Enc}_{\mathbf{pk}}(\text{Diag}(\alpha_i)), e_{\beta_i} \leftarrow \text{Enc}_{\mathbf{pk}}(\text{Diag}(\beta_i))$
5. Each P_i invokes Π_{ZKPoPK} (with diag set to true) as prover on input $(e_{\alpha_1}, \dots, e_{\alpha_n})$ and on input $(e_{\beta_1}, \dots, e_{\beta_n})$, where $e_{\alpha_i}, e_{\beta_i}$ are repeated \sec times, which is the number of ciphertexts Π_{ZKPoPK} requires as input. (This is not very efficient, but only needs to be done once for each player.)
6. All players compute $e_\alpha \leftarrow e_{\alpha_1} \boxplus \dots \boxplus e_{\alpha_n}$, and generate $[\![\text{Diag}(\alpha)]\!] \leftarrow \text{PBracket}(\text{Diag}(\alpha_1), \dots, \text{Diag}(\alpha_n), e_\alpha)$

Pair: This step generates a pair $[\![\mathbf{r}]\!], \langle \mathbf{r} \rangle$, and can be used to generate a single value $[\![\mathbf{r}]\!]$, by not performing the call to Pangle

1. Each P_i generates $\mathbf{r}_i \in (\mathbb{F}_{p^k})^s$. Let $\mathbf{r} := \sum_{i=1}^n \mathbf{r}_i$
2. Each P_i computes and broadcasts $e_{\mathbf{r}_i} \leftarrow \text{Enc}_{\mathbf{pk}}(\mathbf{r}_i)$. Let $e_{\mathbf{r}} = e_{\mathbf{r}_1} \boxplus \dots \boxplus e_{\mathbf{r}_n}$
3. Each P_i invokes Π_{ZKPoPK} as prover on the ciphertext he generated
4. Players generate

$$[\![\mathbf{r}]\!] \leftarrow \text{PBracket}(\mathbf{r}_1, \dots, \mathbf{r}_n, e_{\mathbf{r}}), \langle \mathbf{r} \rangle \leftarrow \text{PAngle}(\mathbf{r}_1, \dots, \mathbf{r}_n, e_{\mathbf{r}})$$

Triple: This step generates a multiplicative triple $\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle$

1. Each P_i generates $\mathbf{a}_i, \mathbf{b}_i \in (\mathbb{F}_{p^k})^s$. Let $\mathbf{a} := \sum_{i=1}^n \mathbf{a}_i, \mathbf{b} := \sum_{i=1}^n \mathbf{b}_i$
2. Each P_i computes and broadcasts $e_{\mathbf{a}_i} \leftarrow \text{Enc}_{\mathbf{pk}}(\mathbf{a}_i), e_{\mathbf{b}_i} \leftarrow \text{Enc}_{\mathbf{pk}}(\mathbf{b}_i)$
3. Each P_i invokes Π_{ZKPoPK} as prover on the ciphertexts he generated.
4. The players set $e_{\mathbf{a}} \leftarrow e_{\mathbf{a}_1} \boxplus \dots \boxplus e_{\mathbf{a}_n}$ and $e_{\mathbf{b}} \leftarrow e_{\mathbf{b}_1} \boxplus \dots \boxplus e_{\mathbf{b}_n}$
5. Players generate
 $\langle \mathbf{a} \rangle \leftarrow \text{PAngle}(\mathbf{a}_1, \dots, \mathbf{a}_n, e_{\mathbf{a}}), \langle \mathbf{b} \rangle \leftarrow \text{PAngle}(\mathbf{b}_1, \dots, \mathbf{b}_n, e_{\mathbf{b}})$.
6. All players compute $e_{\mathbf{c}} \leftarrow e_{\mathbf{a}} \boxtimes e_{\mathbf{b}}$
7. Players set $(\mathbf{c}_1, \dots, \mathbf{c}_n, e'_{\mathbf{c}}) \leftarrow \text{Reshare}(e_{\mathbf{c}}, \text{NewCiphertext})$.
8. Players generate $\langle \mathbf{c} \rangle \leftarrow \text{PAngle}(\mathbf{c}_1, \dots, \mathbf{c}_n, e'_{\mathbf{c}})$.

Fig. 7. The protocol for constructing the global key $[\![\alpha]\!]$, pairs $[\![\mathbf{r}]\!], \langle \mathbf{r} \rangle$ and multiplicative triples $\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle$

We pick a large integer q , whose size we will determine later, and defined $A_q := (\mathbb{Z}/q\mathbb{Z})[X]/F(X)$, i.e. the ring of integer polynomials modulo reduction by $F(X)$ and q . In practice we consider the image of `encode` to lie in A_q , and thus we abuse notation, by writing addition and multiplication in A_q by $+$ and \cdot . Note, that this means that applying `decode` to elements obtained from `encode` followed by a series of arithmetic operations may not result in the value in M which one would expect. This corresponds to where our scheme can only evaluate circuits from a given set C .

The ciphertext space G is defined to be A_q^3 , with addition \boxplus defined componentwise. The multiplicative operator \boxtimes is defined as follows

$$(\mathbf{a}_0, \mathbf{a}_1, 0) \boxtimes (\mathbf{b}_0, \mathbf{b}_1, 0) := (\mathbf{a}_0 \cdot \mathbf{b}_0, \mathbf{a}_1 \cdot \mathbf{b}_0 + \mathbf{a}_0 \cdot \mathbf{b}_1, -\mathbf{a}_1 \cdot \mathbf{b}_1),$$

i.e. multiplication is only defined on elements whose third coefficient is zero.

We define D_ρ^d as follows: The discrete Gaussian $D_{\mathbb{Z}^N, s}$, with *Gaussian parameter* s , is defined to be the random variable on \mathbb{Z}_q^N (centered around the origin) obtained from sampling $\mathbf{x} \in \mathbb{R}^N$, with probability proportional to $\exp(-\pi \cdot \|\mathbf{x}\|_2/s^2)$, and then rounding the result to the nearest lattice point and reducing it modulo q . Note, sampling from the distribution with probability density function proportional to $\exp(-\pi \cdot \|\mathbf{x}\|_2/s^2)$, means using a normal variate with mean zero, and standard deviation $r := s/\sqrt{2 \cdot \pi}$. In our concrete scheme we set $d := 3 \cdot N$ and define D_ρ^d to be the distribution defined by $(D_{\mathbb{Z}^N, s})^3$. Note, that in the notation D_ρ^d the implicit dependence on q has been suppressed to ease readability. The determining of q and r as functions of all the other parameters, we leave until we discuss security of the scheme.

KeyGen(): We will use the public key version of the Brakerski–Vaikuntanathan scheme [5]. Given the above set up, key generation proceeds as follows: First one samples elements $\mathbf{a} \leftarrow A_q$ and $\mathbf{s}, \mathbf{e} \leftarrow D_{\mathbb{Z}^N, s}$. Then treating \mathbf{s} and \mathbf{e} as elements of A_q one computes $\mathbf{b} \leftarrow (\mathbf{a} \cdot \mathbf{s}) + (p \cdot \mathbf{e})$. The public and private key are then set to be $\mathbf{pk} \leftarrow (\mathbf{a}, \mathbf{b})$ and $\mathbf{sk} \leftarrow \mathbf{s}$.

Enc_{pk}(x, r): Given a message $\mathbf{x} \leftarrow \text{encode}(m)$ where $m \in M$, and $\mathbf{r} \in D_\rho^d$, we proceed as follows: The element \mathbf{r} is parsed as $(\mathbf{u}, \mathbf{v}, \mathbf{w}) \in (\mathbb{Z}^N)^3$. Then the encryptor computes $\mathbf{c}_0 \leftarrow (\mathbf{b} \cdot \mathbf{v}) + (p \cdot \mathbf{w}) + \mathbf{x}$ and $\mathbf{c}_1 \leftarrow (\mathbf{a} \cdot \mathbf{v}) + (p \cdot \mathbf{u})$. Finally returning the ciphertext $(\mathbf{c}_0, \mathbf{c}_1, 0)$.

Dec_{sk}(c): Given a secret key $\mathbf{sk} = \mathbf{s}$ and a ciphertext $c = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)$ this algorithm computes the element in A_q satisfying $\mathbf{t} = \mathbf{c}_0 - (\mathbf{s} \cdot \mathbf{c}_1) - (\mathbf{s} \cdot \mathbf{s} \cdot \mathbf{c}_2)$. On reduction by q the value of $\|\mathbf{t}\|_\infty$ will be bounded by a relatively small constant B ; assuming of course that the “noise” within a ciphertext has not grown too large. We shall refer to the value $\mathbf{t} \bmod q$ as the “noise”, despite it also containing the message to be decrypted. At this point the decryptor simply reduces \mathbf{t} modulo p to obtain the desired plaintext in A_q , which can then be decoded via the `decode` algorithm.

KeyGen^{*}(): This simply samples $\hat{\mathbf{a}}, \hat{\mathbf{b}} \leftarrow A_q$ and returns $\hat{\mathbf{pk}} := (\hat{\mathbf{a}}, \hat{\mathbf{b}})$.

Following the discussion in [5] we see that with this *fixed* ciphertext space, our scheme is somewhat homomorphic. It can support a relatively large number of addition operations, and a single multiplication.

Distributed Version. We now extend the scheme above to enable distributed decryption. We first set up the distributed keys as follows. After invoking the functionality for key generation, each player obtains a share $\mathbf{sk}_i = (\mathbf{s}_{i,1}, \mathbf{s}_{i,2})$, these are chosen uniformly such that the master secret is written as

$$\mathbf{s} = \mathbf{s}_{1,1} + \cdots + \mathbf{s}_{n,1}, \quad \mathbf{s} \cdot \mathbf{s} = \mathbf{s}_{1,2} + \cdots + \mathbf{s}_{n,2}.$$

As remarked earlier this one-time setup procedure can be accomplished by standard UC-secure multiparty computation protocols such as that described in [3]. The following theorem is proved in the full version. It depends on the constant B defined above. In the full version we compute the value of B when the input ciphertext is $(B_{\text{plain}}, B_{\text{rand}}, C)$ -admissible, and show how to choose parameters for the cryptosystem such that the required bound on B is satisfied.

Theorem 4. *In the $\mathcal{F}_{\text{KEYGEN}}$ -hybrid model, the protocol Π_{DDEC} (Figure 8) implements $\mathcal{F}_{\text{KEYGENDEC}}$ with statistical security against any static active adversary corrupting up to $n - 1$ parties if $B + 2^{\text{sec}} \cdot B < q/2$.*

Protocol Π_{DDEC}

Initialize: Each party P_i on being given the ciphertext $c = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)$, and an upper bound B on the infinity norm of \mathbf{t} above, computes

$$\mathbf{v}_i \leftarrow \begin{cases} \mathbf{c}_0 - (\mathbf{s}_{i,1} \cdot \mathbf{c}_1) - (\mathbf{s}_{i,2} \cdot \mathbf{c}_2) & \text{if } i = 1 \\ -(\mathbf{s}_{i,1} \cdot \mathbf{c}_1) - (\mathbf{s}_{i,2} \cdot \mathbf{c}_2) & \text{if } i \neq 1 \end{cases}$$

and sets $\mathbf{t}_i \leftarrow \mathbf{v}_i + p \cdot \mathbf{r}_i$ where \mathbf{r}_i is a random element with infinity norm bounded by $2^{\text{sec}} \cdot B/(n \cdot p)$.

Public Decryption: All the players are supposed to learn the message.

- Each party P_i broadcasts \mathbf{t}_i
- All players compute $\mathbf{t}' \leftarrow \mathbf{t}_1 + \dots + \mathbf{t}_n$ and obtain a message $m' \leftarrow \text{decode}(\mathbf{t}' \bmod p)$.

Private Decryption: Only player P_j is supposed to learn the message.

- Each party P_i sends \mathbf{t}_i to P_j
- P_j computes $\mathbf{t}' \leftarrow \mathbf{t}_1 + \dots + \mathbf{t}_n$ and obtain a message $m' \leftarrow \text{decode}(\mathbf{t}' \bmod p)$.

Fig. 8. The distributed decryption protocol

Acknowledgements. The first, second and fourth author acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which [part of] this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.

The third author was supported by the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II and via an ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, by EPSRC via grant COED-EP/I03126X, the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number FA8750-11-2-0079, and by a Royal Society Wolfson Merit Award. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily

representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL, the U.S. Government, the European Commission or EPSRC.

The authors would like to thank Robin Chapman, Henri Cohen and Rob Harley for various discussions whilst this work was carried out.

References

1. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (2012)
2. Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.) *CRYPTO 1991*. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992)
3. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic Encryption and Multiparty Computation. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 169–188. Springer, Heidelberg (2011)
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)* 18, 111 (2011)
5. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) *CRYPTO 2011*. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
6. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
7. Damgård, I., Keller, M., Larraia, E., Miles, C., Smart, N.P.: Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. *IACR Cryptology ePrint Archive*, 2012:262 (2012)
8. Damgård, I.B., Nielsen, J.B.: Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In: Yung, M. (ed.) *CRYPTO 2002*. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
9. Damgård, I., Orlandi, C.: Multiparty Computation for Dishonest Majority: From Passive to Active Security at Low Cost. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 558–576. Springer, Heidelberg (2010)
10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178. ACM (2009)
11. Gentry, C., Halevi, S., Smart, N.P.: Fully Homomorphic Encryption with Polylog Overhead. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
12. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) STOC, pp. 21–30. ACM (2007)
13. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
14. Lindell, Y.: Highly-Efficient Universally-Composable Commitments Based on the DDH Assumption. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 446–466. Springer, Heidelberg (2011)

15. Myers, S., Sergi, M., Shelat, A.: Threshold fully homomorphic encryption and secure computation. IACR Cryptology ePrint Archive, 2011:454 (2011)
16. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. IACR Cryptology ePrint Archive, 2011:91 (2011)
17. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. IACR Cryptology ePrint Archive, 2011:133 (2011)
18. Winkler, S., Wullschleger, J.: On the Efficiency of Classical and Quantum Oblivious Transfer Reductions. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 707–723. Springer, Heidelberg (2010)

Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority

Eli Ben-Sasson¹, Serge Fehr², and Rafail Ostrovsky³

¹ Department of Computer Science, Technion, Haifa, Israel,
and Microsoft Research New-England, Cambridge, MA
`eli@cs.technion.ac.il`

² Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
`serge.fehr@cwi.nl`

³ Department of Computer Science and Department of Mathematics, UCLA
`rafail@cs.ucla.edu`

Abstract. In the setting of unconditionally-secure MPC, where dishonest players are unbounded and no cryptographic assumptions are used, it was known since the 1980’s that an honest majority of players is both necessary and sufficient to achieve privacy and correctness, assuming secure point-to-point and broadcast channels. The main open question that was left is to establish the exact communication complexity.

We settle the above question by showing an unconditionally-secure MPC protocol, secure against a dishonest minority of malicious players, that matches the communication complexity of the best known MPC protocol in the honest-but-curious setting. More specifically, we present a new n -player MPC protocol that is secure against a computationally-unbounded malicious adversary that can adaptively corrupt $t < n/2$ of the players. For polynomially-large binary circuits that are not too unshaped, our protocol has an amortized communication complexity of $O(n \log n + \kappa/n^{\text{const}})$ bits per multiplication (i.e. AND) gate, where κ denotes the security parameter and $\text{const} \in \mathbb{Z}$ is an arbitrary non-negative constant. This improves on the previously most efficient protocol with the same security guarantee, which offers an amortized communication complexity of $O(n^2\kappa)$ bits per multiplication gate. For any κ polynomial in n , the amortized communication complexity of our protocol matches the $O(n \log n)$ bit communication complexity of the best known MPC protocol with passive security.

We introduce several novel techniques that are of independent interest and we believe will have wider applicability. One is a novel idea of computing authentication tags by means of a *mini MPC*, which allows us to avoid expensive double-sharings; the other is a *batch-wise multiplication verification* that allows us to speedup Beaver’s “multiplication triples”.

1 Introduction

Background. In secure multiparty computation (MPC), a set of n players wish to evaluate an arbitrary but fixed function F on private inputs. The function F

is known to all the players and it is typically given as an arithmetic circuit \mathcal{C} over some finite field \mathbb{F} . It should be guaranteed that the inputs remain private and at the same time that the output of the computation is correct, even in the presence of an *adversary* that can *corrupt* a certain number t of the players. In case of a *passive* adversary, corrupt players simply reveal all their information to the adversary but otherwise keep following the protocol specification; in case of an *active* adversary, a corrupt player is under full control of the adversary and may arbitrarily misbehave during the protocol execution. By default, the goal is to obtain security against an active adversary.

The problem of MPC was initially introduced by Yao [23], with the first generic solutions presented in [17,9]. These first protocols offered cryptographic (aka. computational) security, meaning that the adversary is assumed to be computationally bounded, and can tolerate up to $t < n/2$ corrupt players. Subsequently, it was shown in [8,5] that in a setting with perfectly-secure point-to-point communication and with up to $t < n/3$ corrupt players, MPC is possible with unconditional and even perfect security.¹ Finally, in [21,1] it was shown that if a secure broadcast primitive is given — in addition to the secure point-to-point communication — then unconditionally (but not perfectly) secure MPC is possible against up to $t < n/2$ corrupt players.

These early results showed that MPC is possible in principle (in different settings), but they perform rather poorly in terms of communication complexity, i.e., the number of bits that the players need to communicate throughout the protocol. Over the years, a lot of effort has been put into improving the communication complexity of MPC protocols. The table in Figure 1 shows recent achievements and the state of the art in the settings $t < n/2$ (cryptographic or with broadcast) and $t < n/3$ (perfect or unconditional, without broadcast). Additional efficiency improvements are possible if one is willing to sacrifice on the resilience and lower the corruption threshold t by a small constant fraction, as shown in [13,15,14]. Indeed, lowering t enables to apply several powerful tools, like *packed secret sharing* or *committee selection*. We do not consider this option here, but aim for optimal resilience.

We can see from Figure 1 that there is a significant discrepancy between the cryptographic setting with $t < n/2$, or, similarly, the unconditional/perfect setting with $t < n/3$, versus the unconditional setting with $t < n/2$. In the former, MPC is possible for binary circuits with a near-linear amortized communication complexity of $O(n \log n)$ bits per multiplication gate.² In the latter, the best known protocol has an amortized communication complexity of $O(n^2\kappa)$ bits per multiplication gate. This is not very surprising, since it is probably fair to say that the unconditional setting with $t < n/2$ is the most difficult one to deal with. The reason is that no cryptographic tools can be used, like commitments

¹ Unconditional/perfect security means a computationally unbounded adversary and negligible/zero failure probability.

² By *amortized* communication complexity we mean under the assumption that the circuit is large enough so that the terms that are independent of the size of the circuit are irrelevant.

Adv	Resilience	Security	Communication	Ref
passive	$t < n/2$	perfect	$O(c_M n \log n + n^2 \log n)$	[16]
active	$t < n/2$	cryptographic	$O(c_M n^2 \kappa + n^3 \kappa)$	[19]
active	$t < n/2$	cryptographic	$O(c_M n \kappa + n^3 \kappa)$	[20]
active	$t < n/2$	cryptographic	$O(c_M n \log n) + \text{poly}(n\kappa)$	[16]
active	$t < n/3$	unconditional	$O(c_M n^2 \kappa) + \text{poly}(n\kappa)$	[18]
active	$t < n/3$	unconditional	$O(c_M n \log n + d_M n^2 \log n) + \text{poly}(n\kappa)$	[16]
active	$t < n/3$	perfect	$O(c_M n \log n + d_M n^2 \log n + n^3 \log n)$	[4]
active	$t < n/2$	unconditional	$O(c_M n^5 \kappa + n^4 \kappa) + O(c_M n^5 \kappa) \mathcal{BC}$	[10]
active	$t < n/2$	unconditional	$O(c_M n^2 \kappa + n^5 \kappa^2) + O(n^3 \kappa) \mathcal{BC}$	[3]

Fig. 1. Comparison of recent MPC protocols for binary circuits. n denotes the number of players, κ the security parameter (which we assume to be $\geq \log n$), c_M the number of multiplication gates in the circuit (which we assume dominates the number of in- and outputs), and d_M the multiplicative depth of the circuit. The communication complexity counts the number of bits that are communicated in total in an execution, plus, in the setting where a broadcast primitive is needed, the number of bits broadcasted. For circuits over a larger field \mathbb{F} , the $\log n$ -terms should be replaced by $\log(\max\{n, |\mathbb{F}|\})$.

or signatures, as in the cryptographic setting, nor can we use techniques from error correcting codes, as in the case $t < n/3$. Therefore, achieving near-linear amortized communication complexity for the setting of unconditional security and $t < n/2$ has remained a challenging open problem.

We note that, in any of the three settings, $O(n \log n)$ bits per multiplication gate seems to be hard to beat, since not even the best known protocol with *passive* security [16] does better than that.

Our Result. For an arbitrary arithmetic circuit over a finite field \mathbb{F} , we show a novel MPC protocol with unconditional security and corruption threshold $t < n/2$, which has a communication complexity of $O(c_M(n\phi+\kappa)+d_M n^2 \kappa + n^7 \kappa)$ bits plus $O(n^3 \kappa)$ broadcasts, where $\phi = \max\{\log n, \log |\mathbb{F}|\}$. Hence, for binary circuits that are not too “narrow” (meaning that the multiplicative depth d_M is sufficiently smaller than the number of multiplication gates), our protocol achieves an amortized communication complexity of $O(n \log n + \kappa)$ bits per multiplication gate. Furthermore, for any non-negative constant $const \in \mathbb{Z}$, a small modification to our protocol gives $O(n \log n + \kappa/n^{const})$ bits per multiplication gate, so that if $\kappa = O(n^{const+1})$, i.e., κ is at most polynomial in n , we obtain an amortized communication complexity of $O(n \log n)$ bits. Thus, our results show that even in the challenging setting of unconditional security with $t < n/2$, near-linear MPC is possible. Unless there is an additional improvement in the passive setting, this pretty much settles the question of the asymptotic complexity of unconditionally-secure MPC.

We would like to point out that the restriction on the multiplicative depth of the circuit, necessary for the claimed near-linear communication complexity per multiplication gate to hold, is also present in the easier $t < n/3$ setting for the protocols with near-linear communication complexity [16,4]; whether it is an inherent restriction is not known.

Techniques. We borrow several techniques from previous constructions of efficient MPC protocols. For instance, we make use of the *dispute control* technique introduced in [3], and the (near) linear passively-secure multiplication technique from [16]. However, our new protocol and its near-linear amortized communication complexity is to a great extent due to two new techniques, which we briefly discuss here. More details will be given in Section 2.7 and the full version [6].

Efficient batch verification of multiplication triples. The first technique allows to efficiently verify that a large list of N shared multiplication-triples are correct, i.e., satisfy the required multiplicative relation. These multiplication triples are used in order to implement Beaver’s method of evaluating multiplication gates, and our new protocol allows us to guarantee all N triples in one shot using communication complexity that is (nearly) independent of N .

Our new technique is inspired by a method that plays an important role in the construction of PCP proofs. Given oracle access to three sequences of bits, or elements from a “small” finite field, $a^1, \dots, a^N, b^1, \dots, b^N$ and c^1, \dots, c^N , we wish to verify that $a^i \cdot b^i = c^i$ for all $i = 1, \dots, N$. The procedure should be query-efficient, i.e., (much) more efficient than when querying and verifying all triples. Suppose the triples are encoded as low-degree polynomials. This means, we are given oracle access to evaluations of polynomials f and g of degree $< N$ and h of degree $< 2N - 1$, with $f(x_i) = a^i$, $g(x_i) = b^i$ and $h(x_i) = c^i$ for all $i \in \{1, \dots, N\}$, where x_1, \dots, x_N are fixed disjoint points and h is supposed to be $h = f \cdot g$. The key observation is this: by the fundamental theorem of algebra, if $f \cdot g \neq h$ then $f(\sigma) \cdot g(\sigma) \neq h(\sigma)$ except with probability at most $\frac{2N-1}{|\mathbb{K}|}$ for a randomly chosen $\sigma \in \mathbb{K}$, and for any suitably large extension field \mathbb{K} .

In our setting, it will turn out that we can indeed enforce the shared multiplication triples to be encoded via low-degree polynomials as above. So, by the above technique, it is possible to verify N multiplication triples with just *one* (random) query to f, g and h , and thus with a communication complexity that essentially only depends on the aspired error probability.

In independent work [12], Cramer *et al.* propose a 2-party batch zero-knowledge proof for committed multiplication triples. The techniques used there show some resemblance, but there are also differences due to the fact that in our setting, the a^i, b^i and c^i ’s are not known to any party.

Multiparty-computing the authentication tags. Our other technique is a new way to “commit” the players to their shares, so that dishonest players who lie about their shares during reconstruction are caught. This is necessary in the setting $t < n/2$, where plain Shamir shares do not carry enough redundancy to reconstruct in the presence of incorrect shares.

The way we “commit” player P_i to his share σ_i is by attaching an *authentication tag* τ to σ_i , where the corresponding *authentication key* is held by some other player V , acting as *verifier*.³ The reader may think of τ as $\tau = \mu \cdot \sigma_i + \nu$ over some large finite field, where (μ, ν) forms the key. It is well known and easy

³ Actually, σ_i comes along with n tags, one for each player acting as verifier V .

to see that if P_i does not know the key (μ, ν) , then he is not able to come up with $\sigma'_i \neq \sigma_i$ and τ' such that $\tau' = \mu \cdot \sigma'_i + \nu$, except with small probability. Thus, incorrect shares can be detected and filtered out.

This idea is not new, and actually goes back to [21], but in all previous work the tag τ is *locally* computed by some party, usually the dealer that prepared the share σ_i . Obviously, this requires that the dealer *knows* the key (μ, ν) ; otherwise, he cannot compute $\tau = \mu \cdot \sigma_i + \nu$. As a consequence, if the dealer is dishonest, the authentication tag τ is useless, because with the knowledge of the key, an authentication tag τ' for an incorrect share σ'_i can easily be forged. In previous work, as in [21,10,3], this problem was overcome by means of a *double* sharing, where every share σ_i is again shared, and the authentication tags are attached to the second-level shares. However, such a double sharing obviously leads to a (at least) quadratic communication complexity.

Instead, here we propose to compute the tag τ by means of a *mini MPC*, to which P_i provides his share σ_i as input, and V his key (μ, ν) , and the tag τ is securely computed jointly by all the players. This way, no one beyond V learns the key (μ, ν) , and forging a tag remains hard, and no expensive double sharing is necessary.

At first glance this may look hopeless since MPC typically is very expensive, and we cannot expect to increase the efficiency of MPC by using an expensive MPC as subprotocol. What saves us is that our mini MPC is for a very specific function in a very specific setting. We use several tricks, like re-using parts of the authentication key, batching etc., to obtain a *tailored* mini MPC for computing the tag τ , with an amortized communication complexity that has no significant impact. One of the crucial new tricks is to make use of the fact that Shamir's secret sharing scheme is "symmetric" in terms of what is the shared secret and what are the shares; this allows us to avoid having to re-share the share σ_i for the mini MPC, but instead we can use the other shares σ_j as shares of σ_i .

2 Near-Linear MPC: Our Result and Approach

2.1 Communication and Corruption Model

We consider a set of $n = 2t + 1$ players P_1, \dots, P_n , which are connected by means of a complete network of secure synchronous communication channels. Additionally, we assume a broadcast channel, available to all the players. For simplicity, we assume the broadcast channel to broadcast single bits; longer messages are broadcasted bit-wise. For a protocol that instructs the players to communicate (in total) X bits and to broadcast Y bits, we say that the protocol has communication complexity $X + Y \cdot BC$.

We consider a computationally-unbounded active adversary that can adaptively corrupt up to t of the players. Adaptivity means that the adversary can corrupt players during the execution of the protocol, and depending on the

information gathered so far. Once a player is corrupted, the adversary learns the internal state of the player, which consists of the complete history of that player, and takes over full control of that player and can make him deviate from the protocol in any desired manner.

For any given arithmetic circuit \mathcal{C} over a finite field \mathbb{F} , the goal is to have a protocol that permits the n players to securely evaluate \mathcal{C} on their private inputs. For simplicity, we assume that all the players should learn the entire result. Security means that the adversary cannot influence the result of the computation more than by selecting the inputs for the corrupt players, and the adversary should learn nothing about the uncorrupt players' inputs beyond what can be deduced from the result of the computation. This should hold unconditionally, meaning without any computational restrictions on the adversary, and up to a negligible failure probability ε .

2.2 Main Result

For an arithmetic circuit \mathcal{C} over a finite field \mathbb{F} , we denote the respective numbers of input, output, addition, and multiplication gates in \mathcal{C} by c_I, c_O, c_A , and c_M , and we write $c_{tot} = c_I + c_O + c_M$ (not counting c_A). Furthermore, we write d_M to denote its multiplicative depth, i.e., the maximal number of multiplication gates on any path from an input gate to an output gate.

Theorem 1. *For every $n, \kappa \in \mathbb{N}$, and for every arithmetic circuit \mathcal{C} over a finite field \mathbb{F} with $|\mathbb{F}| \leq 2^{\kappa+n}$, there exists an n -party MPC protocol that securely computes \mathcal{C} against an unbounded active adaptive adversary corrupting up to $t < n/2$ players, with failure probability $\varepsilon \leq O(c_{tot}n)/2^\kappa$ and communication complexity $O(c_{tot} \cdot (n\phi + \kappa) + d_M n^2 \kappa + n^7 \kappa) + O(n^3 \kappa) \cdot \mathcal{BC}$, where $\phi = \max\{\log |\mathbb{F}|, \log n\}$. More generally, for any $const \in \mathbb{Z}$, there exists such a MPC protocol with communication complexity $O(c_{tot} \cdot (n\phi + \kappa/n^{const}) + d_M n^2 \kappa + n^7 \kappa) + O(n^3 \kappa) \cdot \mathcal{BC}$.*

Theorem 1 guarantees that for large enough circuits that are not too “narrow”, meaning that the multiplicative depth d_M is significantly smaller than the number c_M of multiplication gates (e.g. $d_M \leq c_M/(n\kappa)$ is good enough), the communication complexity per multiplication gate (assuming that c_M dominates c_I, c_O and c_R) is $O(n\phi + \kappa/n^{const})$ bits, i.e., $O(n \log n + \kappa/n^{const})$ for binary circuits, for an arbitrary non-negative $const \in \mathbb{Z}$. Recall, the best previous MPC scheme in this setting [3] required $O(n^2 \kappa)$ bits per multiplication gate. For simplicity, we focus on the case $const = 0$ and merely give some indication on how to adapt the same for larger $const$.

2.3 The Set Up

We are given positive integers n and κ , and an arithmetic circuit \mathcal{C} over a finite field \mathbb{F} . We assume that $|\mathbb{F}| \geq 2n^2$ (or $|\mathbb{F}| \geq 2n^{2+const}$ for an arbitrary $const$)

— otherwise we consider \mathcal{C} over an appropriate extension field⁴ — and we write $\phi = \log(|\mathbb{F}|)$, i.e., ϕ denotes the number of bits needed to represent an element in \mathbb{F} . We may assume that $\kappa \geq n$ (otherwise, we set $\kappa = n$) and thus that κ is an integer multiple of n . We fix an extension field \mathbb{K} of \mathbb{F} such that $|\mathbb{K}| \geq 2^{2(\kappa+n)}$. Finally, we set $M = 2(c_M + c_R + c_I)$.

As convention, we write elements in \mathbb{F} as Roman letters, and elements in \mathbb{K} as Greek letters. Note that \mathbb{F} is naturally a subset of \mathbb{K} , and thus for $s \in \mathbb{F}$ and $\lambda \in \mathbb{K}$, the product $\lambda \cdot s$ is a well defined element in \mathbb{K} . Also note that by fixing an \mathbb{F} -linear bijection $\mathbb{F}^e \rightarrow \mathbb{K}$, where e is the extension degree $e = [\mathbb{K}:\mathbb{F}]$ we can understand a vector $(s^1, \dots, s^e) \in \mathbb{F}^e$ as a field element $\sigma \in \mathbb{K}$, and a vector $(s^1, \dots, s^{q \cdot e}) \in \mathbb{F}^{q \cdot e}$ for $q \in \mathbb{N}$ as a vector $\sigma = (\sigma^1, \dots, \sigma^q) \in \mathbb{K}^q$ of q field elements in \mathbb{K} .

2.4 Dispute Control

We make use of the *dispute control* framework due to Beerliová-Trubíniová and Hirt. The idea of dispute control is to divide (the different phases of) the MPC protocol into n^2 *segments* (of equal “size”), and to execute the segments sequentially. If the execution of a segment should fail due to malicious behavior of some corrupt parties, then two players are identified that are in dispute and of which at least one must be corrupt. Then, the failed segment is freshly re-executed, but now in such a way that the two players in dispute will *not* be able to get into dispute anymore, during this segment and during all the remaining segments. This ensures that overall there can be at most n^2 disputes (actually fewer, because two uncorrupt players will never get into a dispute), and therefore at most n^2 times a segment needs to be re-executed. This means that overall there are at most $2n^2$ executions of a segment.

We will show that (if d_M is small enough) any segment of size $m = M/n^2$ can be executed with bit communication complexity $O(m(n\phi + \kappa) + n^5\kappa) + O(n\kappa) \cdot \mathcal{BC}$; it thus follows that the communication complexity of the overall scheme is $2n^2 \cdot O(m(n\phi + \kappa) + n^5\kappa) = O(M(n\phi + \kappa) + n^7\kappa)$ bits plus $O(n^3\kappa) \cdot \mathcal{BC}$, which amounts to $O(n\phi + \kappa)$ bits per multiplication gate for large enough circuits.

A dispute between two players P_i and P_j typically arises when player P_j claims to have received message msg from P_i whereas P_i claims that he had actually sent $msg' \neq msg$ to P_j . In order to ensure that two players P_i and P_j in dispute will not get into a new dispute again, they will not communicate anymore with each other. This is achieved by means of the following two means:

- (1) If P_i is supposed to share a secret w and distribute the shares to the players, then he chooses the sharing polynomial so that P_j 's share w_j vanishes, and thus there is no need to communicate the share, P_j just takes $w_j = 0$ as his share. Using the terminology from [3], we call such a share that is enforced to be 0 a *Kudzu* share (see also Section 2.5).

⁴ In this case one has to make sure that the inputs provided by the players belong to the original base field; this can easily be taken care of by means of our techniques, without increasing the asymptotic communication complexity.

- (2) For other messages that P_i needs to communicate to P_j , he sends to P_j via a *relay*: the first player P_r that is not in dispute with P_i and not with P_j .

In order to keep track of the disputes and the players that were caught cheating, the players maintain two sets, Corr and Disp , which at the beginning of the execution are both initialized to be empty. Whenever the players jointly identify a player P_i to be corrupt, then P_i is added to Corr . Additionally, $\{P_i, P_j\}$ will be added to Disp for every $j \in \{1, \dots, n\}$. Whenever there is a dispute between two players P_i and P_j , so that one of them must be corrupt but it cannot be resolved which of the two, then $\{P_i, P_j\}$ is added to Disp . Whenever a player P_i is in dispute with more than t players, then he must be corrupt and is added to Corr (and Disp is updated accordingly). We write Disp_i for the set of all players P_j with $\{P_i, P_j\} \in \text{Disp}$. Players that are in dispute (with some other players) still take part in the protocol, but they do not communicate anymore with each other. Players in Corr , i.e., players that have been identified to be corrupt, are excluded from (the remainder of) the protocol execution. We do not always make this explicit in the description of the protocol when we quantify over all players but actually mean all players not in Corr . Also, we do not make it always explicit but understand it as clear that whenever a new dispute is found, the remainder of the execution of the current segment is skipped, and the segment is freshly executed with the updated Disp (and Corr).

2.5 The Different Sharings

We will be using different variants and extensions of Shamir's secret sharing scheme [22]. We introduce here these different versions and the notation that we will be using for the remainder of the paper. We consider the field \mathbb{F} from Section 2.3, and fix distinct elements $x_0, x_1, \dots, x_n \in \mathbb{F}$ with $x_0 = 0$. We also fix an additional $2n^2 - n - 1$ elements $x_{n+1}, \dots, x_{2n^2-1}$ with the property that every pair x_i, x_j with $i \neq j \in \{0, \dots, 2n^2-1\}$ is disjoint; these additional elements will be used later on. It may be convenient to view the different kinds of sharings we introduce below as different *data structures* for representing an element $w \in \mathbb{F}$ by data held among the players.

- A *degree- t (Shamir) sharing* of $w \in \mathbb{F}$ consists of n shares $w_1, \dots, w_n \in \mathbb{F}$ of the following form: there exists a sharing polynomial $f(X) \in \mathbb{F}[X]$ of degree at most t such that $w = f(0)$ and $w_j = f(x_j)$ for $j \in \{1, \dots, n\}$. Furthermore, share w_j is held by player P_j for $j \in \{1, \dots, n\}$. We denote such a sharing as $[w]$. If a designated player P_d (e.g. the dealer) knows all the shares, and thus also w , we indicate this by denoting the sharing as $[w]_d$.
- A *degree- $2t$ (Shamir) sharing* of $w \in \mathbb{F}$ is defined as the degree- t sharing above, except that the degree of the sharing polynomial f is at most $2t$. We write $\langle w \rangle$ for such a sharing, and $\langle w \rangle_d$ for such a sharing when P_d knows all the shares.
- A *twisted degree- t* sharing of $w \in \mathbb{F}$ with respect to player P_i , denoted as $[w|_i^i]$, consists of $n - 1$ shares $w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n \in \mathbb{F}$, of the following form: there exists a sharing polynomial $f(X) \in \mathbb{F}[X]$ of degree at most t

such that $w = f(x_i)$, $f(0) = 0$, and $w_j = f(x_j)$ for $j \in \{1, \dots, n\} \setminus \{i\}$.⁵ Share w_j for $j \in \{1, \dots, n\} \setminus \{i\}$ is known to player P_j . We write $\lceil w \rceil_d^i$ for such a sharing when P_d knows all the shares.

- A *twisted* degree- $2t$ sharing of $w \in \mathbb{F}$ with respect to P_i , denoted as $\langle w \rangle^i$ respectively $\langle w \rangle_d^i$ when P_d knows all the shares, is defined as the twisted degree- t sharing above, except that the degree of the sharing polynomial f is at most $2t$.
- A *two-level (degree- t /sum) sharing* $\llbracket w \rrbracket$ consists of n degree- t Shamir sharings $[w(1)]_1, \dots, [w(n)]_n$ with $w = \sum_d w(d)$.⁶ The shares $w_1(d), \dots, w_n(d)$ given by $[w(d)]_d$ for $d \in \{1, \dots, n\}$ then define a degree- t sharing $[w]$ of w by means of $w_j = \sum_d w_j(d)$ for $j \in \{1, \dots, n\}$ (see Figure 2, left). We point out that the second level shares $w_i(d)$ can be understood as Shamir shares of the sum-shares $w(d)$ of w , as well as sum-shares of the Shamir shares w_i of w .
- A *two-level (degree- $2t$ /sum) sharing* $\langle\langle w \rangle\rangle$ is defined similar to above as $\langle\langle w \rangle\rangle = (\langle w(1) \rangle_1, \dots, \langle w(n) \rangle_n)$ with $w = \sum_d w(d)$.

The above list merely specifies the structures of the different sharings, but does not address privacy. In our scheme, the different sharings will be prepared in such a way that the standard privacy requirement holds: the shares of any t players reveals no information on the shared secret. In the case of a *twisted* sharing $\lceil w \rceil^i$, privacy is slightly more subtle. Because player P_i is given no share, but, on the other hand, the sharing polynomial vanishes at 0, privacy will only hold in case P_i is (or gets) corrupted, so that the t corrupted players miss one polynomial evaluation; this will be good enough for our purpose.

We note that the players can, by means of local computations, perform certain computations on the sharings. For instance, by linearity of Shamir's secret sharing scheme, it follows that if the players locally add their shares of a degree- t sharing $[v]$ of v to their shares of a degree- t sharing $[w]$ of w , then they obtain a degree- t sharing $[v+w]$ of $v+w$. We denote this computation as $[v] + [w] = [v+w]$. Also, multiplication with a known constant: $c[w] = [cw]$, or adding a known constant: $[w] + d = [w+d]$, can be performed by means of local computations. This holds for all the different sharings discussed above: $\langle v \rangle + c\langle w \rangle + d = \langle v + cw + d \rangle$, $\llbracket v \rrbracket + c\llbracket w \rrbracket + d = \llbracket v + cw + d \rrbracket$ etc. Furthermore, locally multiplying the shares of two degree- t shared secrets results in a degree- $2t$ sharing of the product: $[v] \cdot [w] = \langle v \cdot w \rangle$. Finally, locally multiplying the shares $[v]$ of an ordinarily degree- t shared secret with the shares $\lceil w \rceil^i$ of a twisted degree- t shared secret results in a twisted degree- $2t$ sharing of the product of P_i 's share v_i of $[v]$ and w : $[v] \cdot \lceil w \rceil^i = \langle v_i \cdot w \rangle^i$. This property of a twisted sharing is of crucial importance to us; thus, we encourage the reader to verify this claim.

⁵ Thus, instead of plugging the secret into the evaluation at 0 (i.e. into the constant coefficient of f), we plug it into the evaluation at x_i , and require $f(0)$ to vanish and give player P_i no share.

⁶ We point out that $w(1), \dots, w(n)$ are simply n elements in \mathbb{F} , indexed by $d = 1, \dots, n$, that add up to w , and they should not be understood as function evaluations. Our convention is to write $w(1), \dots, w(n)$ as sum-shares of w , and w_1, \dots, w_n as Shamir shares of w , and $w_1(d), \dots, w_n(d)$ as Shamir shares of $w(d)$, etc.

We point out that opening such a product of sharings, like $\langle v \cdot w \rangle = [v] \cdot [w]$, reveals more information on v and w than just their product. This will be of no concern to us, because in our scheme, such sharings will only be opened in the form of $\langle u + v \cdot w \rangle = \langle u \rangle + [v] \cdot [w]$, i.e., when masked with a random degree- $2t$ sharing, which ensures that no information on u, v, w is revealed beyond $u + v \cdot w$.

Borrowing the terminology from [3], we say that a sharing $[s]_d$ has *Kudzu* shares, if the share s_j of every player P_j that currently is in \mathcal{Disp}_d is set to $s_j = 0$, i.e., the sharing polynomial $f(x)$ is such that $f(x_j) = 0$ for every $P_j \in \mathcal{Disp}_d$. The same terminology correspondingly applies to sharings $\langle s \rangle_d$, $[s]_d^i$ and $\langle s \rangle_d^i$. Furthermore, a two-level sharing $\llbracket s \rrbracket$ is said to have Kudzu shares if $[s(d)]_d$ has Kudzu shares for all $P_d \notin \mathcal{Corr}$, and $[s(d)]_d$ consist of all-0 shares for all $P_d \in \mathcal{Corr}$, and similarly for $\langle\langle s \rangle\rangle$.

Finally, we would like to point out that due to the linearity, e sharings $[s^1], \dots, [s^e]$ of secrets $s^1, \dots, s^e \in \mathbb{F}$ can also be understood and treated as a sharing $[\sigma]$ of $\sigma = (s^1, \dots, s^e)$, viewed as an element in \mathbb{K} and with shares $\sigma_i \in \mathbb{K}$, by means of a sharing polynomial $f(X) \in \mathbb{K}[X]$, but with the same interpolation points $x_1, \dots, x_n \in \mathbb{F} \subseteq \mathbb{K}$.

2.6 Protocol Overview

The protocol consists of three phases: the *preparation phase*, the *input phase*, and the *computation phase*. We briefly discuss (the goal of) these three phases here. As discussed in Section 2.4, every phase will be performed in segments; and whenever a segment fails, then a new dispute is found and added to \mathcal{Disp} , and the segment is re-executed.

Preparation Phase. In this phase, the following data structure is prepared.

Two-level shared multiplication triples: A list \mathcal{M} of M correctly two-level shared triples $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$, where for every⁷ $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket) \in \mathcal{M}$, the values a and b are uniformly distributed in \mathbb{F} (and independent of each other and of the other triples in \mathcal{M}) and unknown to the adversary, and $c = a \cdot b$. We write $\cup \mathcal{M}$ for the list of $\llbracket a \rrbracket$, $\llbracket b \rrbracket$ and $\llbracket c \rrbracket$ sharings contained in \mathcal{M} , i.e., $\cup \mathcal{M} = \bigcup_{(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket) \in \mathcal{M}} \{\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket\}$, where the union is over all $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket) \in \mathcal{M}$

Local base sharings: The two-level sharings of the multiplication triples are not fully independent. Instead, for every player P_d there exists a list $\mathcal{S}(d)$ of $L = O(M/n)$ so-called *local base sharings* $[s(d)]_d$ with $s(d) \in \mathbb{F}$, such that for every $\llbracket w \rrbracket \in \cup \mathcal{M}$, the sharing $[w(d)]_d$ (which is part of $\llbracket w \rrbracket$) is a linear combination (with known coefficients) of the local base sharings:⁸

$$[w(d)]_d = \sum_{s(d) \in \mathcal{S}(d)} u_{s(d)} [s(d)]_d + u_\circ.$$

⁷ We use set-notation for lists: for a list $\mathcal{L} = (\ell_1, \dots, \ell_m)$, the expression $\ell \in \mathcal{L}$ is understood as ℓ_i for $i \in \{1, \dots, m\}$. Also, $\sum_{\ell \in \mathcal{L}} u_\ell \ell$ should be understood as $\sum_{i=1}^m u_i \ell_i$.

⁸ As a consequence, even though every player implicitly holds in total $3Mn$ subshares of the $\llbracket w \rrbracket \in \cup \mathcal{M}$, he only needs to explicitly store $n \cdot L = O(M)$ values. Thus, to communicate all these subshares (for all the players), only $O(Mn)$ elements in \mathbb{F} need to be communicated, i.e., a linear number per multiplication triple.

Although there are dependencies among the second-level shares of different $\llbracket w \rrbracket \in \cup\mathcal{M}$ (which means we have to pay special attention when revealing those, or the local base sharings), it will be the case that the first-level Shamir sharings $[w]$ are independent among all $\llbracket w \rrbracket \in \cup\mathcal{M}$.

For every P_d , the list $\mathcal{S}(d)$ will be divided into n^3 blocks, each block containing L/n^3 sharings $[s(d)]_d$ from $\mathcal{S}(d)$. Each such block, we can write as $[\sigma(d)]_d$ with $\sigma(d) \in \mathbb{K}^q$, and understand it as a list of $q = L/(n^3e)$ sharings $[\sigma(d)]_d$ of elements $\sigma(d) \in \mathbb{K}$, where $e = [\mathbb{K} : \mathbb{F}]$. As such, $\mathcal{S}(d)$ can now be understood as a list of n^3 sharings $[\sigma(d)]_d$.⁹

Authentication tags: For every player P_d , every block $[\sigma(d)]_d \in \mathcal{S}(d)$, every player P_i holding the shares $\sigma_i(d) \in \mathbb{K}^q$ of block $[\sigma(d)]_d$, and every player P_V (acting as verifier), the following holds. P_V holds a random *long-term authentication key* $\mu \in \mathbb{K}^q$ and a random *one-time authentication key* $\nu \in \mathbb{K}$, and P_i holds the (*one-time*) *authentication tag*

$$\tau = \mu \odot \sigma_i(d) + \nu \in \mathbb{K},$$

where \odot denotes the standard inner product over \mathbb{K} . We stress that ν and, consequently, τ are fresh for every P_d , every block $[\sigma(d)]_d \in \mathcal{S}(d)$, and every P_i and P_V , but μ is somewhat re-used: P_V uses the same μ for every P_d (but fresh μ 's for different P_i 's) and for n out of the n^3 blocks $[\sigma(d)]_d \in \mathcal{S}(d)$.¹⁰

This data structure is illustrated in Figure 2.

$$\begin{array}{ccccccc}
 w & \rightarrow & w(1) & w(2) & \cdots & w(n) & w(d) \\
 \Downarrow & & \Downarrow & \Downarrow & \cdots & \Downarrow & \Downarrow \\
 w_1 & \rightarrow & w_1(1) & w_1(2) & \cdots & w_1(n) & w_1(d) \\
 w_2 & \rightarrow & w_2(1) & w_2(2) & \cdots & w_2(n) & w_2(d) \\
 \vdots & & \vdots & \vdots & & \vdots & \vdots \\
 w_n & \rightarrow & w_n(1) & w_n(2) & \cdots & w_n(n) & w_n(d)
 \end{array}
 \in \text{span} \left\{ \begin{array}{l} s(d) \\ \Downarrow \\ s_1(d) \\ s_2(d) \\ \vdots \\ s_n(d) \end{array} \right\} \quad \tau = \mu \odot \sigma_i(d) + \nu$$

Fig. 2. For every multiplication triple $(a, b, c) \in \mathcal{M}$, every $w \in \{a, b, c\}$ is two-level shared as $\llbracket w \rrbracket$ (left), and $[w(d)]_d$ is a linear combination of P_d 's local base sharings $[s_i(d)]_d$ (center), and $s_i(d)$ is authenticated within a batch $\sigma_i(d)$ (right)

The purpose of the *authentication tags* (and *keys*) is to be able to identify an incorrect share $\sigma_i(d)$ claimed by a corrupt player P_i . Indeed, it is well known (and goes back to Carter and Wegman [7]) that if the adversary has no information on μ beyond knowing the tags τ for several $\sigma_i(d)$ with fresh one-time keys ν ,

⁹ We silently assume here that the fraction $L/(n^3e)$ is an integer, and we will similarly do so for a few other fractions later. We may always do so without loss of generality.

¹⁰ As a consequence, the total number of fresh one-time keys μ equals the total number of $\sigma(d)$'s (over all d 's), and thus sharing them (which will be needed) does not increase the overall asymptotic communication complexity.

then the probability for the adversary to produce $\sigma'_i(d) \neq \sigma_i(d)$ and τ' with $\tau' = \mu \odot \sigma'_i(d) + \nu$ is at most $1/|\mathbb{K}| \leq 2^{-2(\kappa+n)}$. Informally, this means that with the given data structure, a dishonest player P_i will not be able to lie about his share $\sigma_i(d)$ without being caught.

The use of authentication tags to (try to) commit players to their (sub)share is not new. What distinguishes our approach from previous work is that here the tag τ will be computed in a *multi-party fashion* so that no one beyond the verifier P_V knows the corresponding key. This gives us the decisive advantage over previous work.

Input Phase. For every player P_i , and for every input $x \in \mathbb{F}$ of that player to the circuit, a fresh multiplication triple $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$ is chosen from \mathcal{M} , and a is reconstructed towards P_i . Then P_i announces $d = x - a$, and the players compute the sharing $\llbracket x \rrbracket = d + \llbracket a \rrbracket$. The used triple $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$ is then removed from \mathcal{M} .

Essentially the only thing corrupt players can do to disrupt the computation phase, is to provide incorrect shares when P_i is supposed to reconstruct some shared a . However, because every $[a(d)]_d$ is a linear combination of the local base sharings $[s(d)]_d$, and because players are committed to their local base sharings (block-wise) by means of the authentication tags, players that hand in incorrect shares can be caught.

Computation Phase. The actual computation is done in a gate-by-gate fashion. To start with, we say that the input values are *computed*. Then, inductively, for every gate in the circuit whose input values have already been computed, the corresponding output value of the gate is computed. This is done as follows. Let $\llbracket x \rrbracket$ and $\llbracket y \rrbracket$ be the sharings of the input values to the gate. If the gate is an addition gate, then the output value is computed locally as $\llbracket z \rrbracket = \llbracket x + y \rrbracket = \llbracket x \rrbracket + \llbracket y \rrbracket$. If the gate is a multiplication gate, then the output value is computed by using Beavers technique [2] as follows. A fresh multiplication triple $\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket$ is selected and the differences $\llbracket x - a \rrbracket = \llbracket x \rrbracket - \llbracket a \rrbracket$ and $\llbracket y - b \rrbracket = \llbracket y \rrbracket - \llbracket b \rrbracket$ are reconstructed. Then, the output value of the gate is computed locally as $\llbracket z \rrbracket = \llbracket x \cdot y \rrbracket = (x - a)(y - b) + (x - a)\llbracket b \rrbracket + (y - b)\llbracket a \rrbracket - \llbracket c \rrbracket$. In the end, once the output values of the circuit have been computed, they are reconstructed.¹¹

Essentially the only thing corrupt players can do to disrupt the computation phase, is to provide incorrect shares when the players (try to) reconstruct a shared value $\llbracket w \rrbracket$. Since the latter is a linear combination of sharings in $\cup \mathcal{M}$ so that every $[w(d)]_d$ is a linear combination of the local base sharings $[s(d)]_d$, and because players are committed to their local base sharings (block-wise) by the authentication tags, players that hand in incorrect shares can be caught.

2.7 Two New Essential Ingredients

We present here the two main new components that enable our improved communication complexity.

¹¹ For simplicity we assume that all the players are supposed to learn all output values of the circuit. It is straightforward to adjust our scheme so that different players learn different output values.

Batch-Wise Multiplication Verification. Assume we have two sharings $[a]$ and $[b]$ (over \mathbb{F}), and the players have computed a sharing $[c]$, which is supposed to be $c = a \cdot b$, using an *optimistic* multiplication protocol (i.e., one that assumes that players behave). And now the players want to verify that indeed $c = a \cdot b$, without revealing anything beyond about a, b, c . The standard way of doing so (see e.g. [11] or [3]) has a failure probability of $1/|\mathbb{F}|$, which is too large for us, or when performed over the bigger field \mathbb{K} , has a sufficiently small failure probability of $1/|\mathbb{K}|$, but requires to share an element from \mathbb{K} for every triple to be verified. This means we get a communication complexity of at least $O(n\kappa)$ bits per multiplication gate, whereas we want $O(n\phi + \kappa)$.

We achieve the latter by verifying $c = a \cdot b$ *batch-wise*. This is done by means of the following method. Let $([a^1], [b^1], [c^1]), \dots, ([a^N], [b^N], [c^N])$ be $N = n^2$ multiplication triples that need to be verified. Consider the degree- $(N-1)$ polynomials f and g with $f(x_k) = a^k$ and $g(x_k) = b^k$ for all $k \in \{1, \dots, N\}$. The players can locally compute $[a^k]$ and $[b^k]$ with $f(x_k) = a^k$ and $g(x_k) = b^k$ for all $k \in \{N+1, \dots, 2N-1\}$. Furthermore, by using the optimistic multiplication protocol, we let them compute $[c^{N+1}], \dots, [c^{2N-1}]$ where c^k is supposed to be $a^k \cdot b^k$. Let h be the degree- $(2\ell-2)$ polynomial with $h(x_k) = c^k$ for all $k \in \{1, \dots, 2N-1\}$. It now holds that all the multiplication triples are correct — i.e., that $c^k = a^k \cdot b^k$ for all $k \in \{1, \dots, 2N-1\}$ — if and only if $h = f \cdot g$ as polynomials. In order to test if $h = f \cdot g$ or not, the players can simply choose a random challenge $\sigma \in \mathbb{K}$ and see if $h(\sigma) = f(\sigma) \cdot g(\sigma)$ or not. For the latter, the players locally compute their shares of $[f(\sigma)]$, $[g(\sigma)]$ and $[h(\sigma)]$ — each is a linear combination of the shares of f, g, h that the player holds — and apply the “expensive” standard multiplication verification to $[f(\sigma)]$, $[g(\sigma)]$ and $[h(\sigma)]$.

Multiparty Computation of the Tags. As mentioned before, the tags τ should be computed in a multi-party fashion, without blowing up the asymptotic communication complexity. To simplify the exposition here, we assume for the moment that each tag τ is computed as $\tau = \mu \cdot \sigma_i(d) + \nu$ for $\mu \in \mathbb{K}$, and where $\sigma_i(d) \in \mathbb{K}$ is the i -th share of $[\sigma(d)]$. A first step in a multi-party computation usually is to share the inputs; here: μ , $\sigma_i(d)$ and ν . However, this blows up the communication complexity by a factor n , which we cannot afford. Note that sharing μ is actually ok, since the μ 's are (partly) re-used, and thus we can also re-use their sharings. Also, sharing ν is ok, since in the actual authentication scheme we are using (not the simplified version we are discussing here), there is only one ν for many $\sigma_i(d)$'s. What is problematic, however, is the sharing of $\sigma_i(d)$. And this is where our second new method comes into play. We make use of the fact that $\sigma_i(d)$ is not an arbitrary input to the multi-party computation, but that it is actually a share of a shared secret $\sigma(d)$. Due to the symmetry of Shamir's secret sharing scheme, we may then view $\sigma_i(d)$ as the *secret* and the remaining shares $\sigma_j(d)$ as a sharing of $\sigma_i(d)$. Indeed, any $t+1$ of the shares $\sigma_j(d)$ can be used to recover $\sigma_i(d)$. Thus, in that sense, $\sigma_i(d)$ is already shared, and there is no need to share it once more.

Using this idea, the players can compute τ in a multi-party way as follows.¹² Player P_V , holding μ and ν , shares μ as a twisted degree- t sharing $\lceil \mu \rceil_V^i$, and ν as a twisted degree- $2t$ sharing $\langle \nu \rangle_V^i$. The players now locally compute $\lceil \mu \rceil_V^i \cdot [\sigma(d)] + \langle \nu \rangle_V^i$, which results in a twisted degree- $2t$ sharing $\langle \mu \cdot \sigma_i(d) + \nu \rangle^i$ of $\tau = \mu \cdot \sigma_i(d) + \nu$, as explained at the end of Section 2.5. These shares can now be sent to P_i for reconstruction (and correctness of τ will be verified by a cut-and-choose technique).

We point out that by corrupting t players P_j that do not include P_V or P_i , the adversary can learn μ from the (twisted) shares of the players in P_j . However, it that case, the adversary *cannot* anymore corrupt player P_i , and thus knowledge of μ is of no use. What is important is that the adversary does not learn μ in case it corrupts P_i , and this we will show to hold.

Adapting the above to $\tau = \mu \odot \sigma_i(d) + \nu$, and re-using μ and its twisted sharing, gives the players the means to compute their tags with a communication complexity that is negligible for large enough circuits.

We now give the detailed protocol for multiparty computing the tag τ . We assume μ to be shared (component-wise) as $\lceil \mu^1 \rceil_V^i, \dots, \lceil \mu^q \rceil_V^i$. The one-time key ν and the tag τ are chosen/computed by means of the following subprotocol, unless P_i is in dispute with P_d or with P_V . In the former case, his shares are fixed to 0 anyway, and in the latter, P_i and P_V accuse each other anyway. For simpler notation, we write $[\sigma]_d$ instead of $[\sigma(d)]_d$, etc.

Protocol. $\text{TagComp}_{V,i,d}$

Player P_V chooses a random $\nu \in \mathbb{K}$ and shares it (non-verifiably) as $\langle \nu \rangle_V^i$ with Kudzu shares. Similarly, player P_d shares $o = 0$ (i.e., zero) over \mathbb{K} as $\langle o \rangle_V^i$ with Kudzu shares. The players locally compute $\langle \tau \rangle^i = \sum_{k=1}^q [\sigma^k]_d \lceil \mu^k \rceil_V^i + \langle \nu \rangle_V^i + \langle o \rangle_d^i$ and send their shares to P_i . If $P_j \in \text{Disp}_i$, then P_j sends his share of $\langle \tau \rangle^i$ to P_i via a *relay*, i.e., via the first player that is not in dispute with both P_i and P_j ; for any player $P_j \in \text{Corr}$, P_i takes 0 as this player's share. P_i can now compute the unique degree- $2t$ polynomial that fits these shares and obtains τ as the evaluation at x_i .

It is easy to verify that if all players follow the protocol, then P_i obtains $\tau = \mu \odot \sigma_i + \nu$ (where σ_i is determined by $[\sigma]_d$ and ν by $\langle \nu \rangle_V^i$). The correctness of the computed tags can be verified by a simple cut-and-choose technique; for the details, we refer to the full version [6].

The two crucial observations regarding the efficiency of $\text{TagComp}_{V,i,d}$ are that the twisted sharings $\lceil \mu^k \rceil_V^i$ can be re-used and thus only need to be prepared *once and for all*, and that the communication complexity of $\text{TagComp}_{V,i,d}$ is *independent* of q , i.e., of the number of shares that are authenticated in one go. As such, the communication complexity of the runs of $\text{TagComp}_{V,i,d}$ is asymptotically negligible; hence, we can authenticate the shares “for free”.

Proposition 1 (Privacy of the keys). *If P_V remains honest and the adversary corrupts at most $t - 1$ players different to P_i , then the adversary learns no*

¹² The actual scheme will be slightly more complicated due to some issue that we ignore right now for simplicity.

information on $\mu = (\mu^1, \dots, \mu^q)$ and ν , beyond $\tau = \sum_k \sigma_i^k \mu^k + \nu$ (for the correct shares σ_i^k , defined by the shares of the uncorrupt players).

By the security of the underlying authentication scheme, this guarantees that if at some later point player P_i lies about his shares, then he will be caught by P_V except with probability $1/|\mathbb{K}|$. Interestingly, if the adversary corrupts t players not including P_i (nor P_V) then he actually learns player P_V 's long-term key μ (that P_V uses to verify P_i 's shares); however, in this case, P_i is guaranteed to remain honest and provide correct shares. So, this does not help the adversary.

Proof. It is sufficient to prove the claim in case of a corrupt dealer P_d and a corrupt player P_i , and thus we may assume that the adversary learns the shares of $\langle \tau \rangle^i = \sum_k [\sigma^k] \lceil \mu^k \rfloor_V^i + \langle \nu \rangle_V^i$, i.e., we may assume that all the shares of σ are 0. We understand $[\sigma^k]$ as the *correct* sharing of some σ^k , determined by the shares of the uncorrupt players. As such, the data structure $\langle \tau \rangle^i = \sum_k [\sigma^k] \lceil \mu^k \rfloor_V^i + \langle \nu \rangle_V^i$, and in particular τ , is well defined, even though the corrupt players may perform additional computations on their shares of μ^k and ν . First note that (by assumption) there are at most $t-1$ corrupt players P_j that hold a (twisted) share of μ^k ; thus, the $\lceil \mu^k \rfloor_V^i$'s give away no information on the μ^k 's to the adversary. However, this is not sufficient to argue privacy, since the adversary also learns all shares of $\langle \tau \rangle^i = \sum_k [\sigma^k] \lceil \mu^k \rfloor_V^i + \langle \nu \rangle_V^i$, which potentially may leak additional information on the μ^k 's and on ν (beyond τ). To argue privacy, consider a twisted sharing $[\delta^1]_V^i$ of an arbitrary $\delta^1 \in \mathbb{K}$, but with the additional property that the shares of all corrupt players are 0. Thus, the adversary cannot distinguish the sharing $\lceil \mu^1 \rfloor_V^i$ from $\lceil \tilde{\mu}^1 \rfloor_V^i = \lceil \mu^1 + \delta^1 \rfloor_V^i = \lceil \mu^1 \rfloor_V^i + \lceil \delta^1 \rfloor_V^i$. Furthermore, the adversary cannot distinguish the sharing $\langle \nu \rangle_V^i$ from $\langle \tilde{\nu} \rangle_V^i = \langle \nu - \sigma^1 \delta^1 \rangle_V^i = \langle \nu \rangle_V^i - [\sigma^1]_d \lceil \delta^1 \rfloor_V^i$. But now, since

$$\begin{aligned} & [\sigma^1]_d \lceil \tilde{\mu}^1 \rfloor_V^i + \sum_{k>1} [\sigma^k]_d \lceil \mu^k \rfloor_V^i + \langle \tilde{\nu} \rangle_V^i \\ &= [\sigma^1]_d \lceil \mu^1 \rfloor + [\sigma^1]_d \lceil \delta^1 \rfloor_V^i + \sum_{k>1} [\sigma^k]_d \lceil \mu^k \rfloor_V^i + \langle \nu \rangle_V^i - [\sigma^1]_d \lceil \delta^1 \rfloor_V^i = \langle \tau \rangle^i \end{aligned}$$

it holds that the adversary has no information on whether μ^1 and ν had been shared (even when given the remaining μ^k 's), or $\tilde{\mu}^1$ and $\tilde{\nu}$. This means that every pair (μ^1, ν) with $\sum_k \sigma_i^k \mu^k + \nu = \tau$ is equally likely for the adversary, and similarly one can argue for the other μ^k 's. \square

Proposition 2 (Privacy of the shares). *If P_d remains honest, then the adversary learns no information on $\sigma = (\sigma^1, \dots, \sigma^q)$.*

The proof of Proposition 2 is similar to that of Proposition 1; for the details, we refer to [6].

2.8 A High Level Sketch of Our Construction

For the preparation phase, every player, acting as dealer P_d , produces many sharings $[s(d)]_d$. Correctness is verified batch-wise by means of a standard cut-and-choose technique. Every list of sharings $[s(1)]_1, \dots, [s(n)]_n$ then gives rise

to $t + 1$ two-level sharings $\llbracket a \rrbracket$ by setting $a = \sum_{d=1}^n s(d)x_j^d$ for $t + 1$ different choices of j . This way, preparing *one* $\llbracket a \rrbracket \in \cup\mathcal{M}$ (and the same for $\llbracket b \rrbracket \in \cup\mathcal{M}$) amounts to preparing *one* $[s(d)]_d$ (up to constant factors), which has linear amortized complexity (meaning: a linear number of elements in \mathbb{F}). This technique is borrowed from [16]. Then, $\llbracket c \rrbracket$, where c is supposed to be $a \cdot b$, is computed by means of the passively-secure multiplication protocol due to [16], which has linear communication complexity. In order to verify the correctness of the c 's, we use the *batch-wise multiplication verification* described in Section 2.7. Using batches of size $N = n^2$, verifying the correctness of N multiplication triples essentially boils down to reconstructing a *constant* number of sharings over the big field \mathbb{K} , which consists of every player sending his share (in \mathbb{K}) to every other player. Per multiplication triple, this then amounts to $O(\kappa)$ bits. Using batches of size $N = n^{2+const}$ reduces this to $O(\kappa/n^{const})$.

It remains to compute the authentication tags. As explained in Section 2.7, for a tag $\tau = \boldsymbol{\mu} \cdot \boldsymbol{\sigma}_i(d) + \nu$ (where $\boldsymbol{\sigma}_i(d)$ consists of many $s_i(d)$'s), this can be done by computing $\langle \tau \rangle^i = \sum_k [\sigma^k(d)]_d [\mu^k]_V^i + \langle \nu \rangle_V^i + \langle o \rangle_d^i$. Since the $\boldsymbol{\mu}$'s (and their twisted shares) are re-used to some extent, and since the $\boldsymbol{\sigma}(d)$'s are already shared, the communication complexity is dominated by communicating the shares $\langle \nu \rangle_V^i$, $\langle o \rangle_d^i$ and $\langle \tau \rangle^i$; this consists of a linear number of elements in \mathbb{K} per (large) block $\boldsymbol{\sigma}_i(d)$ (and per P_V and P_i), making the overall communication complexity per $s(d)$, and thus per multiplication triple, negligible. The correctness of the tags is verified by a standard cut-and-choose technique. The details are worked out in the full paper [6].

Once the data structure as described in Section 2.6 is prepared, we are in good shape. Essentially, the only thing that can cause problems during the input and the computation phase is that corrupt players hand in incorrect shares; but this will be detected (since the shares then do not lie on a degree- t polynomial), and the corrupt players will be found with the help of the authentication tags (on the local base sharings). The details are explained in [6].

2.9 The Full Protocol

Taking care of all the details when putting the above techniques together is rather cumbersome, and the resulting detailed protocol description and its analysis is quite complex and lengthy. Therefore, due to the space limitation, it is given in the full version [6].

3 Conclusion

We showed that MPC with unconditional security against $t < n/2$ corrupt players is possible with amortized asymptotic near-linear communication complexity $O(n \log n)$ bits per multiplication gate for binary circuits. For circuits over a bigger field \mathbb{F} , the $\log n$ term is replaced by $\max\{\log n, \log |\mathbb{F}|\}$. This matches the communication complexity of the best scheme in the much simpler honest-but-curious setting. Room for improvement exists in the terms of the communication complexity that are circuit-size independent, for instance in the $O(n^7\kappa)$ term. Improving this term permits the amortization to step in for smaller circuits.

Acknowledgment. E.B. was supported by the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement number 240258. S.F. is grateful to UCLA for being a regular host; it is thanks to these visits that this project got started and crucial progress was made. R.O. was supported by NSF grants 0830803, 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, B. John Garrick Foundation, OKAWA Foundation, IBM, Lockheed-Martin Corporation and the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1. Beaver, D.: Multiparty Protocols Tolerating Half Faulty Processors. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 560–572. Springer, Heidelberg (1990)
2. Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992)
3. Beerlová-Trubíniová, Z., Hirt, M.: Efficient Multi-party Computation with Dispute Control. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 305–328. Springer, Heidelberg (2006)
4. Beerlová-Trubíniová, Z., Hirt, M.: Perfectly-Secure MPC with Linear Communication Complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008)
5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: 20th Annual ACM Symposium on Theory of Computing (STOC), pp. 1–10 (1988)
6. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure multi-party computation with a dishonest minority (2011), <http://eprint.iacr.org/2011/629>
7. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences 18(2), 143–154 (1979)
8. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: 20th Annual ACM Symposium on Theory of Computing (STOC), pp. 11–19 (1988)
9. Chaum, D., Damgård, I.B., van de Graaf, J.: Multiparty Computations Ensuring Privacy of Each Party’s Input and Correctness of the Result. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 87–119. Springer, Heidelberg (1988)
10. Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient Multiparty Computations Secure against an Adaptive Adversary. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 311–326. Springer, Heidelberg (1999)
11. Cramer, R., Damgård, I., Maurer, U.: General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
12. Cramer, R., Damgård, I., Pastro, V.: On the Amortized Complexity of Zero Knowledge Protocols for Multiplicative Relations. In: Smith, A. (ed.) ICITS 2012. LNCS, vol. 7412, pp. 62–79. Springer, Heidelberg (2012)

13. Damgård, I., Ishai, Y.: Scalable Secure Multiparty Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006)
14. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly Secure Multiparty Computation and the Computational Overhead of Cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (2010)
15. Damgård, I., Ishai, Y., Krøigaard, M., Nielsen, J.B., Smith, A.: Scalable Multiparty Computation with Nearly Optimal Work and Resilience. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 241–261. Springer, Heidelberg (2008)
16. Damgård, I., Nielsen, J.B.: Scalable and Unconditionally Secure Multiparty Computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007)
17. Goldwasser, S., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: 19th Annual ACM Symposium on Theory of Computing (STOC), pp. 218–229 (1987)
18. Hirt, M., Maurer, U.: Robustness for Free in Unconditional Multi-party Computation. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 101–118. Springer, Heidelberg (2001)
19. Hirt, M., Nielsen, J.B.: Upper Bounds on the Communication Complexity of Optimally Resilient Cryptographic Multiparty Computation. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 79–99. Springer, Heidelberg (2005)
20. Hirt, M., Nielsen, J.B.: Robust Multiparty Computation with Linear Communication Complexity. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 463–482. Springer, Heidelberg (2006)
21. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: 21st Annual ACM Symposium on Theory of Computing (STOC), pp. 73–85 (1989)
22. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
23. Yao, A.: Protocols for secure computations. In: 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 160–164 (1982)

A New Approach to Practical Active-Secure Two-Party Computation*

Jesper Buus Nielsen^{1,***,***}, Peter Sebastian Nordholt^{1,**}, Claudio Orlandi^{2,†},
and Sai Sheshank Burra³

¹ Aarhus University

² Bar-Ilan University

³ Indian Institute of Technology Guwahati

Abstract. We propose a new approach to practical two-party computation secure against an active adversary. All prior practical protocols were based on Yao’s garbled circuits. We use an OT-based approach and get efficiency via OT extension in the random oracle model. To get a practical protocol we introduce a number of novel techniques for relating the outputs and inputs of OTs in a larger construction.

We also report on an implementation of this approach, that shows that our protocol is more efficient than any previous one: For big enough circuits, we can evaluate more than 20000 Boolean gates per second. As an example, evaluating one oblivious AES encryption (~ 34000 gates) takes 64 seconds, but when repeating the task 27 times it only takes less than 3 seconds per instance.

1 Introduction

Secure two-party computation (2PC), introduced by Yao [32], allows two parties to jointly compute any function of their inputs in such a way that 1) the output of the computation is correct and 2) the inputs are kept private. Yao’s protocol is secure only if the participants are *semi-honest* (they follow the protocol but try to learn more than they should by looking at their transcript of the protocol). A more realistic security definition considers *malicious adversaries*, that can arbitrarily deviate from the protocol.

A large number of approaches to 2PC have been proposed, falling into three main types, those based on Yao’s garbled circuit techniques, those based on some form of homomorphic encryption and those based on oblivious transfer. Recently a number of efforts to implement 2PC in practice have been reported on; In sharp contrast to the theory, almost all of these are based on Yao’s garbled circuit technique. A main advantage of Yao’s garbled circuits is that it is primarily based on symmetric primitives: It uses one OT per input bit, but then uses only a few calls to, e.g., a hash function per gate in the circuit to be evaluated. The other approaches are heavy on public-key primitives which are typically orders of magnitude slower than symmetric primitives.

* A full version of this article can be found at <http://eprint.iacr.org/2011/091>.

** Partially supported by the Danish National Research Foundation and the National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation.

*** Partially supported by a Sapere Aude grant from the Danish Council for Independent Research.

† Supported by the European Research Council as part of the ERC project LAST.

However, in 2003 Ishai *et al.* introduced the idea of extending OTs *efficiently* [18]—their protocol allows to turn κ seed OTs based on public-key crypto into any polynomial $\ell = \text{poly}(\kappa)$ number of OTs using only $O(\ell)$ invocations of a cryptographic hash function. For big enough ℓ the cost of the κ seed OTs is amortized away and OT extension essentially turns OT into a symmetric primitive in terms of its computational complexity. Since the basic approach of basing 2PC on OT in [14] is efficient in terms of consumption of OTs and communication, this gives the hope that OT-based 2PC too could be practical. This paper reports on the first implementation made to investigate the practicality of OT-based 2PC.

Our starting point is the efficient passive-secure OT extension protocol of [18] and passive-secure 2PC of [14]. In order to get active security and preserve the high practical efficiency of these protocols we chose to develop substantially different techniques, differentiating from other works that were only interested in *asymptotic* efficiency [15, 20, 29]. We report a number of contributions to the theory and practice of 2PC:

1. We introduce a new technical idea to the area of extending OTs efficiently, which allows to dramatically improve the practical efficiency of active-secure OT extension. Our protocol has the same asymptotic complexity as the previously best protocol in [15], but it is only a small factor slower than the passive-secure protocol in [18].
2. We give the first implementation of the idea of extending OTs efficiently with active security. The protocol generates 500,000 OTs per second, showing that implementations needing a large number of actively secure OTs can be practical.
3. We introduce new technical ideas which allow to relate the outputs and inputs of OTs in a larger construction, via the use of information theoretic tags. This can be seen as a new flavor of committed OT that only requires symmetric cryptography. In combination with our first contribution, our protocol shows how to efficiently extend committed OT. Our protocols assume the existence of OT and are secure in the random oracle model.
4. We give the first implementation of practical 2PC not based on Yao’s garbled circuit technique. Introducing a new practical technique is a significant contribution to the field in itself. In addition, our protocol shows favorable timings compared to the Yao-based implementations.

1.1 Comparison with Related Work

The question on the *asymptotic* computational overhead of cryptography was (essentially) settled in [19]. On the other hand, there is a growing interest in understanding the *practical* overhead of secure computation, and several works have perfected and implemented protocols based on Yao’s garbled circuits [1, 3, 16, 17, 23, 25–28, 30, 31], protocols based on homomorphic encryption [4, 10, 21, 22] and protocols based on OT [6, 20, 24].

A brief comparison of the time needed for oblivious AES evaluation for the best known implementations are shown in Table 1.¹ The protocols in rows (a-b) are for 3 and 4 parties respectively, and are secure against at most one corrupted party. One of the

¹ Oblivious AES has become one of the most common circuits to use for benchmarking generic MPC protocols, due to its reasonable size (about 30000 gates) and its relevance as a building block for constructing specific purpose protocols, like private set intersection [11].

Table 1. Brief comparison with other implementations

		Security	Model	Rounds	Time
(a)	DK [9] (3 parties)	Passive	SM	$O(d)$	1.5s
(b)	DK [9] (4 parties)	Active	SM	$O(d)$	4.5s
(c)	sS [1]	Active	SM	$O(1)$	192s
(d)	HEKM [17]	Passive	ROM	$O(1)$	0.2s
(e)	IPS-LOP [20,24]	Active	SM	$O(d)$	79s
(f)	This (single)	Active	ROM	$O(d)$	64s
(g)	This (27, amortized)	Active	ROM	$O(d)$	2.5s

indicates the round complexity of the protocols, d being the depth of the circuit while the column *Model* indicates whether the protocol was proven secure in the standard model (SM) or the random oracle model (ROM).

The significance of this work is shown in row (g). The reason for the dramatic drop between row (f) and (g) is that in (f), when we only encrypt one block, our implementation preprocesses for many more gates than is needed, for ease of implementation. In (g) we encrypt 27 blocks, which is the minimum value which eats up all the pre-processed values. We consider these results positive: our implementation is as fast or faster than any other 2PC protocol, even when encrypting only one block. And more importantly, when running at full capacity, the price to pay for active security is about a factor 10 against the passive-secure protocol in (d). We stress that this is only a limited comparison, as the different experiments were run on different hardware and network setups: when several options were available, we selected the best time reported by the other implementations. See Sect. 6 for more timings and details of our implementation.

1.2 Overview of Our Approach

We start from a classic textbook protocol for 2PC [13, Sec. 7.3]. In this protocol, Alice holds secret shares x_A, y_A and Bob holds secret shares x_B, y_B of some bits x, y s.t. $x_A \oplus x_B = x$ and $y_A \oplus y_B = y$. Alice and Bob want to compute secret shares of $z = g(x, y)$ where g is some Boolean gate, for instance the AND gate: Alice and Bob need to compute a random sharing z_A, z_B of $z = xy = x_A y_A \oplus x_A y_B \oplus x_B y_A \oplus x_B y_B$. The parties can compute the AND of their local shares ($x_A y_A$ and $x_B y_B$), while they can use oblivious transfer (OT) to compute the cross products ($x_A y_B$ and $x_B y_A$). Now the parties can iterate for the next layer of the circuit, up to the end where they will reconstruct the output values by revealing their shares.

This protocol is secure against a semi-honest adversary: assuming the OT protocol to be secure, Alice and Bob learn nothing about the intermediate values of the computation. It is easy to see that if a large circuit is evaluated, then the protocol is not secure against a malicious adversary: any of the two parties could replace values on any of the internal wires, leading to a possibly incorrect output and/or leakage of information.

To cope with this, we put MACs on all bits. The starting point of our protocol is *oblivious authentication* of bits. One party, the *key holder*, holds a uniformly random

goals of the work in row (c) is how to efficiently support different outputs for different parties: in our OT based protocol this feature comes for free. The time in row (e) is an estimate made by [24] on the running time of their optimized version of the OT-based protocol in [20]. The column *Round*

global key $\Delta \in \{0, 1\}^\kappa$. The other party, the *MAC holder*, holds some secret bits (x, y) , say. For each such bit the key holder holds a corresponding uniformly random *local key* ($K_x, K_y \in \{0, 1\}^\kappa$) and the MAC holder holds the corresponding *MAC* ($M_x = K_x \oplus x\Delta, M_y = K_y \oplus y\Delta$). The key holder does not know the bits and the MAC holder does not know the keys. Note that $M_x \oplus M_y = (K_x \oplus K_y) \oplus (x \oplus y)\Delta$. So, the MAC holder can locally compute a MAC on $x \oplus y$ under the key $K_x \oplus K_y$ which is non-interactively computable by the key holder. This homomorphic property comes from fixing Δ and we exploit it throughout our constructions. From a bottom-up look, our protocol is constructed as follows (see Fig. 1 for the main structure):

Bit Authentication: We first implement oblivious authentication of bits (aBit). As detailed in Sect. 4, to construct authenticated bits we start by extending a few (say $\kappa = 640$) seed $\binom{2}{1}$ -OTs into many (say $\ell = 2^{20}$) OTs, using OT extension. Then, if A wants to get a bit x authenticated, she can input it as the choice bit in an OT, while B can input $(K_x, K_x \oplus \Delta)$, playing the sender in the OT. Now A receives $M_x = K_x \oplus x\Delta$. It should, of course, be ensured that even a corrupted B uses the same value Δ in all OTs. I.e., it should hold for all produced OTs that the XORs of the offered message pairs are constant—this constant value is then taken to be Δ . It turns out, however, that when using the highly efficient *passive-secure* OT extender in [18] and starting from seed OTs where the XORs of message pairs are constant, one also produces OTs where the XORs of message pairs are constant, and we note that for this use the protocol in [18] happens to be *active-secure*! Using cut-and-choose we ensure that most of the XORs of message pairs offered in the seed OTs are constant, and with a new and inexpensive trick we offer privacy and correctness even if few of these XORs have different values. This cut-and-choose technique uses one call to a box EQ for checking equality.

Authenticated local AND: From aBits we then construct *authenticated local ANDs* (aAND), where the MAC holder locally holds random authenticated bits a, b, c with $c = ab$. To create authenticated local ANDs, we let one party compute $c = ab$ for random a and b and get authentications on a, b, c (when creating aANDs, we assume the aBits are already available). The challenge is to ensure that $c = ab$. We construct an efficient proof for this fact, again using the box EQ once. This proof might, however, leak the bit a with small but noticeable probability. We correct this using a combiner.

Authenticated OT: From aBits we also construct *authenticated OTs* (aOT), which are normal $\binom{2}{1}$ -OTs of bits, but where all input bits and output bits are obliviously authenticated. This is done by letting the two parties generate aBits representing the sender messages x_0, x_1 and the receiver choice bit c . To produce the receiver's output, first a random aBit is sampled. Then this bit is “corrected” in order to be

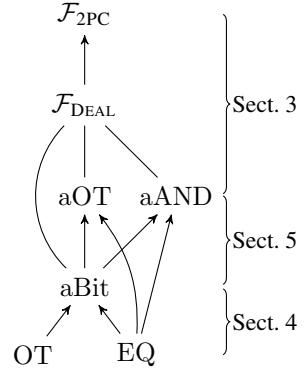


Fig. 1. Paper outline. This order of presentation is chosen to allow the best progression in introduction of our new techniques.

consistent with the run of an OT protocol with input messages x_0, x_1 and choice bit c . This correction might, however, leak the bit c with small but noticeable probability. We correct this using an OT combiner. One call to the box EQ is used.

2PC: Given two aANDs and two aOTs one can evaluate in a very efficient way any Boolean gate: only 4 bits per gate are communicated, as the MACs can be checked in an amortized manner.

That efficient 2PC is possible given enough aBits, aANDs and aOTs is no surprise. In some sense, it is the standard way to base passive-secure 2PC on passive-secure OT enhanced with a particular flavor of committed OT (as in [8, 12]). What is new is that we managed to find a particular committed OT-like primitive which allows both a very efficient generation and a very efficient use: while previous results based on committed OT require hundreds of *exponentiations* per gate, our cost per gate is in the order of hundreds of *hash functions*. To the best of our knowledge, we present the first practical approach to extending a few seed OTs into a large number of committed OT-like primitives. Of more specific technical contributions, the main is that we manage to do all the proofs efficiently, thanks also to the preprocessing nature of our protocol: Creating aBits, we get active security paying only a constant overhead over the passive-secure protocol in [18]. In the generation of aANDs and aOTs, we replace cut-and-choose with efficient, slightly leaky proofs and then use a combiner to get rid of the leakage: When we preprocess for ℓ gates and combine B leaky objects to get each potentially unleaky object, the probability of leaking is $(2\ell)^{-B} = 2^{-\log_2(\ell)(B-1)}$. As an example, if we preprocess for 2^{20} gates with an overhead of $B = 6$, then we get leakage probability 2^{-100} .

As a corollary to being able to generate any $\ell = \text{poly}(\kappa)$ active-secure aBits from $O(\kappa)$ seed OTs and $O(\ell)$ calls to a hash-function, we get that we can generate any $\ell = \text{poly}(\kappa)$ active-secure $\binom{2}{1}$ -OTs of κ -bit strings from $O(\kappa)$ seed OTs and $O(\ell)$ calls to a hash-function, matching the asymptotic complexity of [15] while dramatically reducing their hidden constants.

2 Preliminaries and Notation

We use κ (and sometimes ψ) to denote the security parameter. We require that a poly-time adversary break the protocol with probability at most $\text{poly}(\kappa)2^{-\kappa}$. For a bit-string $S \in \{0, 1\}^*$ we define $0S \stackrel{\text{def}}{=} 0^{|S|}$ and $1S \stackrel{\text{def}}{=} S$. For a finite set S we use $s \in_R S$ to denote that s is chosen uniformly at random in S . For a finite distribution D we use $x \leftarrow D$ to denote that x is sampled according to D .

The UC Framework. We prove our results static, active-secure in the UC framework [5], and we assume the reader to be familiar with it. We will idiosyncratically use the word *box* instead of the usual term *ideal functionality*. To simplify the statements of our results we use the following terminology:

Definition 1. *We say that a box A is reducible to a box B if there exist an actively secure implementation π of A which uses only one call to B. We say that A is locally reducible to B if the parties of π do not communicate (except through the one call to B).*

We say that A is linear reducible to B if the computing time of all parties of π is linear in their inputs and outputs. We use \equiv to denote reducibility in both directions.

It is easy to see that if A is (linear, locally) reducible to B and B is (linear, locally) reducible to C, then A is (linear, locally) reducible to C.

Hash Functions. We use a hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$, which we model as a random oracle (RO). We sometimes use H to mask a message, as in $H(x) \oplus M$. If $|M| \neq \kappa$, this denotes $\text{prg}(H(x)) \oplus M$, where prg is a pseudo-random generator $\text{prg} : \{0, 1\}^\kappa \rightarrow \{0, 1\}^{|M|}$. We also use a collision-resistant hash function $G : \{0, 1\}^{2\kappa} \rightarrow \{0, 1\}^\kappa$.

As other 2PC protocols whose focus is efficiency [17, 23], we are content with a proof in the random oracle model. What is the exact assumption on the hash function that we need for our protocol to be secure, as well as whether this can be implemented under standard cryptographic assumption is an interesting theoretical question, see [2, 7].

Oblivious Transfer. We use a box $\text{OT}(\tau, \ell)$ which can be used to perform $\tau \binom{2}{1}$ -oblivious transfers of strings of bit-length ℓ . In each of the τ OTs the sender S has two inputs $x_0, x_1 \in \{0, 1\}^\ell$, called the *messages*, and the receiver R has an input $c \in \{0, 1\}$, called the *choice bit*. The output to R is $x_c = c(x_0 \oplus x_1) \oplus x_0$. No party learns any other information.

Equality Check. We use a box $\text{EQ}(\ell)$ which allows two parties to check that two strings of length ℓ are equal. If they are different the box leaks both strings to the adversary, which makes secure implementation easier. We define and use this box to simplify the exposition of our protocol. In practice we implement the box using the commit-and-open approach. Hashing a string (together with some randomness) is a secure implementation of commitment in the random oracle model. See the full version for more details.

Leakage Functions. We use a concept of a *leakage function on τ bits*, which is a class \mathcal{L} of distributions, where each $L \in \mathcal{L}$ is a distribution on $(S, c) \in 2^{\{1, \dots, \tau\}} \times \{0, 1\}$, where, as we will see later, $c = 0$ is interpreted as a failure to create leakage and $c = 1$ is interpreted as leakage of the bits indexed by $i \in S$. We say that L is κ -secure if the expected value of $\tau - c|S|$ is at least κ and we say that \mathcal{L} is κ -secure if all $L \in \mathcal{L}$ are κ -secure. See the full version for a more detailed definition.

3 The Two-Party Computation Protocol

We want to implement the box $\mathcal{F}_{2\text{PC}}$ for Boolean two-party secure computation as described in Fig. 4. We will implement this box in the $\mathcal{F}_{\text{DEAL}}$ -hybrid model of Fig. 5. This box provides the parties with aBits, aANDs and aOTs, and models the preprocessing phase of our protocol. In Fig. 3 we introduce notation for working with authenticated bits. The protocol implementing $\mathcal{F}_{2\text{PC}}$ in the dealer model is described in Fig. 6. The dealer offers random authenticated bits (to A or B), random authenticated local AND triples and random authenticated OTs.

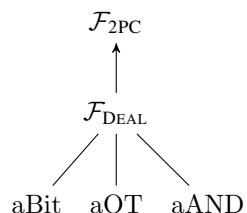


Fig. 2. Sect. 3 outline

Global Key: We call $\Delta_A, \Delta_B \in \{0, 1\}^\kappa$ the two *global keys*, held by B and A respectively.

Authenticated Bit: We write $[x]_A$ to represent an *authenticated secret bit* held by A. Here B knows a key $K_x \in \{0, 1\}^\kappa$ and A knows a bit x and a MAC $M_x = K_x \oplus x\Delta_A \in \{0, 1\}^\kappa$. Let $[x]_A \stackrel{\text{def}}{=} (x, M_x, K_x)$.^a

If $[x]_A = (x, M_x, K_x)$ and $[y]_A = (y, M_y, K_y)$ we write $[z]_A = [x]_A \oplus [y]_A$ to indicate $[z]_A = (z, M_z, K_z) \stackrel{\text{def}}{=} (x \oplus y, M_x \oplus M_y, K_x \oplus K_y)$. Note that no communication is required to compute $[z]_A$ from $[x]_A$ and $[y]_A$.

It is possible to authenticate a constant bit (a value known both to A and B) $b \in \{0, 1\}$ as follows: A sets $M_b = 0^\kappa$, B sets $K_b = b\Delta_A$, now $[b]_A \stackrel{\text{def}}{=} (b, M_b, K_b)$. For a constant b we let $[x]_A \oplus b \stackrel{\text{def}}{=} [x]_A \oplus [b]_A$, and we let $b[x]_A$ be equal to $[0]_A$ if $b = 0$ and $[x]_A$ if $b = 1$.

We say that A *reveals* $[x]_A$ by sending (x, M_x) to B who aborts if $M_x \neq K_x \oplus x\Delta_A$. Alternatively we say that A *announces* x by sending x to B without a MAC.

Authenticated bits belonging to B are written as $[y]_B$ and are defined symmetrically, changing side of all the values and using the global value Δ_B instead of Δ_A .

Authenticated Share: We write $[x]$ to represent the situation where A and B hold $[x_A]_A, [x_B]_B$ and $x = x_A \oplus x_B$, and we write $[x] = ([x_A]_A, [x_B]_B)$ or $[x] = [x_A|x_B]$. If $[x] = [x_A|x_B]$ and $[y] = [y_A|y_B]$ we write $[z] = [x] \oplus [y]$ to indicate $[z] = ([z_A]_A, [z_B]_B) = ([x_A]_A \oplus [y_A]_A, [x_B]_B \oplus [y_B]_B)$. Note that no communication is required to compute $[z]$ from $[x]$ and $[y]$.

It is possible to create an authenticated share of a constant $b \in \{0, 1\}$ as follows: A and B create $[b] = [b|0]$. For a constant value $b \in \{0, 1\}$, we define $b[x]$ to be equal to $[0]$ if $b = 0$ and $[x]$ if $b = 1$.

When an authenticated share is *revealed*, the parties reveal to each other their authenticated bits and abort if the MACs are not correct.

^a Since Δ_A is a global value we will not always write it explicitly. Note that in $x\Delta_A$, x represents a *value*, 0 or 1, and that in $[x]_A$, K_x and M_x it represents a *variable name*. I.e., there is only one key (MAC) per authenticated bit, and for the bit named x , the key (MAC) is named K_x (M_x). If $x = 0$, then $M_x = K_x$. If $x = 1$, then $M_x = K_x \oplus \Delta_A$.

Fig. 3. Notation for authenticated and shared bits

Those are all the ingredients that we need to build the 2PC protocol. Note that the dealer offers randomized versions of all commands: this is not a problem as the “standard” version of the commands (the one where the parties can specify their input bits instead of getting them at random from the box) are linearly reducible to the randomized version, as can be easily deduced from the protocol description. The following result is proven in the full version.

Theorem 1. *The protocol in Fig. 6 securely implements the box \mathcal{F}_{2PC} in the \mathcal{F}_{DEAL} -hybrid model with security parameter κ .*

Why the global key queries? The \mathcal{F}_{DEAL} box (Fig. 5) allows the adversary to guess the value of the global key, and it informs it if its guess is correct. This is needed for technical reasons: When \mathcal{F}_{DEAL} is proven UC secure, the environment has access to either \mathcal{F}_{DEAL} or the protocol implementing \mathcal{F}_{DEAL} . In both cases the environment learns the global keys Δ_A and Δ_B . In particular, the environment learns Δ_A even if B is honest. This requires

Rand: On input (rand, vid) from A and B, with vid a fresh identifier, the box picks $r \in_{\mathbb{R}} \{0, 1\}$ and stores (vid, r) .

Input: On input $(\text{input}, P, vid, x)$ from $P \in \{\text{A}, \text{B}\}$ and $(\text{input}, P, vid, ?)$ from the other party, with vid a fresh identifier, the box stores (vid, x) .

XOR: On command $(\text{xor}, vid_1, vid_2, vid_3)$ from both parties (if vid_1, vid_2 are defined and vid_3 is fresh), the box retrieves $(vid_1, x), (vid_2, y)$ and stores $(vid_3, x \oplus y)$.

AND: As **XOR**, but store $(vid_3, x \cdot y)$.

Output: On input (output, P, vid) from both parties, with $P \in \{\text{A}, \text{B}\}$ (and vid defined), the box retrieves (vid, x) and outputs it to P.

At each command the box leaks to the environment which command is being executed (keeping the value x in **Input** secret), and delivers messages only when the environment says so.

Fig. 4. The box \mathcal{F}_{2PC} for Boolean Two-party Computation

Initialize: On input (init) from A and (init) from B, the box samples $\Delta_A, \Delta_B \in \{0, 1\}^\kappa$, stores them and outputs Δ_B to A and Δ_A to B. If A (resp. B) is corrupted, she gets to choose Δ_B (resp. Δ_A).

Authenticated Bit (A): On input (aBIT, A) from A and B, the box samples a random $[x]_A = (x, M_x, K_x)$ with $M_x = K_x \oplus x\Delta_A$ and outputs it (x, M_x to A and K_x to B). If B is corrupted he gets to choose K_x . If A is corrupted she gets to choose (x, M_x) , and the box sets $K_x = M_x \oplus x\Delta_A$.

Authenticated Bit (B): On input (aBIT, B) from A and B, the box samples a random $[x]_B = (x, M_x, K_x)$ with $M_x = K_x \oplus x\Delta_B$ and outputs it (x, M_x to B and K_x to A). As in **Authenticated Bit (A)**, corrupted parties can choose their own randomness.

Authenticated local AND (A): On input (aAND, A) from A and B, the box samples random $[x]_A, [y]_A$ and $[z]_A$ with $z = xy$ and outputs them. As in **Authenticated Bit (A)**, corrupted parties can choose their own randomness.

Authenticated local AND (B): Defined symmetrically.

Authenticated OT (A-B): On input $(\text{aOT}, \text{A}, \text{B})$ from A and B, the box samples random $[x_0]_A, [x_1]_A, [c]_B$ and $[z]_B$ with $z = x_c = c(x_0 \oplus x_1) \oplus x_0$ and outputs them. As in **Authenticated Bit**, corrupted parties can choose their own randomness.

Authenticated OT (B-A): Defined symmetrically.^a

Global Key Queries: The adversary can at any point input (A, Δ) and be told whether $\Delta = \Delta_B$. And it can at any point input (B, Δ) and be told whether $\Delta = \Delta_A$.

^a The dealer offers aOTs in both directions. Notice that the dealer could offer aOT only in one direction and the parties could then “turn” them: as regular OT, aOT is symmetric as well.

Fig. 5. The box \mathcal{F}_{DEAL} for dealing preprocessed values

us to prove the sub-protocol for \mathcal{F}_{DEAL} secure to an adversary knowing Δ_A even if B is honest: to be able to do this, the simulator needs to recognize Δ_A if it sees it—hence the global key queries. Note, however, that in the context where we use \mathcal{F}_{DEAL} (Fig. 6), the environment does *not* learn the global key Δ_A when B is honest: A corrupted A only sees MACs on one bit using the same local key, so all MACs are uniformly random in the view of a corrupted A, and B never makes the local keys public.

Initialize: When activated the first time, A and B activate $\mathcal{F}_{\text{DEAL}}$ and receive Δ_B and Δ_A respectively.

Rand: A and B ask $\mathcal{F}_{\text{DEAL}}$ for random authenticated bits $[r_A]_A, [r_B]_B$ and stores $[r] = [r_A|r_B]$ under vid .

Input: If $P = A$, then A asks $\mathcal{F}_{\text{DEAL}}$ for an authenticated bit $[x_A]_A$ and announces (i.e., no MAC is sent together with the bit) $x_B = x \oplus x_A$, and the parties build $[x_B]_B$ and define $[x] = [x_A|x_B]$. The protocol is symmetric for B.

XOR: A and B retrieve $[x], [y]$ stored under vid_1, vid_2 and store $[z] = [x] \oplus [y]$ under vid_3 . For brevity we drop explicit mentioning of variable identifiers below.

AND: A and B retrieve $[x], [y]$ and compute $[z] = [xy]$ as follows:

1. The parties ask $\mathcal{F}_{\text{DEAL}}$ for a random AND triplet $[u]_A, [v]_A, [w]_A$ with $w = uv$. A reveals $[f]_A = [u]_A \oplus [x_A]_A$ and $[g]_A = [v]_A \oplus [y_A]_A$. The parties compute $[x_{AYA}]_A = f[y_A]_A \oplus g[x_A]_A \oplus [w]_A \oplus fg$.
2. Symmetrically the parties compute $[x_{BYB}]_B$.
3. The parties ask $\mathcal{F}_{\text{DEAL}}$ for a random authenticated OT $[u_0]_A, [u_1]_A, [c]_B, [w]_B$ with $w = u_c$. They also ask for an authenticated bit $[r_A]_A$. Now B reveals $[d]_B = [c]_B \oplus [y_B]_B$. A reveals $[f]_A = [u_0]_A \oplus [u_1]_A \oplus [x_A]_A$ and $[g]_A = [r_A]_A \oplus [u_0]_A \oplus d[x_A]_A$. Compute $[s_B]_B = [w]_B \oplus f[c]_B \oplus g$. Note that at this point $[s_B]_B = [r_A \oplus x_{AYB}]_B$.
4. Symmetrically the parties compute $[s_A]_A = [r_B \oplus x_{BYA}]_A$.

A and B compute $[z_A]_A = [r_A]_A \oplus [s_A]_A \oplus [x_{AYA}]_A$ and $[z_B]_B = [r_B]_B \oplus [s_B]_B \oplus [x_{BYB}]_B$ and let $[z] = [z_A|z_B]$.

Output: The parties retrieve $[x] = [x_A|x_B]$. If A is to learn x , B reveals x_B . If B is to learn x , A reveals x_A .

Fig. 6. Protocol for \mathcal{F}_{2PC} in the $\mathcal{F}_{\text{DEAL}}$ -hybrid model

Amortized MAC checks. In the protocol of Fig. 6, there is no need to send MACs and check them every time we do a “reveal”. In fact, it is straightforward to verify that before an **Output** command is executed, the protocol is perfectly secure even if the MACs are not checked. Notice then that a key holder checks a MAC M_x on a bit x by computing $M'_x = K_x \oplus x\Delta$ and comparing M'_x to the M_x which was sent along with x . These equality checks can be deferred and amortized. Initially the MAC holder, e.g. A, sets $N = 0^\kappa$ and the key holder, e.g. B, sets $N' = 0^\kappa$. As long as no **Output** command is executed, when A reveals x she updates $N \leftarrow G(N, H(M_x))$ for the MAC M_x she should have sent along with x , and B updates $N' \leftarrow G(N', H(M'_x))$; Here G is a collision resistant hash function. Before executing an **Output**, A sends N to B who aborts if $N \neq N'$. Security of this check is easily proved in the random oracle model. The optimization brings the communication complexity of the protocol down from $O(\kappa|C|)$ to $O(|C| + ok)$, where o is the number of rounds in which outputs are opened. For a circuit of depth $O(|C|/\kappa)$, the communication is $O(|C|)$.

Implementing $\mathcal{F}_{\text{DEAL}}$. In the following sections we show how to implement $\mathcal{F}_{\text{DEAL}}$. In Sect. 4 we implement just the part with the commands **Authenticated Bits**. In Sect. 5 we show how to extend with the **Authenticated OT** command, by showing how to implement many aOTs from many aBits. In the full version we show how to further extend with the **Authenticated local AND** command. We describe the extensions separately,

but since they both maintain the value of the global keys, they will produce aANDs and aOTs with the same keys as the aBits used, giving an implementation of $\mathcal{F}_{\text{DEAL}}$.

4 Bit Authentication

In this section we show how to efficiently implement (oblivious) bit authentication, i.e., we want to be in a situation where A knows some bits x_1, \dots, x_ℓ together with MACs M_1, \dots, M_ℓ , while B holds a global key Δ_A and local keys K_1, \dots, K_ℓ s.t. $M_i = K_i \oplus x_i \Delta_A$, as described in $\mathcal{F}_{\text{DEAL}}$ (Fig. 5). Given the complete symmetry of $\mathcal{F}_{\text{DEAL}}$, we only describe the case where A is MAC holder.

If the parties were honest, we could do the following: A and B run an OT where B inputs the two messages $(K_i, K_i \oplus \Delta_A)$ and A inputs choice bit x_i , to receive $M_i = K_i \oplus x_i \Delta_A$. However, if B is dishonest he might not use the same Δ_A in all OTs. The main ideas that make the protocol secure against cheating parties are the following:

1. For reasons that will be apparent later, we will actually start in the opposite direction and let B receive some authenticated bits y_i using an OT, where A is supposed to always use the same global key Γ_B . Thus an honest A inputs $(L_i, L_i \oplus \Gamma_B)$ in the OTs and B receives $N_i = L_i \oplus y_i \Gamma_B$. To check that A is playing honest in most OTs, the authenticated bits are arranged into pairs and a check is performed, which restricts A to cheat in at most a few OTs.
2. We then notice that what A gains by using different Γ_B 's in a few OTs is no more than learning a few of B's bits y_i . We call this a leaky aBit, or LaBit.
3. We show how to turn this situation into an equivalent one where A (not B) receives authenticated random bits x_i (none of which leaks to B) under a “slightly insecure” global key Γ_A . The insecurity comes from the fact that the leakage of the y_i 's turns into the leakage of a few bits of the global key Γ_A towards A. We call this an aBit with weak global key, or WaBit.
4. Using privacy amplification, we amplify the previous setting to a new one where A receives authenticated bits under a (shorter) fully secure global key Δ_A , where no bits of Δ_A are known to A, finally implementing the aBit command of the dealer box.

We will proceed in reverse order and start with step 4 in the previous description: we will start with showing how we can turn authenticated bits under an “insecure” global key Γ_A into authenticated bits under a “secure” (but shorter) global key Δ_A .

4.1 Bit Authentication with Weak Global Key (WaBit)

We will first define the box providing bit authentication, but where some of the bits of the global key might leak. We call this box WaBit (bit authentication with weak global

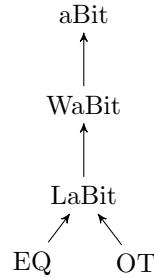


Fig. 7. Sect. 4 outline

Honest Parties:

1. The box samples $\Gamma_A \in_R \{0, 1\}^\tau$ and outputs it to B.
2. The box samples and outputs $[x_1]_A, \dots, [x_\ell]_A$. Each $[x_i]_A = (x_i, M'_i, K'_i) \in \{0, 1\}^{1+2\tau}$ s.t. $M'_i = K'_i \oplus x_i \Gamma_A$.

Corrupted Parties:

1. If A is corrupted, then A may choose a leakage function $L \in \mathcal{L}$. Then the box samples $(S, c) \leftarrow L$. If $c = 0$ the box outputs fail to B and terminates. If $c = 1$, the box outputs $\{(i, (\Gamma_A)_i)\}_{i \in S}$ to A.
2. If A is corrupted, then A chooses the x_i and the M'_i and then $K'_i = M'_i \oplus x_i \Gamma_A$.
3. If B is corrupted, then B chooses Γ_A and the K'_i .

Global Key Queries: The adversary can input Γ and will be told if $\Gamma = \Gamma_A$.

Fig. 8. The box $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$ for Bit Authentication with Weak Global Key

1. The parties invoke $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$ with $\tau = \frac{22}{3}\psi$. The output to A is $((M'_1, x_1), \dots, (M'_\ell, x_\ell))$. The output to B is $(\Gamma_A, K'_1, \dots, K'_\ell)$.
2. B samples $\mathbf{A} \in_R \{0, 1\}^{\psi \times \tau}$, a random binary matrix with ψ rows and τ columns, and sends \mathbf{A} to A.
3. A computes $M_i = \mathbf{A}M'_i \in \{0, 1\}^\psi$ and outputs $((M_1, x_1), \dots, (M_\ell, x_\ell))$.
4. B computes $\Delta_A = \mathbf{A}\Gamma_A$ and $K_i = \mathbf{A}K'_i$ and outputs $(\Delta_A, K_1, \dots, K_\ell)$.

Fig. 9. Sub-protocol for reducing $\text{aBit}(\ell, \psi)$ to $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$

Honest Parties:

1. The box samples $\Gamma_B \in_R \{0, 1\}^\ell$ and outputs it to A.
2. The box samples and outputs $[y_1]_B, \dots, [y_\tau]_B$. Each $[y_i]_B = (y_i, N_i, L_i) \in \{0, 1\}^{1+2\ell}$ s.t. $N_i = L_i \oplus y_i \Gamma_B$.

Corrupted Parties:

1. If A is corrupted, then A may input a leakage function $L \in \mathcal{L}$. Then the box samples $(S, c) \leftarrow L$. If $c = 0$ the box outputs fail to B and terminates. If $c = 1$, the box outputs $\{(i, y_i)\}_{i \in S}$ to A.
2. Corrupted parties get to specify their outputs as in Fig. 8.

Choice Bit Queries: The adversary can input Δ and will be told if $\Delta = (y_1, \dots, y_\tau)$.

Fig. 10. The box $\text{LaBit}^{\mathcal{L}}(\tau, \ell)$ for Bit Authentication with Leaking Bits

key) and we formally describe it in Fig. 8. The box $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$ outputs ℓ bits with keys of length τ . The box is also parameterized by a class \mathcal{L} of leakage functions on τ bits. The box $\text{aBit}(\ell, \psi)$ is the box $\text{WaBit}^{\mathcal{L}}(\ell, \psi)$ where \mathcal{L} is the class of leakage functions that never leak.

In Fig. 9 we describe a protocol which takes a box WaBit , where one quarter of the bits of the global key might leak, and amplifies it to a box aBit where the global key is perfectly secret. The protocol is described for general \mathcal{L} and it is parameterized by a desired security level ψ . The proof of the following theorem can be found in the full version.

1. A and B invoke $\text{LaBit}^{\mathcal{L}}(\tau, \ell)$. B learns $((N_1, y_1), \dots, (N_\tau, y_\tau))$ and A learns $(\Gamma_B, L_1, \dots, L_\tau)$.
2. A lets x_j be the j -th bit of Γ_B and M_j the string consisting of the j -th bits from all the strings L_i , i.e. $M_j = L_{1,j} || L_{2,j} || \dots || L_{\ell,j}$.
3. B lets Γ_A be the string consisting of all the bits y_i , i.e. $\Gamma_A = y_1 || y_2 || \dots || y_\ell$, and lets K_j be the string consisting of the j -th bits from all the strings N_i , i.e. $K_j = N_{1,j} || N_{2,j} || \dots || N_{\ell,j}$.
4. A and B now hold $[x_j]_A = (x_j, M_j, K_j)$ for $j = 1, \dots, \ell$.

Fig. 11. Sub-protocol for reducing $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$ to $\text{LaBit}^{\mathcal{L}}(\tau, \ell)$

Theorem 2. Let $\tau = \frac{22}{3}\psi$ and \mathcal{L} be a $(\frac{3}{4}\tau)$ -secure leakage function on τ bits. The protocol in Fig. 9 securely implements $\text{aBit}(\ell, \psi)$ in the $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$ -hybrid model with security parameter ψ . The communication is $O(\psi^2)$ and the work is $O(\psi^2\ell)$.

4.2 Bit Authentication with Leaking Bits (LaBit)

We now show another insecure box for aBit. The new box is insecure in the sense that a few of the bits to be authenticated might leak to the other party. We call this box an aBit with leaking bits, or LaBit and formally describe it in Fig. 10. The box $\text{LaBit}^{\mathcal{L}}(\tau, \ell)$ outputs τ authenticated bits with keys of length ℓ , and is parameterized by a class of leakage functions \mathcal{L} on τ -bits. We show that $\text{WaBit}^{\mathcal{L}}$ can be reduced to $\text{LaBit}^{\mathcal{L}}$. In the reduction, a LaBit that outputs authenticated bits $[y_i]_B$ to B is turned into a WaBit that outputs authenticated bits $[x_j]_A$ to A, therefore we present the LaBit box that outputs bits to B. The reduction is strongly inspired by the OT extension techniques in [18].

Theorem 3. For all ℓ, τ and \mathcal{L} the boxes $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$ and $\text{LaBit}^{\mathcal{L}}(\tau, \ell)$ are linear locally equivalent, i.e., can be implemented given the other in linear time without interaction.

The proof can be found in the full version. Note that since we turn $\text{LaBit}^{\mathcal{L}}(\tau, \ell)$ into $\text{WaBit}^{\mathcal{L}}(\ell, \tau)$, if we choose $\ell = \text{poly}(\psi)$ we can turn a relatively small number ($\tau = \frac{22}{3}\psi$) of authenticated bits towards one player into a very larger number (ℓ) of authenticated bits towards the other player.

4.3 A Protocol for Bit Authentication with Leaking Bits

In this section we show how to construct authenticated bits starting from OTs. The protocol ensures that most of the authenticated bits will be kept secret, as specified by the LaBit box in Fig. 10.

The main idea of the protocol, described in Fig. 12, is the following: many authenticated bits $[y_i]_B$ for B are created using OTs, where A is supposed to input messages $(L_i, L_i \oplus \Gamma_B)$. To check that A is using the same Γ_B in every OT, the authenticated bits are randomly paired. Given a pair of authenticated bits $[y_i]_B, [y_j]_B$, A and B compute $[z_i]_B = [y_i]_B \oplus [y_j]_B \oplus d_i$ where $d_i = y_i \oplus y_j$ is announced by B. If A behaved

1. A samples $\Gamma_B \in_R \{0, 1\}^\ell$ and for $i = 1, \dots, \mathcal{T}$ samples $L_i \in_R \{0, 1\}^\ell$, where $\mathcal{T} = 2\tau$.
2. B samples $(y_1, \dots, y_{\mathcal{T}}) \in_R \{0, 1\}^{\mathcal{T}}$.
3. They run \mathcal{T} OTs, where for $i = 1, \dots, \mathcal{T}$ party A offers $(Y_{i,0}, Y_{i,1}) = (L_i, L_i \oplus \Gamma_B)$ and B selects y_i and receives $N_i = Y_{i,y_i} = L_i \oplus y_i \Gamma_B$. Let $[y_1]_B, \dots, [y_{\mathcal{T}}]_B$ be the candidate authenticated bits produced so far.
4. B picks a uniformly random pairing π (a permutation $\pi : \{1, \dots, \mathcal{T}\} \rightarrow \{1, \dots, \mathcal{T}\}$ where $\forall i, \pi(\pi(i)) = i$), and sends π to A. Given a pairing π , let $\mathcal{S}(\pi) = \{i | i \leq \pi(i)\}$, i.e., for each pair, add the smallest index to $\mathcal{S}(\pi)$.
5. For all τ indices $i \in \mathcal{S}(\pi)$:
 - (a) B announces $d_i = y_i \oplus y_{\pi(i)}$.
 - (b) A and B compute $[z_i]_B = [y_i]_B \oplus [y_{\pi(i)}]_B \oplus d_i$.
 - (c) Let Z_i and W_i be the MAC and the local key for z_i held by A respectively B. They compare these using EQ and abort if they are different.
- The τ comparisons are done using *one* call on the $\tau\ell$ -bit strings $(Z_i)_{i \in \mathcal{S}(\pi)}$ and $(W_i)_{i \in \mathcal{S}(\pi)}$.
6. For all $i \in \mathcal{S}(\pi)$ A and B output $[y_i]_B$.

Fig. 12. The protocol for reducing LaBit(τ, ℓ) to OT($2\tau, \ell$) and EQ($\tau\ell$)

honestly, she knows the MAC that B holds on z_i , otherwise she has 1 bit of entropy on this MAC, as shown below. The parties can check if A knows the MAC using the EQ box described in Sect. 2. As B reveals $y_i \oplus y_j$, they waste $[y_j]_B$ and only use $[y_i]_B$ as output from the protocol—as y_j is uniformly random $y_i \oplus y_j$ leaks no information on y_i . Note that we cannot simply let A reveal the MAC on z_i , as a malicious B could announce $1 \oplus z_i$: this would allow B to learn a MAC on z_i and $1 \oplus z_i$ at the same time, thus leaking Γ_B . Using EQ forces A thus cheating B to guess the MAC on a bit which he did not see, which he can do only with negligible probability $2^{-\ell}$.

Note that if A uses different Γ_B in two paired instances, Γ_i and Γ_j say, then the MAC held by B on $y_i \oplus y_j$ (and therefore also z_i) is $(L_i \oplus y_i \Gamma_i) \oplus (L_j \oplus y_j \Gamma_j) = (L_i \oplus L_j) \oplus (y_i \oplus y_j) \Gamma_j \oplus y_i (\Gamma_i \oplus \Gamma_j)$. Since $(\Gamma_i \oplus \Gamma_j) \neq 0^\ell$ and $y_i \oplus y_j$ is fixed by announcing d_i , guessing this MAC is equivalent to guessing y_i . As A only knows $L_i, L_j, \Gamma_i, \Gamma_j$ and $y_i \oplus y_j$, she cannot guess y_i with probability better than $1/2$. Therefore, if A cheats in many OTs, she will get caught with high probability. If she only cheats on a few instances she might pass the test. Doing so confirms her guess on y_i in the pairs where she cheated. Now assume that she cheated in instance i and offered $(L_i, L_i \oplus \Gamma'_B)$ instead of $(L_i, L_i \oplus \Gamma_B)$. After getting her guess on y_i confirmed she can explain the run as an honest run: If $y_i = 0$, the run is equivalent to having offered $(L_i, L_i \oplus \Gamma_B)$, as B gets no information on the second message when $y_i = 0$. If $y_i = 1$, then the run is equivalent to having offered $(L'_i, L'_i \oplus \Gamma_B)$ with $L'_i = L_i \oplus (\Gamma_B \oplus \Gamma'_B)$, as $L'_i \oplus \Gamma_B = L_i \oplus \Gamma_B$ and B gets no information on the first message when $y_i = 1$. So, any cheating strategy of A can be simulated by letting her honestly use the same Γ_B in all pairs and then let her try to guess some bits y_i . If she guesses wrong, the deviation is reported to B. If she guesses right, she is told so and the deviation is not reported to B. This, in turn, can be captured using some appropriate class of leakage functions \mathcal{L} . Nailing down the exact \mathcal{L} needed to simulate a given behavior of A, including defining what is the “right” Γ_B , and showing that the needed \mathcal{L} is always κ -secure is a relatively straight-forward but very tedious business. The proof of the following theorem can be found in the full version.

Theorem 4. Let $\kappa = \frac{3}{4}\tau$, and let \mathcal{L} be a κ secure leakage function on τ bits. The protocol in Fig. 12 securely implements LaBit $^{\mathcal{L}}(\tau, \ell)$ in the $(\text{OT}(2\tau, \ell), \text{EQ}(\tau\ell))$ -hybrid model. The communication is $O(\tau^2)$. The work is $O(\tau\ell)$.

Corollary 1. Let ψ denote the security parameter and let $\ell = \text{poly}(\psi)$. The box aBit (ℓ, ψ) can be reduced to $(\text{OT}(\frac{44}{3}\psi, \psi), \text{EQ}(\psi))$. The communication is $O(\psi\ell + \psi^2)$ and the work is $O(\psi^2\ell)$.

Proof. Combining the above theorems we have that aBit (ℓ, ψ) can be reduced to $(\text{OT}(\frac{44}{3}\psi, \ell), \text{EQ}(\frac{22}{3}\psi\ell))$ with communication $O(\psi^2)$ and work $O(\psi^2\ell)$. For any polynomial ℓ , we can implement $\text{OT}(\frac{44}{3}\psi, \ell)$ given $\text{OT}(\frac{44}{3}\psi, \psi)$ and a pseudo-random generator prg : $\{0, 1\}^\psi \rightarrow \{0, 1\}^\ell$. Namely, seeds are sent using the OTs and the prg is used to one-time pad encrypt the messages. The communication is 2ℓ . If we use the RO to implement the pseudo-random generator and count the hashing of κ bits as $O(\kappa)$ work, then the work is $O(\ell\psi)$. We can implement $\text{EQ}(\frac{22}{3}\psi\ell)$ by comparing short hashes produced using the RO. The work is $O(\psi\ell)$. \square

Since the oracles $(\text{OT}(\frac{44}{3}\psi, \psi), \text{EQ}(\psi))$ are independent of ℓ , the cost of essentially any reasonable implementation of them can be amortized away by picking ℓ large enough. See the full version for a more detailed complexity analysis.

Efficient OT Extension: We notice that the WaBit box resembles an intermediate step of the OT extension protocol of [18]. Completing their protocol (i.e., “hashing away” the fact that all messages pairs have the same XOR), gives an efficient protocol for OT extension, with the same asymptotic complexity as [15], but with dramatically smaller constants. See the full version for details.

5 Authenticated Oblivious Transfer

In this section we show how to implement aOTs. We implemented aBits in Sect. 4, so what remains is to show how to implement aOTs from aBits i.e., to implement the $\mathcal{F}_{\text{DEAL}}$ box when it outputs $[x_0]_{\mathbf{A}}, [x_1]_{\mathbf{A}}, [c]_{\mathbf{B}}, [z]_{\mathbf{B}}$ with $z = c(x_0 \oplus x_1) \oplus x_0 = x_c$. Because of symmetry we only show the construction of aOTs from aBits with \mathbf{A} as sender and \mathbf{B} as receiver.

We go via a leaky version of authenticated OT, or LaOT, described in Fig. 14. The LaOT box is leaky in the sense that choice bits may leak when \mathbf{A} is corrupted: a corrupted \mathbf{A} is allowed to make guesses on choice bits, but if the guess is wrong the box aborts revealing that \mathbf{A} is cheating. This means that if the box does not abort, with very high probability \mathbf{A} only tried to guess a few choice bits.

The protocol to construct a leaky aOT, described in Fig. 15, proceeds as follows: First \mathbf{A} and \mathbf{B} get $[x_0]_{\mathbf{A}}, [x_1]_{\mathbf{A}}$ (\mathbf{A} 's messages), $[c]_{\mathbf{B}}$ (\mathbf{B} 's choice bit) and $[r]_{\mathbf{B}}$. Then \mathbf{A} transfers the message $z = x_c$ to \mathbf{B} in the following way: \mathbf{B} knows the MAC for his choice bit M_c , while \mathbf{A} knows the two keys K_c and Δ_B . This allows \mathbf{A} to compute

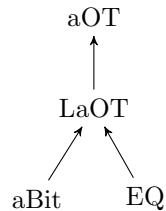


Fig. 13. Sect. 5 outline

Honest Parties: For $i = 1, \dots, \ell$, the box outputs random $[x_0^i]_A, [x_1^i]_A, [c^i]_B, [z^i]_B$ with $z^i = c^i(x_0^i \oplus x_1^i) \oplus x_0^i$.

Corrupted Parties:

1. If B is corrupted he gets to choose all his random values.
2. If A is corrupted she gets to choose all her random values. Also, she may, at any point before B received his outputs, input (i, g^i) to the box in order to try to guess c^i . If $c^i \neq g^i$ the box will output `fail` and terminate. Otherwise the box proceeds as if nothing has happened and A will know the guess was correct. She may input as many guesses as she desires.

Global Key Queries: The adversary can at any point input (A, Δ) and will be returned whether $\Delta = \Delta_B$. And it can at any point input (B, Δ) and will be returned whether $\Delta = \Delta_A$.

Fig. 14. The Leaky Authenticated OT box LaOT(ℓ)

The protocol runs ℓ times in parallel. The description is for a single run of LaOT.

1. A and B get $[x_0]_A, [x_1]_A, [c]_B$ and $[r]_B$ from the dealer using the aBit box.
2. Let $[x_0]_A = (x_0, M_{x_0}, K_{x_0}), [x_1]_A = (x_1, M_{x_1}, K_{x_1}), [c]_B = (c, M_c, K_c), [r]_B = (r, M_r, K_r)$.
3. A chooses random strings $T_0, T_1 \in \{0, 1\}^\kappa$.
4. A sends (X_0, X_1) to B where $X_0 = H(K_c) \oplus (x_0 || M_{x_0} || T_{x_0})$ and $X_1 = H(K_c \oplus \Delta_B) \oplus (x_1 || M_{x_1} || T_{x_1})$.
5. B computes $(x_c || M_{x_c} || T_{x_c}) = X_c \oplus H(M_c)$. B aborts if $M_{x_c} \neq K_{x_c} \oplus x_c \Delta_A$. Otherwise, let $z = x_c$.
6. B announces $d = z \oplus r$ to A and the parties compute $[z]_B = [r]_B \oplus d$. Let $[z]_B = (z, M_z, K_z)$.
7. A sends (I_0, I_1) to B where $I_0 = H(K_z) \oplus T_1$ and $I_1 = H(K_z \oplus \Delta_B) \oplus T_0$.
8. B computes $T_{1 \oplus z} = I_z \oplus H(M_z)$. Notice that now B has both (T_0, T_1) .
9. A and B both input (T_0, T_1) to EQ. The comparisons are done using one call to $\text{EQ}(\ell 2\kappa)$.
10. If the values are the same, they output $[x_0]_A, [x_1]_A, [c]_B, [z]_B$.

Fig. 15. The protocol for authenticated OT with leaky choice bit

the two possible MACs $(K_c, K_c \oplus \Delta_B)$ respectively for the case of $c = 0$ and $c = 1$. Hashing these values leaves A with two uncorrelated strings $H(K_c)$ and $H(K_c \oplus \Delta_B)$, one of which B can compute as $H(M_c)$. These values can be used as a one-time pad for A's bits x_0, x_1 (and some other values as described later). B can retrieve x_c and announce the difference $d = x_c \oplus r$ and therefore compute the output $[z]_B = [r]_B \oplus d$.

In order to check if A is transmitting the correct bits x_0, x_1 , she will transfer the respective MACs together with the bits: as B is supposed to learn x_c , revealing the MAC on this bit does not introduce any insecurity. However, A can now mount a selective failure attack: A can check if B's choice bit c is equal to, e.g., 0 by sending x_0 with the right MAC and x_1 together with a random string. Now if $c = 0$ B only sees the valid MAC and continues the protocol, while if $c = 1$ B aborts because of the wrong MAC.

1. A and B generate $\ell' = B\ell$ authenticated OTs using LaOT(ℓ'). If the box does not abort, name the outputs $\{[x_0^i]_A, [x_1^i]_A, [c^i]_B, [z^i]_B\}_{i=1}^{\ell'}$.
2. B sends a B -wise independent permutation π on $\{1, \dots, \ell'\}$ to A. For $j = 0, \dots, \ell - 1$, the B quadruples $\{[x_0^{\pi(i)}]_A, [x_1^{\pi(i)}]_A, [c^{\pi(i)}]_B, [z^{\pi(i)}]_B\}_{i=jB+1}^{jB+B}$ are defined to be in the j -th bucket.
3. We describe how to combine two OTs from a bucket, call them $[x_0^1]_A, [x_1^1]_A, [c^1]_B, [z^1]_B$ and $[x_0^2]_A, [x_1^2]_A, [c^2]_B, [z^2]_B$. Call the result $[x_0]_A, [x_1]_A, [c]_B, [z]_B$. To combine more than two, just iterate by taking the result and combine it with the next leaky OT.
 - (a) A reveals $d = x_0^1 \oplus x_1^1 \oplus x_0^2 \oplus x_1^2$.
 - (b) Compute: $[c]_B = [c^1]_B \oplus [c^2]_B, [z]_B = [z^1]_B \oplus [z^2]_B \oplus d[c^1]_B, [x_0]_A = [x_0^1]_A \oplus [x_0^2]_A, [x_1]_A = [x_0^1]_A \oplus [x_1^2]_A$.

Fig. 16. From Leaky Authenticated OTs to Authenticated OTs

A similar attack can be mounted to check if $c = 1$. We will fix this later by randomly partitioning and combining a few LaOTs together.

On the other hand, if B is corrupted, he could be announcing the wrong value d . In particular, A needs to check that the authenticated bit $[z]_B$ is equal to x_c without learning c . In order to do this, we have A choosing two random strings T_0, T_1 , and append them, respectively, to x_0, x_1 and the MACs on those bits, so that B learns T_c together with x_c . After B announces d , we can again use the MAC and the keys for z to perform a new transfer: A uses $H(K_z)$ as a one-time pad for T_1 and $H(K_z \oplus \Delta_B)$ as a one-time pad for T_0 . Using M_z , the MAC on z , B can retrieve $T_{1 \oplus z}$. This means that an honest B, that sets $z = x_c$, will know both T_0 and T_1 , while a dishonest B will not be able to know both values except with negligible probability. Using the EQ box A can check that B knows both values T_0, T_1 . Note that we cannot simply have B openly announce these values, as this would open the possibility for new attacks on A's side. The proof of the following theorem can be found in the full version.

Theorem 5. *The protocol in Fig. 15 securely implements LaOT(ℓ) in the (aBit($4\ell, \kappa$), EQ($2\ell\kappa$))-hybrid model.*

To deal with the leakage of the LaOT box, we let B randomly partition the LaOTs in small buckets: all the LaOTs in a bucket will be combined using an OT combiner, as shown in Fig. 16, in such a way that if at least one choice bit in every bucket is unknown to A, then the resulting aOT will not be leaky. The overall protocol is secure because of the OT combiner and the probability that any bucket is filled only with OTs where the choice bit leaked is negligible, as shown in the full version.

Theorem 6. *Let aOT(ℓ) denote the box which outputs ℓ aOTs as in $\mathcal{F}_{\text{DEAL}}$. If $(\log_2(\ell) + 1)(B - 1) \geq \psi$, then the protocol in Fig. 16 securely implements aOT(ℓ) in the LaOT($B\ell$)-hybrid model with security parameter ψ .*

6 Experimental Results

We did a proof-of-concept implementation in Java. The hash function in our protocol was implemented using Java's standard implementation of SHA256. The implementation

consists of a circuit-independent protocol for preprocessing all the random values output by $\mathcal{F}_{\text{DEAL}}$, a framework for constructing circuits for a given computation, and a run-time system which takes preprocessed values, circuits and inputs and carry out the secure computation.

We will not dwell on the details of the implementation, except for one detail regarding the generation of the circuits. In our implementation, we do not compile the function to be evaluated into a circuit in a separate step. The reason is that this would involve storing a huge, often highly redundant, circuit on the disk, and reading it back. This heavy disk access turned out to constitute a significant part of the running time in an earlier of our prototype implementations, which we discarded. Instead, in the current prototype, circuits are generated on the fly, in chunks which are large enough that their evaluation generate large enough network packages that we can amortize away communication latency, but small enough that the circuit chunks can be kept in memory during their evaluation. A circuit compiled is hence replaced by a succinct program which generates the circuit in a streaming manner. This circuit stream is then sent through the runtime machine, which receives a separate stream of preprocessed $\mathcal{F}_{\text{DEAL}}$ -values from the disk and then evaluates the circuit chunk by chunk in concert with the runtime machine at the other party in the protocol. The stream of preprocessed $\mathcal{F}_{\text{DEAL}}$ -values from the disk is still expensive, but we currently see no way to avoid this disk access, as the random nature of the preprocessed values seems to rule out a succinct representation.

For timing we did oblivious ECB-AES encryption. (Both parties input a secret 128-bit key K_A respectively K_B , defining an AES key $K = K_A \oplus K_B$. A inputs a secret ℓ -block message $(m_1, \dots, m_\ell) \in \{0, 1\}^{128\ell}$. B learns $(E_K(m_1), \dots, E_K(m_\ell))$.) We used a modified version of the AES circuit from [31] and we thank Benny Pinkas, Thomas Schneider, Nigel P. Smart and Stephen C. Williams for providing us with this circuit.

The reason for using AES is that it provides a reasonable sized circuit which is also reasonably complex in terms of the structure of the circuit and the depth, as opposed to just running a lot of AND gates in parallel. Also, AES has been used for benchmark in previous implementations, like [31], which allows us to do a crude comparison to previous implementations. The comparison can only become crude, as the experiments were run in different experimental setups.

In the timings we ran A and B on two different machines on Aarhus University's intranet (using two Intel Xeon E3430 2.40GHz cores on each machine). We recorded the number of Boolean gates evaluated (G), the time spent in preprocessing (T_{pre}) and the time spent by the run-time system (T_{onl}). In the table in Fig. 17 we also give the amortized time per AES encryption (T_{tot}/ℓ with $T_{\text{tot}} \stackrel{\text{def}}{=} T_{\text{pre}} + T_{\text{onl}}$) and the number of gates handled per second (G/T_{tot}). The time T_{pre} covers the time spent on computing and communicating during the generation of the values preprocessed by $\mathcal{F}_{\text{DEAL}}$, and the time spent storing these value to a local disk. The time T_{onl} covers the time spent on generating the circuit and the computation and communication involved in evaluating the circuit given the values preprocessed by $\mathcal{F}_{\text{DEAL}}$.

We work with two security parameters. The computational security parameter κ specifies that a poly-time adversary should have probability at most $\text{poly}(\kappa)2^{-\kappa}$ in breaking the protocol. The statistical security parameter σ specifies that we allow the protocol to break with probability $2^{-\sigma}$ independent of the computational power of the

ℓ	G	σ	T_{pre}	T_{onl}	T_{tot}/ℓ	G/T_{tot}
1	34,520	55	38	4	44	822
27	922,056	55	38	5	1.6	21,545
54	1,842,728	58	79	6	1.6	21,623
81	2,765,400	60	126	10	1.7	20,405
108	3,721,208	61	170	12	1.7	20,541
135	4,642,880	62	210	15	1.7	20,637
256	8,739,200	65	406	16	1.7	20,709
512	17,478,016	68	907	26	1.8	18,733
1,024	34,955,648	71	2,303	52	2.3	14,843
2,048	69,910,912	74	5,324	143	2.7	12,788
4,096	139,821,440	77	11,238	194	2.8	12,231
8,192	279,642,496	80	22,720	258	2.8	12,170
16,384	559,284,608	83	46,584	517	2.9	11,874

Fig. 17. Timings: The reported time for $\ell \leq 135$ is the average over 5 runs. For $\ell \geq 256$ is for single runs. Units are as follows: ℓ is number of 128-bit blocks encrypted, G is Boolean gates, σ is bits of security, T_{pre} , T_{onl} , T_{tot} are seconds.

adversary. As an example of the use of κ , our keys and therefore MACs have length κ . This is needed as the adversary learns $H(K_i)$ and $H(K_i \oplus \Delta)$ in our protocols and can break the protocol given Δ . As an example of the use of σ , when we generate ℓ gates with bucket size B , then $\sigma \leq (\log_2(\ell) + 1)(B - 1)$ due to the probability $(2\ell)^{1-B}$ that a bucket might end up containing only leaky components. This probability is independent of the computational power of the adversary, as the components are being bucketed by the honest party after it is determined which of them are leaky.

In the timings, the computational security parameter has been set to 120. Since our implementation has a fixed bucket size of 4, the statistical security level depends on ℓ . In the table, we specify the statistical security level attained (σ means insecurity $2^{-\sigma}$). At computational security level 120, the implementation needs to do 640 seed OTs. The timings do not include the time needed to do these, as that would depend on the implementation of the seed OTs, which is not the focus here. We note, however, that using, e.g., the implementation in [31], the seed OTs could be done in around 20 seconds, so they would not significantly affect the amortized times reported.

The dramatic drop in amortized time from $\ell = 1$ to $\ell = 27$ is due to the fact that the preprocessor, due to implementation choices, has a smallest unit of gates it can preprocess for. The largest number of AES circuits needing only one, two, three, four and five units is 27, 54, 81, 108 and 135, respectively. Hence we preprocess equally many gates when $\ell = 1$ and $\ell = 27$.

As for total time, we found the best amortized behavior at $\ell = 54$, where oblivious AES encryption of one block takes amortized 1.6 seconds, and we handle 21,623 gates per second. As for online time, we found the best amortized behavior at $\ell = 2048$, where handling one AES block online takes amortized 32 milliseconds, and online we handle 1,083,885 gates per second. We find these timings encouraging and we plan an implementation in a more machine-near language, exploiting some of the findings from implementing the prototype.

References

1. Shelat, A., Shen, C.-H.: Two-Output Secure Computation with Malicious Adversaries. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 386–405. Springer, Heidelberg (2011)
2. Applebaum, B., Harnik, D., Ishai, Y.: Semantic security under related-key attacks and applications. In: *ICS*, pp. 45–60 (2011)
3. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a system for secure multi-party computation. In: *CCS*, pp. 257–266 (2008)
4. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic Encryption and Multiparty Computation. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 169–188. Springer, Heidelberg (2011)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: *FOCS*, pp. 136–145 (2001)
6. Choi, S.G., Hwang, K.-W., Katz, J., Malkin, T., Rubenstein, D.: Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces. In: Dunkelman, O. (ed.) *CT-RSA 2012*. LNCS, vol. 7178, pp. 416–432. Springer, Heidelberg (2012)
7. Choi, S.G., Katz, J., Kumaresan, R., Zhou, H.-S.: On the Security of the “Free-XOR” Technique. In: Cramer, R. (ed.) *TCC 2012*. LNCS, vol. 7194, pp. 39–53. Springer, Heidelberg (2012)
8. Crépeau, C., van de Graaf, J., Tapp, A.: Committed Oblivious Transfer and Private Multiparty Computation. In: Coppersmith, D. (ed.) *CRYPTO 1995*. LNCS, vol. 963, pp. 110–123. Springer, Heidelberg (1995)
9. Damgård, I., Keller, M.: Secure Multiparty AES. In: Sion, R. (ed.) *FC 2010*. LNCS, vol. 6052, pp. 367–374. Springer, Heidelberg (2010)
10. Damgård, I., Orlandi, C.: Multiparty Computation for Dishonest Majority: From Passive to Active Security at Low Cost. In: Rabin, T. (ed.) *CRYPTO 2010*. LNCS, vol. 6223, pp. 558–576. Springer, Heidelberg (2010)
11. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword Search and Oblivious Pseudorandom Functions. In: Kilian, J. (ed.) *TCC 2005*. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005)
12. Garay, J.A., MacKenzie, P., Yang, K.: Efficient and Universally Composable Committed Oblivious Transfer and Applications. In: Naor, M. (ed.) *TCC 2004*. LNCS, vol. 2951, pp. 297–316. Springer, Heidelberg (2004)
13. Goldreich, O.: Foundations of Cryptography, vol. 2. Cambridge University Press (2004), <http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html>
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: *STOC*, pp. 218–229 (1987)
15. Harnik, D., Ishai, Y., Kushilevitz, E., Nielsen, J.B.: OT-Combiners via Secure Computation. In: Canetti, R. (ed.) *TCC 2008*. LNCS, vol. 4948, pp. 393–411. Springer, Heidelberg (2008)
16. Henecka, W., Kögl, S., Sadeghi, A.-R., Schneider, T., Wehrenberg, I.: Tasty: tool for automating secure two-party computations. In: *CCS*, pp. 451–462 (2010)
17. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium (2011)
18. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending Oblivious Transfers Efficiently. In: Boneh, D. (ed.) *CRYPTO 2003*. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
19. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: *STOC*, pp. 433–442 (2008)
20. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)

21. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure Arithmetic Computation with No Honest Majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 294–314. Springer, Heidelberg (2009)
22. Jakobsen, T.P., Makkes, M.X., Nielsen, J.D.: Efficient Implementation of the Orlandi Protocol. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 255–272. Springer, Heidelberg (2010)
23. Kolesnikov, V., Schneider, T.: Improved Garbled Circuit: Free XOR Gates and Applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórssón, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
24. Lindell, Y., Oxman, E., Pinkas, B.: The IPS Compiler: Optimizations, Variants and Concrete Efficiency. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 259–276. Springer, Heidelberg (2011)
25. Lindell, Y., Pinkas, B.: Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer, Heidelberg (2011)
26. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
27. Malka, L., Katz, J.: VMCrypt - modular software architecture for scalable secure computation. Cryptology ePrint Archive, Report 2010/584 (2010), <http://eprint.iacr.org/>
28. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: USENIX Security Symposium, pp. 287–302 (2004)
29. Nielsen, J.B.: Extending oblivious transfers efficiently - how to get robustness almost for free. Cryptology ePrint Archive, Report 2007/215 (2007), <http://eprint.iacr.org/>
30. Nielsen, J.B., Orlandi, C.: LEGO for Two-Party Secure Computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009)
31. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation Is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
32. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: FOCS, pp. 160–164 (1982)

The Curious Case of Non-Interactive Commitments – On the Power of Black-Box vs. Non-Black-Box Use of Primitives

Mohammad Mahmoody and Rafael Pass^{*}

Cornell
`{mohammad,rafael}@cs.cornell.edu`

Abstract. It is well-known that one-way permutations (and even one-to-one one-way functions) imply the existence of non-interactive commitments. Furthermore the construction is *black-box* (i.e., the underlying one-way function is used as an oracle to implement the commitment scheme, and an adversary attacking the commitment scheme is used as an oracle in the proof of security).

We rule out the possibility of black-box constructions of non-interactive commitments from general (possibly not one-to-one) one-way functions. As far as we know, this is the first result showing a natural cryptographic task that can be achieved in a black-box way from one-way permutations but not from one-way functions.

We next extend our black-box separation to constructions of non-interactive commitments from a stronger notion of one-way functions, which we refer to as *hitting* one-way functions. Perhaps surprisingly, Barak, Ong, and Vadhan (Siam JoC '07) showed that there does exist a non-black-box construction of non-interactive commitments from hitting one-way functions. As far as we know, this is the first result to establish a “separation” between the power of black-box and non-black-box use of a primitive to implement a natural cryptographic task.

We finally show that unless the complexity class NP has program checkers, the above separations extend also to non-interactive instance-based commitments, and 3-message public-coin honest-verifier zero-knowledge protocols with $O(\log n)$ -bit verifier messages. The well-known classical zero-knowledge proof for NP fall into this category.

Keywords: Non-Black-Box Constructions, Black-Box Separations, One-Way Functions, Non-Interactive Commitments, Zero-Knowledge Proofs, Program Checkers, Hitting Set Generators.

* Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF CAREER Award CCF-0746990, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US government.

1 Introduction

It is well-known that most of the cryptographic constructions are “black-box” in the sense that they ignore the specific implementation of the primitive, and they use both the primitive and the adversary (in the proof of security) as an oracle. Thus black-box constructions capture a main body of our techniques in cryptography for designing protocols and proving their security. In addition, black-box constructions are usually much more efficient than their non-black-box counterparts. In light of this, studying the power and limits of black-box constructions has been a major line of research in cryptography, aiming at finding the “minimal cryptographic primitives” under which a cryptographic task \mathcal{Q} is possible and “separating” \mathcal{Q} from “weaker primitives”.

Black-Box Separations. The seminal work of Impagliazzo and Rudich [44] put forward a framework for proving the limits of black-box constructions by separating public-key cryptography from private-key cryptography when the construction is black-box. Many other black-box separation results were subsequently established (e.g., [12, 21, 22, 46, 51, 62, 64]¹). Reingold, Trevisan, and Vadhan [60] further studied various forms of black-box constructions (based on their proof of security). ² In search of the “minimal” computational primitives required for accomplishing cryptographic tasks, one-way functions emerge as the central player: Almost all natural cryptographic primitives “imply” one-way functions [38, 43, 57]; moreover, all these constructions are black-box.

One-Way Functions vs. Permutations. One-way permutations are a closely related primitive to one-way functions. Even though it is known that there is no black-box construction of one-way permutations from one-way functions [8, 41, 45, 61, 63]³, a surprisingly successful line of research has been to first realize a cryptographic task securely based on the existence of one-way permutations, weaken the assumption to one-to-one one-way functions, and then eventually obtain a construction solely based on the existence of general one-way functions. Examples of this phenomenon include works on pseudorandom generators [11, 25, 26, 42, 49, 66] and statistical zero-knowledge arguments as well as statistically-hiding commitments [13, 14, 18, 23, 31, 35, 37, 39, 40, 55, 56].

¹ A closely related line of research aimed at proving lower-bounds on the *efficiency* of black-box constructions (e.g., [6, 17, 19, 34, 47, 50]).

² Our notion of black-box construction here corresponds to the notion of *fully* black-box construction as defined in [60] where we also include the security parameter; see Definition 4.

³ The results implicit in [8, 41, 63] show that there is no fully black-box construction of one-way permutations from one-way functions (see [51] for an exposition of this argument). This results extends even to separating one-way functions from injective one-way functions. Rudich [61] observes that this separation is implicit in those previous works and improves them to separate one-way permutations from *random oracles*, even if the construction is allowed to have small completeness error, at the cost of assuming a combinatorial conjecture that was later resolved in [45]. See [61] for more discussions.

Why Trying to Rely on One-Way Functions? We emphasize that all known candidates for one-way permutations are based on structured number-theoretic assumptions, and the vulnerability of such structured primitives to possible algebraic (sub-exponential) attacks [48] makes the feasibility of using one-way functions (rather than permutations) interesting both from theoretical and practical points of view. This puts forward the following basic question:

Main Question 1: *Is there any natural cryptographic task that can be accomplished based on the black-box assumption of one-way permutations but not one-way functions?*

We consider one-way functions and permutations both as *computational assumptions* and not as natural cryptographic *tasks*, and so the separation of one-way permutations from one-way functions does not answer our question above.

The Power of Black-Box vs. Non-Black-Box Constructions. Another similar successful line of research in the foundations of cryptography has been to start by providing non-black-box constructions of a primitive and later turning them into black-box ones. Examples include e.g., secure computations from various primitive [15, 16, 32, 36, 65], oblivious transfer from semi-honest oblivious transfer [33], constant-round zero-knowledge arguments and trapdoor commitments from one-way functions [58], etc. Despite this, as far as we know the following intriguing question has remained open:

Main Question 2: *Is there a natural cryptographic task \mathcal{Q} that can be based on a cryptographic primitive \mathcal{P} in a non-black-box way, while no black-box construction of \mathcal{Q} based on \mathcal{P} exists?*

In this work we answer both the above questions affirmatively: **(1)** There is a cryptographic task that can be based on one-way permutations but not one-way functions in a black-box way. **(2)** The same primitive can be used to separates the power of black-box and non-black-box constructions. Interestingly, the primitive is a very natural cryptographic building block: *non-interactive commitments*.

Commitment Schemes. Bit-commitments are one of the most fundamental cryptographic building blocks. Their application ranges from zero-knowledge proofs [28, 30] to secure computations [27]. Roughly speaking, a commitments scheme is a two-stage protocol between two parties: the sender and the receiver. In the first, so-called, *commitment phase*, the sender commits to a secret bit b ; and then later in the *decommitment phase*, the sender reveals the bit b together with some additional information which allows the receiver to verify the correctness of the decommitment. Commitment schemes are required to satisfy two properties: *hiding* and *binding*. Roughly speaking, the hiding property stipulates that after the commitment phase the bit b should remain hidden to the receiver, whereas the binding property asserts that in the decommitment phase the sender is not able to decommit successfully to both $b = 0$ and $b = 1$.

The results of Naor [54] and Håstad, Impagliazzo, Luby and Levin [42] establish that the existence of one-way functions implies the existence of commitment schemes where the commitment phase consists of two messages. Furthermore their construction is black-box and the commitment scheme uses the underlying one-way function as an oracle. On the other hand, Impagliazzo and Luby [43] establish that the existence of commitment schemes implies the existence of one-way functions (in a black-box way).

In this work we focus on the black-box complexity of *non-interactive commitments*—namely, commitment schemes where both the commitment phase and the decommitment phase consist of a single message. The results of [9, 26] establish the existence of non-interactive commitments based on one-way permutations (or even one-to-one one-way functions) using a black-box construction. These results extend even to the case of *families* of one-way permutations where given an index p one can efficiently verify that f_p is indeed a permutation.⁴ The work of Naor showed how to obtain interactive commitments based on any one-way function in a black-box way, where the commitment phase consists only of a random message from the receiver followed by a message from the sender (thus the first message can be eliminated in the common reference string model). In this work we study the following natural question left open by previous work: *Is there a black-box construction of non-interactive commitments from one-way functions?* We provide a negative answer:

Theorem 1. *There is no black-box construction of non-interactive commitments from one-way functions.*

The separation extends to stronger primitives than one-way functions (e.g., *families* of collision-resistant hash function). As far as we know, this is the first result showing a natural cryptographic task that can be constructed in a black-box way from one-way permutations but not from one-way functions resolving our first question affirmatively.

Non-Black-Box Non-Interactive Commitments from One-Way Functions. The elegant work by Barak, Ong and Vadhan [7] provides a *non-black-box* construction of non-interactive commitments assuming the existence of one-way functions and certain hitting-set generators (see the discussion in Section 3) against co-nondeterministic circuits (see Definition 6 for a formalization) which can be constructed under worst-case complexity assumptions. Roughly speaking, the hitting-set generator $G: \{0, 1\}^\ell \mapsto \{0, 1\}^{\text{poly}(n)}$ is used to derandomize Naor’s 2-message commitment scheme by executing the commitment in parallel over *all* of $G(\{0, 1\}^\ell)$ as the “first messages” of the protocol (thus we require $2^\ell = \text{poly}(n)$). Naor’s commitment has the nice property that for every one-way function used, most of $\{0, 1\}^n$ can be fixed as the first message to make the scheme perfectly binding. The hitting property of the generator G guarantees that at least one of the fixed first messages $G(\{0, 1\}^\ell)$ makes the (non-interactive) scheme binding.

⁴ For example, one can sample a random prime number p and define the permutation f_p to be the discrete logarithm function in the group \mathbb{Z}_p^* . Primality of p can be tested efficiently [1, 52, 59] and this guarantees f_p is indeed a permutation.

Conditional Separation of the Power of Black-Box and Non-Black-Box Constructions. The result of [7] together with our Theorem 1 show that under any complexity assumption that guarantees the existence of hitting-set generators against co-nondeterministic circuits, non-black-box constructions are more powerful than black-box constructions (since a non-black-box construction of non-interactive commitments from one-way functions would exist, while no such black-box construction exists). As we will see shortly, we are able to make this “separation” (between the power of the two models) *unconditional* by defining a new primitive that can be used as a hitting-set generator.

Non-Interactive Commitments from Hitting One-Way Functions. Inspired by the work of [7], we introduce the notion of *hitting one-way functions*; roughly speaking, a (one-way) function f is said to be *hitting*, if for every co-nondeterministic circuit of size n which accepts at least half of its inputs, there exists at least one input $x \in [1, \dots, n^2] \subseteq \{0, 1\}^n$ which $f(x)$ is accepted by the circuit. It is easy to see that a random oracle is a hitting one-way function with overwhelming probability (see Lemma 8). Furthermore, we show that there exists a non-black-box construction of non-interactive commitments from hitting one-way functions as follows. Following [7], we derandomize Naor’s commitment scheme by evaluating the hitting one-way function f on the inputs $1, \dots, n^2$ (appropriately planted in $\{0, 1\}^n$), where n is a polynomial that is determined by the size of the verification circuit in Naor’s commitment. Since Naor’s commitment also relies on the use of the one-way function f , the choice of n depends on the circuit size of f ; thus the construction is non-black-box. Thus we obtain the following theorem whose proof can be found in the full version of the paper.⁵

Theorem 2. *There is a non-black-box construction of non-interactive commitments from hitting one-way functions.*

In contrast, we prove the following theorem in the black-box regime.

Theorem 3. *There is no black-box construction of non-interactive commitments from hitting one-way functions.*

As far as we know, this constitutes the first separation between the power of black-box and non-black-box use of a primitive in the implementation of a natural cryptographic task. This is different from the results of Barak [5] and Goldreich-Krawczyk [24] which provide a separation between the power of black-box and non-black-box *proofs of security*, and in this work all proofs of security are black-box. Thus we also resolve our second main question affirmatively.

Extensions to 3-Message Zero-Knowledge and Instance-Based Commitments. A major application of commitment schemes is to construct zero-knowledge proofs for NP. We also directly study the constructions of 3-message zero-knowledge

⁵ Our positive and negative results are robust to choosing n^2 as the size of the hitting set generator and they can be adopted to work with any function $\omega(n)$. We choose to use n^2 for sake of simplicity.

proofs based on one-way functions and also the type of non-interactive (instance-based) commitments that are useful to construct such zero-knowledge proofs. We extend our impossibility result (of black-box constructions from one-way functions) also for these primitives, but in a *conditional* way. Namely, our separations hold assuming that the complexity class NP does not have “program checkers” [10]. For these results we refer the reader to the full version of the paper.

2 Separation from One-Way Functions

Here we outline the proof of Theorem 1. Due to lack of space, here we settle this theorem only for the natural setting that the verification of the decommitment is deterministic and the scheme has perfect completeness; we refer the reader to full version of the paper for the proof of the general case.

We start by formalizing the notion of black-box constructions by following the paradigm of [60] and incorporating the security parameter. Roughly speaking, black-box constructions consist of two reductions: implementation and proof of security. The implementation Q of the new primitive Q uses any implementation P of the base primitive \mathcal{P} only as an oracle. The security reduction S bases the security of Q^P on the security of P as follows: for every (unbounded) adversary A who breaks the security of Q^P , $S^{A,P}$ breaks the security of P . Note that a commitment scheme has two players, and so breaking the security amounts to breaking either of hiding or binding properties. The following definition formalizes the above definition for the case of commitment schemes.

Definition 4. A black-box construction of non-interactive commitments from one-way functions is a pair of efficient oracle algorithms $\text{COM}^{(\cdot)} = (S^{(\cdot)}, R^{(\cdot)})$ such that: The parties receive the common input 1^n as the security parameter and access an oracle $f = \{f_m : \{0, 1\}^m \mapsto \{0, 1\}^m\}$. The security of the scheme is guaranteed through reductions to the one-wayness of f as follows.

- **Proving the Hiding:** There is an efficient security reduction H that proves that COM^f is hiding. Namely, for every oracle f and every malicious receiver \hat{R} (who could arbitrarily depend on f) that distinguishes commitments to 0, 1 with non-negligible advantage $\varepsilon > 1/\text{poly}(n)$, the oracle algorithm $H^{f, \hat{R}}$ breaks the one-wayness of f with probability at least $\text{poly}(\varepsilon/n)$ over a polynomially related $m = n^{\Theta(1)}$ input length:

$$\Pr_{y \xleftarrow{s} f(\mathbf{U}_m)} [H^{f, \hat{S}}(y) \in f^{-1}(y)] \geq \left(\frac{\varepsilon}{m}\right)^{O(1)}.$$

- **Proving the Binding:** It is defined similarly to the definition of Hiding using another reduction B that inverts f with non-negligible probability given oracle address to f and any adversary who breaks the binding of COM^f .

In order to prove Theorem 1, we employ the methodology formally described in the following lemma (which is also used in the previous works of [6, 17]). See [17] for a proof of a stronger version of this lemma.

Lemma 5. *There is no black-box construction of non-interactive commitments from OWFs, if there is any randomized oracle \mathbf{O} with the following properties:*

1. *The hiding or binding of $\text{COM}^{\mathbf{O}}$ is violated by a poly(n)-query attack.*
2. *\mathbf{O} is strongly one-way in the sense that no poly(n)-query computationally-unbounded adversary can invert \mathbf{O} over $\mathbf{O}(\mathbf{U}_n)$ with probability $\geq 1/\text{poly}(n)$.*

In the following we describe how to find a distribution for the randomized oracle \mathbf{O} so that we can apply Lemma 5 to prove Theorem 1.

\mathbf{O} Cannot be a Random Oracle. We first note that we can not simply use \mathbf{O} to be a random oracle which is indeed a common method to derive separations from one-way functions. This is expected, since otherwise we could also get a separation from one-way permutations (since random oracle and random permutation oracle are indistinguishable over large enough input lengths), and this would be a contradiction. In particular, relative to a random oracle, with high probability, there exists a *one-to-one* one-way function⁶ which is indeed sufficient for constructing non-interactive commitments in a black-box way [9].

Partially-Fixed Random Oracles. We overcome the above obstacle by choosing the distribution of our oracle \mathbf{O} to be *fixed* over a polynomial-size subset \mathcal{F} of its domain (which in fact depends on the construction COM itself), and at any other point out of \mathcal{F} we choose the answers randomly. In general, we call oracles *partially-fixed random*. Partially-fixed random oracles allow us to bypass the obstacle explained above against random oracles, because the way we fix the part \mathcal{F} most likely makes the oracle \mathbf{O} have collisions; thus, \mathbf{O} will not be one-to-one. In fact, the collisions of \mathbf{O} are planted in an adversarial way against the construction COM and that is why the distribution of \mathbf{O} depends on COM .⁷

It is easy to see that a partially-fixed random oracle is still hard to invert using poly(n)-query attacks. We show how to define the the distribution of \mathbf{O} such that, either of the binding or hiding properties of $\text{COM}^{\mathbf{O}}$ will be violated through a poly(n)-query attack. As we discussed above, this is sufficient for deriving a black-box separation. We prove the existence of such partially-fixed random oracle \mathbf{O} by proving that there are in fact *two* partially-fixed random oracles \mathbf{O}_R and \mathbf{O}_S such that *either* of the following holds:

1. The hiding of $\text{COM}^{\mathbf{O}_R}$ is broken by a poly(n)-query malicious receiver \hat{R} .
2. The binding of $\text{COM}^{\mathbf{O}_S}$ is broken by a poly(n)-query malicious sender \hat{S} .

Therefore, there always exists an oracle $\mathbf{O} \in \{\mathbf{O}_S, \mathbf{O}_R\}$ relative to which either of the hiding or binding properties of COM is broken by some $\text{ADV} \in \{\hat{R}, \hat{S}\}$.

⁶ For example, using standard tricks one can make the output of the random oracle long enough, say n^3 bits, while the input is only n bits. Such function is one-to-one with overwhelming probability.

⁷ As far as we know, this way of choosing the oracle's distribution based on the scheme itself was fist employed in the work of Gertner et al. [22].

The Cheating Strategies \hat{S}, \hat{R} . The cheating sender \hat{S} and the distribution of \mathbf{O}_S are defined assuming that \hat{R} fails in its attack, but that is still sufficient for us. The oracle \mathbf{O}_R is simply the random oracle, but the oracle \mathbf{O}_S will always be fixed over a polynomial-size domain (thus the final oracle $\mathbf{O} \in \{\mathbf{O}_R, \mathbf{O}_S\}$) will always be a partially-fixed random oracle. The algorithm of the malicious \hat{R} is in fact very simple: try to learn any query q that has a non-negligible chance of being asked by the sender during the generation of the commitment C , and after learning these queries make a guess about the committed bit b by outputting the more likely value of b conditioned on the knowledge learned about the random oracle \mathbf{O}_R . In the following we formally describe this algorithm and will show that if this algorithm fails in guessing the bit b correctly with probability $1/2 + 1/\text{poly}(n)$, then we can come up with a partially-fixed random oracle \mathbf{O}_S such that the binding of $\text{Com}^{\mathbf{O}_S}$ could be violated.

2.1 Technical Tool: Learning Heavy Queries

Suppose $\text{COM} = (S, R)$ is a non-interactive commitment scheme in a model where some randomized oracle \mathbf{O} (e.g., the random oracle) is accessed by the sender S and the receiver R and suppose S generates a commitment C to a random bit $b \xleftarrow{\$} \{0, 1\}$. Let \mathbf{S} be the view of the sender consisting its randomness as well as its oracle query-answers and \mathbf{R} be the view of the receiver after the verification of C which consists of C itself, the revealed bit b and some “decommitment” string D justifying the claim of S that he had committed to b . We can look at all of $\mathbf{S}, \mathbf{R}, C, b$, and D as random variables depending on the randomness of the parties and the randomness of \mathbf{O} . That is the case also for the set of queries $\mathcal{Q}(\mathbf{S}), \mathcal{Q}(\mathbf{R})$ asked by the sender and the receiver represented in their views.

Consider the following simple learning algorithm that upon receiving C , which is the commitment to a random $b \xleftarrow{\$} \{0, 1\}$, keeps updating a “learned” set of oracle query-answer pairs \mathcal{L} as follows: As long as there is an oracle query $q \notin \mathcal{L}$ which has ε probability to be asked by the sender during the generation of C or by the receiver during the verification of C :

$$\Pr[q \in \mathcal{Q}(\mathbf{S}) \cup \mathcal{Q}(\mathbf{R}) \mid C, \mathcal{L}] \geq \varepsilon,$$

then go ahead and ask q from the oracle. After asking q from \mathbf{O} , the pair $(q, \mathbf{O}(q))$ will be added to \mathcal{L} and the knowledge of $\mathbf{O}(q)$ will be incorporated in deciding which other queries might be likely as described above. A result due to [6] shows that such learning algorithm would (on average) ask at most $\text{poly}(n/\varepsilon) = \text{poly}(n)$ queries and reach a point that there is no “ ε -heavy” query left for the distribution of the views of the sender and the (honest) receiver conditioned on the learned information (C, \mathcal{L}) . As we will see, this learning algorithm will essentially form the cheating receiver’s algorithm \hat{R} .

2.2 Defining the Cheating Strategies

Suppose we execute the learning algorithm above when the randomized oracle \mathbf{O} in the scheme is simply a random oracle. We focus on the moment that

the learning algorithm stops (i.e., for any query $q \notin \mathcal{L}$ it holds that $\Pr[q \in \mathcal{Q}(S) \cup \mathcal{Q}(R) | C, \mathcal{L}] < \varepsilon]$), and divide possible the cases into two. In each case we show how to derive a cheating party and a corresponding randomized oracle.

Case 1. In the first case, with non-negligible probability $1/\text{poly}(n)$ over the executing of the learning algorithm, at the end there is a value $b \in \{0, 1\}$ such that $\Pr[b \text{ is the committed bit } | (C, \mathcal{L})] > 1/2 + 1/\text{poly}(n)$. In this case we can simply take \mathbf{O}_R to be the random oracle, and relative to \mathbf{O}_R the cheating strategy \hat{R} could just follow the learning algorithm above and output the more likely value of b conditioned on its view (C, \mathcal{L}) at the end. It is easy to see that this malicious receiver \hat{R} can guess the bit b with probability at least $1/2 + 1/\text{poly}(n)$.

Case 2. In the second case, at the end of the learning phase when there is no ε -heavy query left to be learned, with overwhelming probability: both of the values of $b \in \{0, 1\}$ are almost equally likely to be the committed bit conditioned on knowing (C, \mathcal{L}) . We will show that at this point there is always a way to fix a set of oracle query-answer pairs \mathcal{F} for some partially-fixed random oracle \mathbf{O}_S such that \hat{S} can successfully open the commitment C (which is the result of a single execution of the learning algorithm and is fixed forever) into both of $\{0, 1\}$.

Since we are in the case that conditioned on (C, \mathcal{L}) both values of $b \in \{0, 1\}$ have non-zero (in fact $\approx 1/2$) chance to be the committed bit, we can always sample two views $V_0 = (S_0, R_0), V_1 = (S_1, R_1)$ of full executions of the system for the sender and the receiver where they are both consistent with (C, \mathcal{L}) and V_b describes a case where C is a commitment to the bit b . Note that due to the (assumed) perfect completeness of the scheme, in both of the views V_0, V_1 the verification leads to an accept. We claim that if S_0 and S_1 are *consistent* over the query-answer pairs that they posses (i.e., use the same answer for the queries that they *both* have asked: $\mathcal{Q}(S_0) \cap \mathcal{Q}(S_1)$) then we are done, because we can take \mathcal{F} to be the answers to $\mathcal{Q}(S_0) \cup \mathcal{Q}(S_1)$ plus the query-answer pairs of \mathcal{L} and fix \mathcal{F} as part of the partially-fixed random oracle \mathbf{O}_S . This way, whenever the sender wants to decommit to the bit $b \in \{0, 1\}$ it can use the fixed view $S_b \in V_b$ for the needed decommitment, and he knows that such decommitment will always lead to the verification described by $R_b \in V_b$ (since the verification is deterministic) which is an accept.

It only remains to show how to find a pair of *consistent* views V_0, V_1 . Here we use the fact that conditioned on (C, \mathcal{L}) , both of $\{0, 1\}$ have chance $> 1/3$ to be the committed bit. Using a probabilistic analysis and also relying on the fact that there is no ε -heavy query left conditioned on (C, \mathcal{L}) (when the committed bit is considered random), and assuming that the total number of oracle queries of (S, R) is at most m , one can show that with probability $\geq 1 - 3m\varepsilon$ a pair of *random* samples V_0, V_1 , where V_b is sampled conditioned on (C, \mathcal{L}, b) , would have no query in common out of \mathcal{L} (i.e., $\mathcal{Q}(V_0) \cap \mathcal{Q}(V_1) \subseteq \mathcal{L}$). The reason is that for any query q which has probability at most ε to be in the queries of V , by conditioning on $b = 0$ or $b = 1$, this probability can increase at most to 3ε . Therefore, if we sample and fix V_0 , any of the m queries of the sampled V_0 would be sampled in V_1 only with probability at most 3ε . Thus, by a union bound,

with probability at least $1 - 3m\varepsilon$, none of the queries of V_0 will be sampled in V_1 . Since both of V_0, V_1 are sampled conditioned on (and consistent with) \mathcal{L} , we conclude that such samples are in fact consistent.

The Role of Non-Interactivity. Our argument above only applies to the non-interactive setting because of the way we constructed $(\widehat{S}, \mathbf{O}_S)$ in case \widehat{R} does not succeed. In particular, in the interactive setting C would be the transcript of an interactive protocol which could change every time that the protocol is executed, even if the sender commits to the same message using the same randomness, simply because the receiver's randomness might change every time. That should not be a surprise since Naor's commitment scheme [54] is a black-box construction based on one-way functions and has only two messages during the commitment phase (which perfectly complements our negative result of Theorem 1).

3 Separation from Hitting One-Way Functions

In this Section we outline the proof of Theorem 3. Before doing so we need to develop the notion of a hitting one-way function.

3.1 Hitting One-Way Functions

Hitting Set Generators. A (basic) *hitting set generator* G is an efficient deterministic procedure to generate sets that intersect any “dense” set recognized by an efficient circuit. More formally, given $n \geq m$, G runs in time $\text{poly}(n)$ and generates a set of m -bit strings \mathcal{H} such that for any circuit T accepting at least half of $\{0, 1\}^m$, it holds that $T(h) = 1$ for at least one $h \in \mathcal{H}$ (see [29] and references therein for more background on the subject). A hitting set generator G can be directly used to derandomize the complexity class RP and perhaps surprisingly even to derandomize the class BPP [3, 4]. Here we are interested in the notion of hitting set generators against co-nondeterministic circuits defined as follows.

Definition 6 (Co-Nondeterministic Circuits). A nondeterministic Boolean circuit T takes two inputs and accepts the set \mathcal{S}_T defined as follows $\mathcal{S}_T = \{x \mid \exists w, T(x, w) = 1\}$. A co-nondeterministic Boolean circuit T also takes two inputs and accepts the set $\mathcal{S}_T = \{x \mid \forall w, T(x, w) = 0\}$. By abusing the notation we call the first input simply the “input” and call the second input the “witness”. Thus, the input length refers to the length of x . If the input length is n , we call $d_T(n) = \frac{|\mathcal{S}_T \cap \{0, 1\}^n|}{2^n}$ the input density of T .

Now we introduce a new primitive that combines a one-way function and a hitting set generator against co-nondeterministic circuits.

Definition 7 (Hitting One-Way Functions). We say a function $f: \{0, 1\}^n \mapsto \{0, 1\}^n$ hits a co-nondeterministic circuit T of size n and input length m if it holds that $\{f(1)|_m, \dots, f(n^2)|_m\} \cap \mathcal{S}_T \neq \emptyset$ where $1, 2, \dots, n^2$ are analogs of the first n^2 elements of $\{0, 1\}^n$ and $y|_m$ refers to the first m bits of y . We say that

a sequence of functions $\{f_n : \{0, 1\}^n \mapsto \{0, 1\}^n\}$ is a hitting function, if f_n hits every circuit T of size n and input density $d_T \geq 1/2$ for large enough n . A length preserving function family $f = \{f_n : \{0, 1\}^n \mapsto \{0, 1\}^n\}$ is simply called a hitting one-way function, if it is both hitting and one-way simultaneously.

As we will see later, a random oracle is a hitting one-way function with overwhelming probability, and thus being hitting one-way could be thought of as a natural abstracted property of a random oracle (similar to e.g., collision resistance). Moreover, it is easy to see (details in the full version) that hitting one-wayness can be formalized using a standard cryptographic security game, and as such, the assumption that a function f is a hitting one-way function is a falsifiable assumption, in the terminology of Naor [53].⁸

Black-Box Constructions from Hitting One-Way Functions. We skip a separate definition for black-box constructions based on hitting one-way functions, since this definition could be obtained from Definition 4 and Definition 7. Namely, given any oracle adversary that breaks the security of COM^f , the security reduction $\text{SEC}^{f,\text{ADV}}$, with non-negligible probability shall break the hitting one-way property of the oracle f (by either inverting f , or finding a circuit T of input density $d_T \geq 1/2$ that is not hit by f together with a witness of such claim).

3.2 Outline of the Proof of Theorem 3

In order to prove Theorem 3, we rely on the proof of Theorem 1 outlined in Section 2. A natural approach would be to show that our partially-fixed random oracle \mathbf{O} is already a hitting one-way function with overwhelming probability. Doing so would prove Theorem 3 as a direct extension of the proof of Theorem 1, however, the problem with this approach is that a partially-fixed random function \mathbf{f} , in general, might not be a hitting function, simply because the fixed part of the randomized function \mathbf{f} could be adversarially chosen to make it not hit a particular circuit T . However, recall that our oracle \mathbf{O}_R relative to which the cheating receiver \widehat{R} was successful, was indeed the random oracle. So in the following we start by handling the case that \widehat{R} was a successful cheating receiver.

Case 1: The cheating receiver \widehat{R} succeeds relative to a random oracle. It is easy to see that a random oracle is one-way with high probability.⁹ We first show that a random oracle is also a hitting function with overwhelming probability (and so it will be hitting one-way).

⁸ A subtle point here is that the hitting property is defined w.r.t. co-nondeterministic (as opposed to nondeterministic) circuits. Thus when f is not hitting, there always exists a polynomial-size witness for that: a circuit T of size n and input length m and a sequence w_1, \dots, w_{n^2} such that $T(f(i)|_m, w_i) = 1$ for all $i \in [n^2] \subset \{0, 1\}^n$.

⁹ Recall that our random oracle chooses its randomness after the adversary is fixed and is different from the settings of [20, 44] who fix the random oracle after sampling it once and for all.

Lemma 8. *For every $n \in \mathbb{N}$, with probability at least $1 - 2^{-n^2(1-o(1))}$ a random function $\mathbf{f}: \{0,1\}^n \mapsto \{0,1\}^n$ hits all co-nondeterministic circuits of size n and input density $d_T \geq 1/2$.*

Proof. Fix any co-nondeterministic circuit T of size n and input density $d_T \geq 1/2$. Any of the random images of $\mathbf{f}(j)$ for $j \in [n^2] \subseteq \{0,1\}^n$ (when truncated to the right size) will hit an element in \mathcal{S}_T with probability at least the input density of T which is $d_T \geq 1/2$. Therefore, the probability that none of $\{f(1), \dots, f(n^2)\}$ hits \mathcal{S}_T is at most 2^{-n^2} . Since the total number of circuits of size¹⁰ n is at most $2^{O(n \log n)}$, the lemma follows by a union bound.

Lemma 8 implies that for large enough n a random function from $\{0,1\}^n$ to $\{0,1\}^n$ is hitting with overwhelming probability. Therefore Lemma 8 is sufficient for refuting the existence of black-box constructions of non-interactive commitments from hitting one-way functions. Namely, for large enough n , with overwhelming probability, there exists no circuit T of size n that the security reduction SEC (of any potential black-box construction COM) can output to refute the hitting property of f . Therefore, in this case the security reduction SEC might as well just try to invert the random oracle (with the help of the adversary). This means that, if we are in Case 1 (where \mathbf{O}_R is the random oracle), we can safely assume that we are back to the setting of Theorem 1 where the security reduction only tries to invert f , but we have already settled this case!

Generalization. The argument above can be generalized to any black-box separation result that is established through an attack in the *random-oracle model* to also handle primitives that in addition are hitting (e.g., hitting one-way functions, hitting collision resistant hash functions, etc). Thus, the result of [44] can be extended to separate key-agreement from hitting one-way functions.

Case 2: The cheating receiver \hat{R} fails relative to a random oracle. In this case, we would like to follow the general structure of Case 2 in Section 2, but as we mentioned before the issue is that the partially-fixed randomized oracle \mathbf{O}_S might not be a hitting function. However, recall that the fixed part of \mathbf{O}_S was due to the learned set \mathcal{L} and the query-answer pairs inside the two *randomly* sampled views V_0 and V_1 . Therefore, even though we fixed the sampled part of the oracle inside $(\mathcal{L}, Q(V_0), Q(V_1))$ and relied on the remaining randomness of \mathbf{O}_S to conclude that \mathbf{O}_S is one-way, this fixed part was also generated through a randomized process (even though it was fixed after being sampled). This lets us to still have a hope that the whole random process of generating \mathbf{O}_S (also over the randomness of generating the fixed part at the beginning) makes the final result a hitting one-way function with overwhelming probability.

Recall that the two sampled views V_0, V_1 were obtained conditioned on (C, \mathcal{L}) , and) the committed bit to be 0 and 1. Now suppose instead of such samples we would have sampled only one view V (for the sender and the receiver) conditioned on the values of (C, \mathcal{L}) but *without* conditioning the committed bit b to be 0 or

¹⁰ Here we denote the size of a circuit by the number of its wires.

1. Then, since C was already the commitment to a random bit b , V would be a sample from the real distribution of the views of the sender and the receiver conditioned on (C, \mathcal{L}) . Therefore, the joint samples (C, \mathcal{L}, V) together have the same marginal distribution as (C, \mathcal{L}, V') where V' is the *true* view of the parties. Therefore we can conclude the following crucial property of our sampling process: If we first sample (C, \mathcal{L}, V) to get a partial oracle over $\mathcal{F} = (\mathcal{L}, \mathcal{Q}(V))$ and then choose the oracle answers to any query out of \mathcal{F} at random, the final result is a random oracle. The reason simply is that this property holds for (C, \mathcal{L}, V') which has the same marginal distribution as that of (C, \mathcal{L}, V) ; so the same should hold for (C, \mathcal{L}, V) as well! We call such randomized *partial* functions (which are not defined over some of their inputs) *partially-defined* random functions.

Definition 9 (Partially-Defined Random Functions). Suppose \mathbf{f} is a random variable whose output is a partial function from $\{0, 1\}^n$ to $\{0, 1\}^n$ (therefore, a sample $f \leftarrow \mathbf{f}$ might be defined only over a subset of its domain $\{0, 1\}^n$). Define the randomized total function $\tilde{\mathbf{f}}$ over the domain $\{0, 1\}^n$ (as the random extension of \mathbf{f}) as follows: First sample $f \xleftarrow{s} \mathbf{f}$. Then for every point $x \in \{0, 1\}^n$ which is not among the queries answered by \mathbf{f} choose a random answer from $\{0, 1\}^n$. Call the resulting function $\tilde{\mathbf{f}}$ (and its random variable $\tilde{\mathbf{f}}$). If the randomized function $\tilde{\mathbf{f}}$ is distributed exactly the same as a uniformly sampled function from $\{0, 1\}^n$ to $\{0, 1\}^n$, then we call \mathbf{f} a partially-defined random function.

The New Definition of \mathbf{O}_S . The fact that a random extension of the randomized partial function described in (C, \mathcal{L}, V) is a random oracle indicates that our randomized oracle \mathbf{O}_S which was generated through two sampled views V_0, V_1 might have similar properties and be a hitting one-way function. With this intuition in mind, we change the distribution of the randomized oracle \mathbf{O}_S as follows: The two sampled views V_0 and V_1 are sampled independently conditioned on (C, \mathcal{L}) without conditioning on the bit b to be 0 or 1 (just like the way V was sampled). The final (new) definition of the randomized oracle \mathbf{O}_S is as follows. We first sample $(C, \mathcal{L}, V_0, V_1)$ as above to randomly sample a partial oracle \mathbf{f} , and then randomly extend it to a full oracle $\tilde{\mathbf{f}} \equiv \mathbf{O}_S$ according to Definition 9. Since we would like to avoid *rejection-sampling* (not to change the marginal distributions of (C, \mathcal{L}, V_0) and (C, \mathcal{L}, V_1)) if the sampled views V_0, V_1 had contradicting answers for any oracle query q we choose the answer provided by one of V_0, V_1 at random. In the following we will show that a cheating sender \hat{S} still exists relative to \mathbf{O}_S , and that relative to \hat{S} , \mathbf{O}_S remains one-way and hitting.

Concluding Theorem 3. The following three propositions imply Theorem 3.

Proposition 10. If \hat{R} does not break the hiding of $\text{COM}^{\mathbf{O}_R}$, then there exists a malicious sender \hat{S} that breaks the binding of $\text{COM}^{\mathbf{O}_S}$.

Proof. Since we are in the case that the cheating receiver \hat{R} is not successful, thus the distribution of the bit b conditioned on (C, \mathcal{L}) remains close to uniform over $\{0, 1\}$, which means that in our new way of sampling (V_0, V_1) , still with

probability polynomially close to $1/2$ (and so bigger than, say, $1/3$) we get that V_0 (resp. V_1) corresponds to the bit $b = 0$ (resp. $b = 1$) used as the committed bit by the sender. Therefore by choosing the fixed part of \mathbf{O}_S based on the sampled answers of $(\mathcal{L}, \mathcal{Q}(V_0), \mathcal{Q}(V_1))$, with $\Omega(1)$ probability we get a cheating sender \hat{S} who is able to successfully decommit to both values of the bit b using the fixed sampled view V_b .

Proposition 11 (\mathbf{O}_S Remains One-Way). *No $\text{poly}(n)$ -query adversary ADV can invert $\mathbf{O}_S(\mathbf{U}_n)$ with probability $1/\text{poly}(n)$, even when ADV is given oracle access to the cheating sender \hat{S} .*

Proof. Any query out of $\mathcal{L} \cup \mathcal{Q}(V_0) \cup \mathcal{Q}(V_1)$ is answered at random and the cheating sender \hat{S} is defined solely based on (\mathcal{L}, V_0, V_1) .

The main technical meat of the proof of Theorem 3 is found in the following proposition. Due to lack of space, we only provide a very high-level outline.

Proposition 12. *\mathbf{O}_S is hitting with overwhelming probability.*

Proof (Outline). We would like to show that when one evaluates the oracle \mathbf{O}_S over $[n^2]$ it will hit at least one of the accepted inputs of any (co-nondeterministic) circuit T of input density $d_T \geq 1/2$ with “high” probability ρ . We want the probability ρ to be extremely close to one so that we can change the order of quantifiers and conclude that \mathbf{O}_S hits all circuits of size n .

Recall that each of the sampled partial oracles $\mathbf{f}_0, \mathbf{f}_1$ described by the query-answer pairs in (\mathcal{L}, V_0) and (\mathcal{L}, V_1) is a partially-defined random oracle, and that the final oracle \mathbf{O}_S is a random extension of the “combination” of \mathbf{f}_0 and \mathbf{f}_1 (that combines the answers of f_0 and f_1 whenever their sets of queries out of \mathcal{L} do not collide). The intuition is that now, over the domain $[n^2]$ (planted at the beginning of $\{0, 1\}^n$) at least half of the queries are answered randomly and independently and would behave like a random function because they either come from \mathbf{f}_0 , or \mathbf{f}_1 , or from the final random extension of $(\mathbf{f}_0, \mathbf{f}_1)$ to the full domain of $\{0, 1\}^n$ which we denote by \mathbf{f}' (and is chosen independently of $(\mathbf{f}_0, \mathbf{f}_1)$). More formally, since \mathbf{f}' is chosen independently of $(\mathbf{f}_0, \mathbf{f}_1)$, both of $(\mathbf{f}_0, \mathbf{f}')$ and $(\mathbf{f}_1, \mathbf{f}')$ are also partially-defined random oracles. Moreover, we know that over the domain $[n^2]$, at least half of the queries are answered either by $(\mathbf{f}_0, \mathbf{f}')$ or by $(\mathbf{f}_1, \mathbf{f}')$. We would like to use this property to conclude that \mathbf{O}_S hits every circuit with high probability.

Formalizing this intuition, however, is far from easy and the challenge stems from the fact that, as we said before, we want the probability ρ to be extremely close to one. Because of this, we can not afford the $1/\text{poly}(n)$ probability that queries the sampled views V_0 and V_1 might have collisions (out of \mathcal{L}) and resample \mathbf{O}_S again. Therefore, we define the oracle \mathbf{O}_S in a randomized way, even when such collisions happen. To prove that the sample oracle \mathbf{O}_S is hitting with very high probability, we develop and employ new concentration bounds to control the probability that \mathbf{O}_S is not hitting. We refer the reader for the full version of the paper for the full proof.

Acknowledgment. We thank the anonymous referees of Crypto 2012 for their valuable comments. In particular, we thank Marc Fislin for a thorough reading of the paper and valuable comments. We thank Noga Alon for pointing out the anti-concentration bound of Lemma A.2.2 in [2] to us (used in the full proof of Proposition 12). Finally we thank Yuval Ishai for encouraging us to work on the question of separating the power of black-box versus non-back-box cryptographic constructions.

References

1. Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P. Report, Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur-208016, India (August 2002)
2. Alon, N., Spencer, J.H.: The probabilistic method, 3rd edn. Wiley, New York (2008)
3. Andreev, A.E., Clementi, A.E.F., Rolim, J.D.P.: A new general derandomization method. *JACM: Journal of the ACM* 45 (1998)
4. Andreev, A.E., Clementi, A.E.F., Rolim, J.D.P., Trevisan, L.: Weak random sources, hitting sets, and BPP simulations. *SICOMP: SIAM Journal on Computing* 28 (1999)
5. Barak, B.: How to go beyond the black-box simulation barrier. In: Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS), pp. 106–115 (2001)
6. Barak, B., Mahmoody, M.: Lower bounds on signatures from symmetric primitives. In: FOCS: IEEE Symposium on Foundations of Computer Science (2007)
7. Barak, B., Ong, S.J., Vadhan, S.: Derandomization in Cryptography. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 299–315. Springer, Heidelberg (2003)
8. Blum, Impagliazzo: Generic oracles and oracle classes. In: FOCS: IEEE Symposium on Foundations of Computer Science (1987)
9. Blum, M.: Coin flipping by telephone. In: CRYPTO, pp. 11–15 (1981)
10. Blum, M., Kannan, S.: Designing programs that check their work. *J. ACM* 42(1), 269–291 (1995)
11. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo random bits, pp. 112–117 (1982)
12. Boneh, Papakonstantinou, Rackoff, Vahlis, Waters: On the impossibility of basing identity based encryption on trapdoor permutations. In: FOCS: IEEE Symposium on Foundations of Computer Science (2008)
13. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences* 37(2), 156–189 (1988)
14. Brassard, G., Crépeau, C., Yung, M.: Constant-round perfect zero-knowledge computationally convincing protocols. *Theoretical Computer Science* 84(1), 23–52 (1991)
15. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-Box Construction of a Non-malleable Encryption Scheme from Any Semantically Secure One. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 427–444. Springer, Heidelberg (2008)
16. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
17. Dachman-Soled, D., Lindell, Y., Mahmoody, M., Malkin, T.: On the Black-Box Complexity of Optimally-Fair Coin Tossing. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 450–467. Springer, Heidelberg (2011)

18. Damgård, I.B., Pedersen, T.P., Pfitzmann, B.: Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory* 44(3), 1143–1151 (1998)
19. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. *SIAM Journal on Computing* 35(1), 217–246 (2005)
20. Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pp. 305–313 (2000)
21. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science* (2000)
22. Gertner, Y., Malkin, T., Reingold, O.: On the impossibility of basing trapdoor functions on trapdoor predicates. In: *FOCS*, pp. 126–135 (2001)
23. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology* 9(3), 167–190 (1996)
24. Goldreich, O., Krawczyk, H.: Sparse pseudorandom distributions. *Random Structures & Algorithms* 3(2), 163–174 (1992)
25. Goldreich, O., Krawczyk, H., Luby, M.: On the existence of pseudorandom generators. *SIAM Journal on Computing* 22(6), 1163–1175 (1993)
26. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 25–32 (1989)
27. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority, pp. 218–229 (1987)
28. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM* 38(1), 691–729 (1991); Preliminary version in *FOCS* 1986
29. Goldreich, O., Wigderson, A.: Improved Derandomization of BPP Using a Hitting Set Generator. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) *RANDOM-APPROX* 1999. LNCS, vol. 1671, pp. 131–137. Springer, Heidelberg (1999)
30. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM Journal on Computing* 18(1), 186–208 (1989); Preliminary version in *STOC* 1985
31. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing* 17(2), 281–308 (1988); Preliminary version in *FOCS* 1984
32. Goyal, V.: Constant round non-malleable protocols using one way functions (2011)
33. Haitner, I.: Semi-honest to Malicious Oblivious Transfer—The Black-Box Way. In: Canetti, R. (ed.) *TCC* 2008. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (2008)
34. Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols – A tight lower bound on the round complexity of statistically-hiding commitments. In: *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society (2007)
35. Haitner, I., Horvitz, O., Katz, J., Koo, C.-Y., Morselli, R., Shaltiel, R.: Reducing Complexity Assumptions for Statistically-Hiding Commitment. In: Cramer, R. (ed.) *EUROCRYPT* 2005. LNCS, vol. 3494, pp. 58–77. Springer, Heidelberg (2005), See also preliminary draft of full version www.wisdom.weizmann.ac.il/~iftachh/papers/SCfromRegularOWF.pdf
36. Haitner, I., Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions of protocols for secure computation. *SIAM J. Comput.* 40(2), 225–266 (2011)

37. Haitner, I., Nguyen, M.-H., Ong, S.J., Reingold, O., Vadhan, S.: Statistically-hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing* (November 2007)
38. Haitner, I., Omri, E.: Coin flipping with constant bias implies one-way functions (2011)
39. Haitner, I., Reingold, O.: Statistically-hiding commitment from any one-way function. In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press (2007)
40. Haitner, I., Reingold, O., Vadhan, S.P., Wee, H.: Inaccessible entropy (2009)
41. Hartmanis, J., Hemachandra, L.A.: One-way functions, robustness, and the non-isomorphism of *NP*-complete sets. Technical Report 86-796, Department of Computer Science, Cornell University (January 1987)
42. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. *SIAM Journal on Computing* 28(4), 1364–1396 (1999); Preliminary versions in STOC 1989 and STOC 1990
43. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography. In: *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 230–235 (1989)
44. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 44–61. ACM Press (1989)
45. Kahn, J., Saks, M., Smyth, C.: A dual version of Reimer’s inequality and a proof of Rudich’s conjecture. In: *15th Annual IEEE Conference on Computational Complexity*, pp. 98–103 (2000)
46. Katz, J., Schröder, D., Yerukhimovich, A.: Impossibility of Blind Signatures from One-Way Permutations. In: Ishai, Y. (ed.) *TCC 2011. LNCS*, vol. 6597, pp. 615–629. Springer, Heidelberg (2011)
47. Kim, J.H., Simon, D.R., Tetali, P.: Limits on the efficiency of one-way permutation-based hash functions. In: *FOCS*, pp. 535–542 (1999)
48. Lenstra, A.K., Lenstra Jr., H.W. (eds.): *The development of the number field sieve*. Lecture Notes in Mathematics, vol. 1554. Springer, Berlin (1993)
49. Levin, L.A.: One-way functions and pseudorandom generators. *Combinatorica* 7, 357–363 (1987)
50. Lin, H., Trevisan, L., Wee, H.: On Hardness Amplification of One-Way Functions. In: Kilian, J. (ed.) *TCC 2005. LNCS*, vol. 3378, pp. 34–49. Springer, Heidelberg (2005)
51. Matsuda, T., Matsuura, K.: On Black-Box Separations among Injective One-Way Functions. In: Ishai, Y. (ed.) *TCC 2011. LNCS*, vol. 6597, pp. 597–614. Springer, Heidelberg (2011)
52. Miller, G.L.: Riemann’s hypothesis and tests for primality. *Journal of Computer and System Sciences* 13(3), 300–317 (1976)
53. Naor, M.: On Cryptographic Assumptions and Challenges. In: Boneh, D. (ed.) *CRYPTO 2003. LNCS*, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
54. Naor, M.: Bit commitment using pseudorandomness. *Journal of Cryptology* 4(2), 151–158 (1991); Preliminary version In: Brassard, G. (ed.) *CRYPTO 1989. LNCS*, vol. 435, pp. 128–136. Springer, Heidelberg (1990)
55. Naor, M., Ostrovsky, R., Venkatesan, R., Yung, M.: Perfect zero-knowledge arguments for NP using any one-way permutation. *CRYPTO 1992* 11(2), 87–108 (1998); Preliminary version in Brickell, E.F. (ed.): *CRYPTO 1992. LNCS*, vol. 740. Springer, Heidelberg (1993)

56. Nguyen, M.-H., Ong, S.J., Vadhan, S.: Statistical zero-knowledge arguments for NP from any one-way function. In: Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS), pp. 3–14 (2006)
57. Ostrovsky, R., Wigderson, A.: One-way functions are essential for non-trivial zero-knowledge. In: ISTCS, pp. 3–17 (1993)
58. Pass, R., Wee, H.: Black-Box Constructions of Two-Party Protocols from One-Way Functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009)
59. Rabin, M.O.: Probabilistic algorithm for testing primality. Journal of Number Theory 12(1), 128–138 (1980)
60. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of Reducibility between Cryptographic Primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
61. Rudich, S.: Limits on the Provable Consequences of One-Way Functions. PhD thesis, U.C. Berkeley (1988)
62. Simon, D.R.: Findings Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
63. Tardos, G.: Query complexity, or why is it difficult to separate NP^A from $\text{NP}^{\text{co-NP}}$ by random oracles A? Combinatorica 9(4), 385–392 (1989)
64. Vahlis, Y.: Two Is a Crowd? A Black-Box Separation of One-Wayness and Security under Correlated Inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 165–182. Springer, Heidelberg (2010)
65. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: FOCS, pp. 531–540. IEEE Computer Society (2010)
66. Yao, A.C.: Theory and applications of trapdoor functions, pp. 80–91 (1982)

Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems

Itai Dinur¹, Orr Dunkelman^{1,2}, Nathan Keller^{1,3}, and Adi Shamir¹

¹ Computer Science department, The Weizmann Institute, Rehovot, Israel

² Computer Science Department, University of Haifa, Israel

³ Department of Mathematics, Bar-Ilan University, Israel

Abstract. In this paper we show that a large class of diverse problems have a bicomposite structure which makes it possible to solve them with a new type of algorithm called *dissection*, which has much better time/memory tradeoffs than previously known algorithms. A typical example is the problem of finding the key of multiple encryption schemes with r independent n -bit keys. All the previous error-free attacks required time T and memory M satisfying $TM = 2^{rn}$, and even if “false negatives” are allowed, no attack could achieve $TM < 2^{3rn/4}$. Our new technique yields the first algorithm which never errs and finds all the possible keys with a smaller product of TM , such as $T = 2^{4n}$ time and $M = 2^n$ memory for breaking the sequential execution of $r = 7$ block ciphers. The improvement ratio we obtain increases in an unbounded way as r increases, and if we allow algorithms which can sometimes miss solutions, we can get even better tradeoffs by combining our dissection technique with parallel collision search. To demonstrate the generality of the new dissection technique, we show how to use it in a generic way in order to attack hash functions with a rebound attack, to solve hard knapsack problems, and to find the shortest solution to a generalized version of Rubik’s cube with better time complexities (for small memory complexities) than the best previously known algorithms.

Keywords: Cryptanalysis, TM-tradeoff, multi-encryption, knapsacks, bicomposite, dissection, rebound.

1 Introduction

A composite problem is a problem that can be split into several simpler subproblems which can be solved independently of each other. To prevent attacks based on such decompositions, designers of cryptographic schemes usually try to entangle the various parts of the scheme by using a complex key schedule in block ciphers, or a strong message expansion in hash functions. While we can formally split such a structure into a top part that processes the input and a bottom part that produces the output, we cannot solve these subproblems independently of each other due to their strong interactions.

However, when we deal with higher level constructions which combine multiple primitives as black boxes, we often encounter unrelated keys or independently computed outputs which can provide exploitable decompositions. One of the best examples of such a situation was the surprising discovery by Joux [9] in 2004 that finding collisions in hash functions defined by the *parallel execution* of several independent iterated hash functions is much easier than previously believed. In this paper we show the dual result that finding the key of a multiple-encryption scheme defined by the *sequential execution* of several independent cryptosystems is also easier than previously believed.

Since we can usually reduce the time complexity of cryptanalytic attacks by increasing their memory complexity, we will be interested in the full tradeoff curve between these two complexities rather than in a single point on it. We will be primarily interested in algorithms which use an exponential combination of $M = 2^{mn}$ memory and $T = 2^{tn}$ time for a small constant m and a larger constant t , when the key size n grows to infinity. While this setup may sound superficially similar to Hellman's time/memory tradeoff algorithms, it is important to notice that Hellman's preprocessing phase requires time which is equivalent to exhaustive search and memory which is at least the square root of the number of keys, and that in Hellman's online phase the product of time and memory is larger than the number of keys. In our model we do not allow free preprocessing, we can use smaller amounts of memory, and the product of time and memory is strictly smaller than the number of keys.

The type of problems we can solve with our new techniques is characterized by the existence of two orthogonal ways in which we can decompose a given problem into (almost) independent parts. We call such problems *bicomposite*, and demonstrate this notion by considering the problem of cryptanalyzing the sequential execution of r block ciphers which use independent n -bit keys to process n -bit plaintexts. In order to make the full rn -bit key of this scheme unique with a reasonable probability, the cryptanalyst needs r known plaintext/ciphertext pairs. The full encryption process can thus be described by an $r \times r$ matrix whose columns corresponds to the processing of the various plaintexts and whose rows correspond to the application of the various block ciphers. The attacker is given the r plaintexts at the top and the r ciphertexts at the bottom, and his goal is to find all the keys with a generic algorithm which does not assume the existence of any weaknesses in the underlying block ciphers. The reason we say that this problem is bicomposite is that the keys are independently chosen and the plaintexts are independently processed, and thus we can partition the execution matrix both horizontally and vertically into independent parts. In particular, if we know certain subsets of keys and certain subsets of intermediate values, we can independently verify their consistency with the given plaintexts or ciphertexts without knowing all the other values in the execution matrix. This should be contrasted with the standard constructions of iterated block ciphers, in which a partial guess of the key and a partial guess of some state bits in the middle of the encryption process cannot be independently verified by an efficient computation.

The security of multiple-encryption schemes had been analyzed for more than 30 years, but most of the published papers had dealt with either double or triple encryption (which is widely used as a DES-extension in the banking industry). While the exact security of double and triple encryption are well understood and we can not push their analysis any further, our new techniques show that surprisingly efficient attacks can be applied already when we make the next step and consider quadruple encryption, and that additional improvements can be made when we consider even longer combinations.

Standard meet-in-the-middle (MITM) attacks, which account for the best known results against double and triple encryption, try to split such an execution matrix into a top part and a bottom part with a single horizontal partition line which crosses the whole matrix from left to right. Our new techniques use a more complicated way to split the matrix into independent parts by exploiting its two dimensional structure. Consider, for example, the sequential execution of 7 independent block ciphers. We can find the full $7n$ -bit key in just 2^{4n} time and 2^n memory by guessing two of the seven internal states after the application of the third block cipher and one of the seven internal states after the application of the fifth block cipher. We call such an irregular way to partition the execution matrix with partial guesses a *dissection*, since it mimics the way a surgeon operates on a patient by using multiple cuts of various lengths at various locations.

Our new techniques make almost no assumptions about the internal structure of the primitive operations, and in particular they can be extended with just a slight loss of efficiency to primitive operations which are one-way functions rather than easily invertible permutations. This makes it possible to find improved attacks on message authentication codes (MACs) which are defined by the sequential execution of several keyed hash functions. Note that standard MITM attacks cannot be applied in this case, since we have to encrypt the inputs and decrypt the outputs in order to compare the results in the middle of the computation.

To demonstrate the generality of our techniques, we show in this paper how to apply them to several types of combinatorial search problems (some of which have nothing to do with cryptography), such as the knapsack problem: Given n generators a_1, a_2, \dots, a_n which are n -bit numbers, find a subset that sums modulo 2^n to S . The best known special purpose algorithm for this problem was published at Eurocrypt 2011 by Becker et al. [1], but our generic dissection technique provides better time complexities for small memory complexities. To show the connection between knapsack problems and multiple-encryption, describe the solution of the given knapsack problem as a two dimensional $r \times r$ execution matrix, in which we partition the generators into r groups of n/r generators, and partition each number into r blocks of n/r consecutive bits. Each row in the matrix is defined by adding the appropriate subset of generators from the next group to the accumulated sum computed in the previous row. We start with an initial value of zero, and our problem is to find some execution that leads to a desired value S after the last row. This representation is bicomposite since the choices made in the various rows of this matrix are completely independent,

and the computations made in the various columns of this matrix are almost independent since the only way they interact with each other is via the addition carries which do not tend to propagate very far into the next block. This makes it possible to guess and operate on partial states, and thus we can apply almost the same dissection technique we used for multiple-encryption schemes. Note that unlike the case of multiple-encryption in which the value of r was specified as part of the given problem, here we can choose any desired value of r independently of the given value of n in order to optimize the time complexity for any available amount of memory. In particular, by choosing $r = 7$ we can reduce the best known time complexity for hard knapsacks when we use $M = 2^{n/7} = 2^{0.1428n}$ memory from $2^{(3/4-1/7)n} = 2^{0.6071n}$ in [1] to $2^{4n/7} = 2^{0.5714n}$ with our new algorithm.

The algorithm of Becker et al. [1] crucially depends on the fact that addition is an associative and commutative operation on numbers, and that sets can be partitioned into the union of two subsets in an exponential number of ways. Our algorithms make no such assumptions, and thus they can be applied under a much broader set of circumstances. For example, consider a non-commutative variant of the knapsack problem in which the generators a_i are permutations over $\{1, 2, \dots, k\}$, and we have to find a product of length ℓ of these generators which is equal to some given permutation S (a special case of this variant is the problem of finding the fastest way to solve a given state of Rubik's cube [11] by a sequence of face rotations, which was analyzed extensively in the literature). To show that this problem is bicomposite, we have to represent it by an execution matrix with independent rows and columns. Consider an $\ell \times k$ matrix in which the i -th row represents the action of the i -th permutation in the product, and the j -th column represents the current location of element j from the set. Our goal is to start from the identity permutation at the top, and end with the desired permutation S at the bottom. We can reduce this matrix to size $r \times r$ for any desired r by bunching together several permutations in the product and several elements from the set. The independence of the rows in this matrix follows from the fact that we can freely choose the next generators to apply to the current state, and the independence of the columns follows from the fact that we can know the new location of each element j if we know its previous location and which permutation was applied to the state, even when we know nothing about the locations of the other elements in the previous state. This makes it possible to guess partial states at intermediate stages, and thus to apply the same dissection algorithms as in the knapsack problem with the same improved complexities.

We note that generic ideas similar to the basic dissection attacks were used before, in the context of several specific bicomposite problems. These include the algorithms of Schroeppel and Shamir [17] and of Becker et al. [1] which analyzed the knapsack problem, the algorithm of van Oorschot and Wiener [18] which attacked double and triple encryption, and the results of Dinur et al. [3] in the specific case of the block cipher GOST. A common feature of all these algorithms is that none of them could beat the tradeoff curve $TM = N^{3/4}$, where N is the total number of keys. The algorithms of [3,17,18] matched this curve

only for a single point, and the recent algorithm of Becker et al. [1] managed to match it for a significant portion of the tradeoff curve. Our new dissection algorithms not only allow to beat this curve, but actually allow to obtain the relation $TM < N^{3/4}$ for any amount of memory in the range $M \leq N^{1/4}$.

The paper is organized as follows: In Section 3 we introduce the dissection technique and present our best error-free attacks on multiple encryption. In Section 4 we consider the model when “false negatives” are allowed, and show that the dissection algorithms can be combined with the parallel collision algorithm of van Oorschot and Wiener [18] to get an improved time-memory tradeoff curve. Finally, in Section 5 we apply our results to various problems, including knapsacks, rebound attacks on hash functions and search problems in databases.

2 Notations and Conventions

In this paper we denote the basic block cipher by E and assume that it uses n -bit blocks and n -bit keys (we can easily deal with other sizes, but it makes the notation cumbersome). We denote by E^i the encryption process with key k_i , and denote by $E^{[1\dots r]}$ the multiple-encryption scheme which uses r independent keys to encrypt the plaintext P and produce the ciphertext C via $C = E_{k_r}(E_{k_{r-1}}(\dots E_{k_2}(E_{k_1}(P))\dots))$. The intermediate value produced by the encryption of P under $E^{[1\dots i]}$ is denoted by X^i , and the decryption process of $E^{[1\dots r]}$ is denoted by $D^{[1\dots r]}$ (which applies the keys in the reverse order). To attack $E^{[1\dots r]}$, we are usually given r plaintext/ciphertext pairs, which are expected to make the key unique (at intermediate stages, we may be given fewer than $j - i + 1$ plaintext/ciphertext pairs for $E^{[i\dots j]}$, and then we are expected to produce all the compatible keys). In all our exponential complexity estimates, we consider expected rather than maximal possible values (under standard randomness assumptions, they differ by no more than a logarithmic factor), and ignore multiplicative polynomial factors in n and r .

3 Dissecting the Multiple-Encryption Problem

In this section we develop our basic dissection algorithms that allow to solve efficiently the problem of multiple encryption. Given r -encryption with r independent keys, r n -bit plaintext/ciphertext pairs and 2^{mn} memory cells, the algorithms find all possible values of the keys which comply with the plaintext/ciphertext pairs, or prove that there are no such keys. The algorithms are deterministic, in the sense that they do not need random bits and they always succeed since they implicitly scan all possible solutions.

Here, we treat the case of general r and $m = 1$. The generalization of the algorithms to other integer values of m is given in the extended version of this paper [4]. The algorithms can be extended also to fractional values of m and to compositions of one-way functions, which appear in the context of layered Message Authentication Codes, such as NMAC [2].¹ The first non-integer extension

¹ Given a keyed hash-function F , and a key $k = (k_1, k_2)$, the MAC function $NMAC(x)$ which works on inputs x of arbitrary length, is defined as $NMAC_k(x) = F_{k_1}(F_{k_2}(x))$.

is presented in the full version of the paper, whereas the extension to one-way functions is presented in the extended version of this paper [4].

3.1 Previous Work – The Meet in the Middle Attack

The trivial algorithm for recovering the key of r -encryption is exhaustive search over the 2^{rn} possible key values, whose time complexity is 2^{rn} , and whose memory requirement is negligible. In general, with no additional assumptions on the algorithm and on the subkeys, this is the best possible algorithm.

In [14] Merkle and Hellman observed that if the keys used in the encryption are independent, an adversary can trade time and memory complexities, using a meet in the middle approach. In this attack, the adversary chooses a value u , $1 \leq u \leq \lfloor r/2 \rfloor$, and for each possible combination of the first u keys (k_1, k_2, \dots, k_u) she computes the vector $(X_1^u, X_2^u, \dots, X_r^u) = E^{[1\dots u]}(P_1, P_2, \dots, P_r)$ and stores it in a table (along with the respective key candidate). Then, for each value of the last $r - u$ keys, the adversary computes the vector $D^{[u+1\dots r]}(C_1, C_2, \dots, C_r)$ and checks whether the value appears in the table (each such collision suggests a key candidate (k_1, \dots, k_r)). The right key is necessarily suggested by this approach, and in cases when other keys are suggested, additional plaintext/ciphertext pairs can be used to sieve the wrong key candidates.

The time complexity of this algorithm is $T = 2^{(r-u)n}$, whereas its memory complexity is $M = 2^{un}$. Hence, the algorithm allows to achieve the tradeoff curve $TM = 2^{rn}$ for any values T, M such that $M \leq 2^{\lfloor r/2 \rfloor n}$.² Note that the algorithm can be applied also if the number of available plaintext/ciphertext pairs is $r' < r$. In such case, it outputs all the possible key candidates, whose expected number is $2^{(r-r')n}$ (since the plaintext/ciphertext pairs yield an $r'n$ -bit condition on the 2^{rn} possible keys).

The meet in the middle attack, designed for breaking double-encryption, is still the best known generic attack on double encryption schemes. It is also the best known attack for triple encryption upto logarithmic factors,³ which was studied very extensively due to its relevance to the former de-facto encryption standard Triple-DES.

3.2 The Basic Dissection Algorithm: Attacking 4-Encryption

In the followings we show that for $r \geq 4$, the basic meet in the middle algorithm can be outperformed significantly, using a dissection technique. For the basic

² We note that the algorithm, as described above, works only for $u \in \mathbb{N}$. However, it can be easily adapted to non-integer values of $u \leq \lfloor r/2 \rfloor$, preserving the tradeoff curve $TM = 2^{rn}$.

³ We note that a logarithmic time complexity improvement can be achieved in these settings as suggested by Lucks [12]. The improvement relies on the variance in the number of keys encrypting a given plaintext to a given ciphertext. This logarithmic gain in time complexity comes hand in hand with an exponential increase in the data complexity (a factor 8 gain in the time complexity when attacking triple-DES increases the data from 3 plaintext-ciphertext pairs to 2^{45} such pairs).

case $r = 4$, considered in this section, our algorithm runs in time $T = 2^{2n}$ with memory 2^n , thus allowing to reach $TM = 2^{3n}$, which is significantly better than the $TM = 2^{4n}$ curve suggested by the meet-in-the-middle attack.

The main idea behind the algorithm is to dissect the 4-encryption into two 2-encryption schemes, and to apply the meet in the middle attack to each of them separately. The partition is achieved by enumerating parts of the internal state at the dissection point. The basic algorithm, which we call $Dissect_2(4, 1)$ for reasons which will become apparent later, is as follows:

1. Given four known plaintexts (P_1, P_2, P_3, P_4) and their corresponding ciphertexts (C_1, C_2, C_3, C_4) , for each candidate value of $X_1^2 = E_{k_2}(E_{k_1}(P_1))$:
2. (a) Run the standard meet in the middle attack on 2-round encryption with (P_1, X_1^2) as a single plaintext-ciphertext pair. For each of the 2^n values of (k_1, k_2) output by the attack, partially encrypt P_2 using (k_1, k_2) , and store in a table the corresponding values of X_2^2 , along with the values of (k_1, k_2) .
2. (b) Run the standard meet in the middle attack on 2-round encryption with (X_1^2, C_1) as a single plaintext-ciphertext pair. For each of the 2^n values of (k_3, k_4) , partially decrypt C_2 using (k_3, k_4) and check whether the suggested value for X_2^2 appears in the table. If so, check whether the key (k_1, k_2, k_3, k_4) suggested by the table and the current (k_3, k_4) candidate encrypts P_3 and P_4 into C_3 and C_4 , respectively.

It is easy to see that once the right value for X_1^2 is considered, the right values of (k_1, k_2) are found in Step 2(a) and the right values of (k_3, k_4) are found in Step 2(b), and thus, the right value of the key is necessarily found. The time complexity of the algorithm is 2^{2n} . Indeed, Steps 2(a) and 2(b) are called 2^n times (for each value of X_1^2), and each of them runs the basic meet in the middle attack on 2-encryption in expected time and memory of 2^n . The number of expected collisions in the table of X_2^2 is 2^n . Thus, the expected time complexity of the attack⁴ is $2^n \cdot 2^n = 2^{2n}$.

The memory consumption of the 2-encryption meet in the middle steps is expected to be about 2^n . The size of the table “passed” between Steps 2(a) and 2(b) is also 2^n , since each meet in the middle step is expected to output 2^n key candidates. Hence, the expected memory complexity of the entire algorithm is 2^n .

3.3 Natural Extensions of the Basic Dissection Algorithm

We now consider the case ($r > 4, m = 1$) and show that natural extensions of the $Dissect_2(4, 1)$ algorithm presented above, allow to increase the gain over the standard meet in the middle attack significantly for larger values of r .

It is clear that any algorithm for r' -encryption can be extended to attack r -encryption for any $r > r'$, by trying all possible $r - r'$ keys $(k_{r'+1}, \dots, k_r)$, and applying the basic algorithm to the remaining $E^{[1 \dots r']}$. The time complexity is

⁴ We remind the reader that we disregard factors which are polynomial in n and r .

increased by a multiplicative factor of $2^{(r-r')n}$, and hence, the ratio $2^{rn}/TM$ is preserved. This leads to the following natural definition.

Definition 1. Given an algorithm A for r -encryption whose time and memory complexities are T and M , respectively, we define $\text{Gain}(A) = \log(2^{rn}/TM)/n = r - \log(TM)/n$. The maximal gain amongst all deterministic algorithms for r -encryption which use 2^{mn} memory, is denoted by $\text{Gain}_D(r, m)$.

By the trivial argument above, $\text{Gain}_D(r, 1)$ is monotone non-decreasing with r . The $\text{Dissect}_2(4, 1)$ algorithm shows that $\text{Gain}_D(r, 1) \geq 1$ for $r = 4$, and hence, for all $r \geq 4$. Below we suggest two natural extensions, which allow to increase the gain up to \sqrt{r} .

The LogLayer Algorithm: The first extension of the $\text{Dissect}_2(4, 1)$ is the recursive LogLayer_r algorithm, applicable when r is a power of 2, which tries all the possible X_1^{2i} for $i = 1, 2, \dots, r/2 - 1$ and runs simple meet in the middle attacks on each subcipher $E^{[2i+1\dots 2i+2]}$ separately. As each such attack returns 2^n candidate keys (which can be stored in memory of $(r/2) \cdot 2^n$), the algorithm then groups 4 encryptions together, enumerates the values X_2^{4i} for $i = 1, 2, \dots, r/4 - 1$, and runs meet in the middle attacks on each subcipher $E^{[4i+1\dots 4i+4]}$ separately (taking into account that there are only 2^n possibilities for the keys (k_{4i+1}, k_{4i+2}) and 2^n possibilities for the keys (k_{4i+3}, k_{4i+4})). The algorithm continues recursively (with $\log r$ layers in total), until a single key candidate is found.

The memory complexity of LogLayer_r is 2^n (as we need to store a number linear in r of lists of 2^n keys each). As in the j -th layer of the attack, $(r/2^j) - 1$ intermediate values are enumerated, and as each basic meet in the middle attack has time complexity of 2^n , the overall time complexity of the attack is

$$\prod_{j=1}^{\log r} 2^{n((r/2^j)-1)} \cdot 2^n = 2^{n(r-\log r)}.$$

The gain is thus $\text{Gain}(\text{LogLayer}_r) = \log r - 1$, which shows that $\text{Gain}_D(r, 1) \geq \lfloor \log r \rfloor - 1$.

The Square_r Algorithm: This logarithmic gain of LogLayer can be significantly outperformed by the Square_r algorithm, applicable when $r = (r')^2$ is a perfect square. The Square_r algorithm starts by trying all the possible values of $r' - 1$ intermediate values every r' rounds, a total of $(r' - 1)^2$ intermediate encryption values. Namely, the algorithm starts by enumerating all $X_1^{r'}, X_2^{r'}, \dots, X_{r'-1}^{r'}, X_1^{2r'}, X_2^{2r'}, \dots, X_{r'-1}^{2r'}, \dots, X_1^{r'(r'-1)}, X_2^{r'(r'-1)}, \dots, X_{r'-1}^{r'(r'-1)}$. Given these values, the adversary can attack each of the r' -encryptions (e.g., $E^{[1\dots r']}$), separately, and obtain 2^n “solutions” on average. Then, the adversary can treat each r' -round encryption as a single encryption with 2^n possible keys, and apply an r' -encryption attack to recover the key.

The time complexity of Square_r is equivalent to repeating $2^{(r'-1)(r'-1)n}$ times a sequence of $r' + 1$ attacks on r' -encryption. Hence, the time complexity is at

most $2^{[(r'-1)(r'-1)+(r'-1)] \cdot n}$, and the memory complexity is kept at 2^n . Therefore, $\text{Gain}(\text{Square}_r) \geq \sqrt{r} - 1$, which shows that $\text{Gain}_D(r, 1) \geq \lfloor \sqrt{r} \rfloor - 1$.

Obviously, improving the time complexity of attacking r' -encryption with 2^n memory reduces the time complexity of Square_r as well. However, as the best known attacks of this kind yields a gain of $O(\sqrt{r'}) = O(r^{1/4})$, the addition to the overall gain of Square_r is negligible.

3.4 Asymmetric Dissections: 7-Encryption and Beyond

A common feature shared by the LogLayer_r and the Square_r algorithms is their symmetry. In both algorithms, every dissection partitions the composition into parts of the same size. In this section we show that a better gain can be achieved by an asymmetric dissection, and present the optimal dissection algorithms of this type for $m = 1$ and any number r of encryptions.

We observe that the basic dissection attack is asymmetric in its nature. Indeed, after the two separate meet in the middle attacks are performed, the suggestions from the upper part are stored in a table, while the suggestions from the lower part are checked against the table values. As a result, the number of suggestions in the upper part is bounded from above by the size of the memory (which is now assumed to be 2^n and kept in sorted order), while the number of suggestions from the lower part can be arbitrarily large and generated on the fly in an arbitrary order. This suggests that an asymmetric dissection in which the lower part is bigger than the upper part, may result in a better algorithm. This is indeed the case, as illustrated by the following $\text{Dissect}_3(7, 1)$ algorithm:

1. Given 7 plaintext-ciphertext pairs $(P_1, C_1), (P_2, C_2), \dots, (P_7, C_7)$, for each possible value of X_1^3, X_2^3 , perform:
 - (a) Apply the basic MITM algorithm to $E^{[1\dots 3]}$ with (P_1, X_1^3) and (P_2, X_2^3) as the plaintext/ciphertext pairs, and obtain 2^n candidates for the keys (k_1, k_2, k_3) . For each such candidate, partially encrypt the rest of the plaintexts using (k_1, k_2, k_3) and store the values (X_4^3, \dots, X_7^3) in a table, along with the corresponding key candidate (k_1, k_2, k_3) .
 - (b) Apply $\text{Dissect}_2(4, 1)$ to $E^{[4\dots 7]}$ with (X_1^3, C_1) and (X_2^3, C_2) as the plaintext/ciphertext pairs. Note that since only two pairs are given, algorithm $\text{Dissect}_2(4, 1)$ produces 2^{2n} possible values of the keys (k_4, k_5, k_6, k_7) . However, these values are produced sequentially, and can be checked on-the-fly by partially decrypting C_4, C_5, C_6, C_7 , and checking whether the corresponding vector (X_4^3, \dots, X_7^3) appears in the table.

The memory complexity of the algorithm is 2^n , as both the basic meet in the middle attack on triple encryption and the algorithm $\text{Dissect}_2(4, 1)$ require 2^n memory, and the size of the table “passed” between Steps 2(a) and 2(b) is also 2^n .

The time complexity is 2^{4n} . Indeed, two n -bit values are enumerated in the middle, both the basic meet in the middle attack on triple encryption and the algorithm $\text{Dissect}_2(4, 1)$ require 2^{2n} time, and the remaining 2^{2n} possible values of (k_4, k_5, k_6, k_7) are checked instantly. This leads to time complexity of $2^{2n} \cdot 2^{2n} = 2^{4n}$.

This shows that $\text{Gain}(\text{Dissect}_3(7, 1)) = 2$, which is clearly better than the algorithms LogLayer_r and Square_r , which reach gain of 2 for the first time only at $r = 8$ and at $r = 9$, respectively.

Furthermore, the algorithm $\text{Dissect}_3(7, 1)$ can be extended recursively to larger values of r , to yield better asymptotic for the gain function. Given the algorithm $\text{Dissect}_j(r', 1)$ such that $\text{Gain}(\text{Dissect}_j(r', 1)) = \ell - 1$, we define the algorithm $\text{Dissect}_{j+1}^{\text{NEXT}} = \text{Dissect}_{\ell+1}(r' + \ell + 1, 1)$ for r -encryption, where $r = r' + \ell + 1$, as follows:

1. Given r plaintext-ciphertext pairs $(P_1, C_1), (P_2, C_2), \dots, (P_r, C_r)$, for each possible value of $X_1^{\ell+1}, \dots, X_{\ell}^{\ell+1}$, perform:
 - (a) Apply the basic MITM attack to $E^{[1 \dots \ell+1]}$ with $(P_1, X_1^{\ell+1}), \dots, (P_{\ell}, X_{\ell}^{\ell+1})$ as the plaintext/ciphertext pairs, and obtain 2^n candidates for the keys $(k_1, \dots, k_{\ell+1})$. For each such candidate, partially encrypt the rest of the plaintexts using $(k_1, \dots, k_{\ell+1})$ and store the values $(X_{\ell+1}^{\ell+1}, \dots, X_r^{\ell+1})$ in a table, along with the corresponding key candidate $(k_1, \dots, k_{\ell+1})$.
 - (b) Apply $\text{Dissect}_j(r', 1)$ to $E^{[\ell+2 \dots r]}$ with $(X_1^{\ell+1}, C_1), \dots, (X_{\ell}^{\ell+1}, C_{\ell})$ as the plaintext/ciphertext pairs. Check each of the $2^{(r'-\ell)n}$ suggestions for the keys $(k_{\ell+2}, \dots, k_r)$ on-the-fly by partially decrypting $C_{\ell+1}, \dots, C_r$, and checking whether the corresponding vector $(X_{\ell+1}^{\ell+1}, \dots, X_r^{\ell+1})$ appears in the table.

An exactly similar argument as the one used for $\text{Dissect}_3(7, 1)$ shows that the time and memory complexities of $\text{Dissect}_{\ell+1}(r)$ are $2^{r'n}$ and 2^n , respectively, which implies that $\text{Gain}(\text{Dissect}_{\ell+1}(r)) = \ell$. In fact, $\text{Dissect}_3(7, 1)$ can be obtained from $\text{Dissect}_2(4, 1)$ by the recursive construction just described.

The recursion leads to a sequence of asymmetric dissection attacks with memory $M = 2^n$, such that the gain increases by 1 with each step of the sequence. If we denote the number r of ‘‘rounds’’ in the ℓ ’s element of the sequence (i.e., the element for which the gain equals to ℓ) by r_{ℓ} , then by the construction, the sequence satisfies the recursion

$$r_{\ell} = r_{\ell-1} + \ell + 1,$$

which (together with $r_2 = 4$ which follows from $\text{Dissect}_2(4, 1)$) leads to the formula:

$$r_{\ell} = \frac{\ell(\ell+1)}{2} + 1.$$

The asymptotic gain of this sequence is obtained by representing ℓ as a function of r , and is equal to $(\sqrt{8r-7}-1)/2 \approx \sqrt{2r}$, which is bigger than the \sqrt{r} gain of the Square_r algorithm.

A thorough analysis, presented in the extended version of this paper [4], shows that the algorithms obtained by the recursive sequence described above are the optimal amongst all dissection algorithms that split the r rounds into two (not necessarily equal) parts, and attacks each part recursively, using an optimal dissection algorithm.

We conclude that as far as only dissection attacks are concerned, the “magic sequence” of the minimal numbers of rounds for which the gains are $\ell = 0, 1, 2, \dots$, is:

$$\text{Magic}_1 = \{1, 2, 4, 7, 11, 16, 22, 29, 37, 46, 56, \dots\}.$$

This “magic sequence” will appear several more times in the sequel.

4 Parallel Collision Search via Dissection

In Section 3, we considered the scenario of deterministic algorithms which never err for r -encryption, that is, algorithms which find all the possible values of the keys which comply with the plaintext/ciphertext pairs, or prove that there are no such keys. In this scenario, the best previously known generic attack is the meet in the middle attack, which obtains the tradeoff curve $TM = 2^{rn}$, where T and M are the time and memory complexities of the algorithm, respectively. In this model, we presented several dissection algorithms which allow to achieve the curve $TM = 2^{(r-\sqrt{2r})n}$.

In this section, we consider the scenario in which non-deterministic algorithms, which find the right keys with some probability $p < 1$, are allowed. In this case, an improved tradeoff curve of $T^2M = 2^{(3/2)rn}$ can be obtained by the *parallel collision search* algorithm of van Oorschot and Wiener [18]. We now show how to combine the dissection algorithms presented in Section 3 with the parallel collision search algorithm to obtain an even better tradeoff curve with a multiplicative gain of $2^{(\sqrt{2r}/8)n}$ over the curve of [18].

4.1 Brief Description of the Parallel Collision Search Algorithm

We start with a very brief description of the PCS algorithm suggested in [18].

The algorithm consists of two steps:

1. Find *partial collisions*, which are key suggestions which comply with half of the plaintext/ciphertext pairs.
2. For each partial collision, check whether it complies with the second half of the plaintext/ciphertext pairs.

The first step is performed by constructing two step functions:⁵

$$\begin{aligned} F^{upper} : (k_1, \dots, k_{r/2}) &\mapsto (X_1^{r/2}, \dots, X_{r/2}^{r/2}) \quad \text{and} \\ F^{lower} : (k_{r/2+1}, \dots, k_r) &\mapsto (X_1^{r/2}, \dots, X_{r/2}^{r/2}), \end{aligned}$$

which can be computed easily given the pairs $(P_1, C_1), \dots, (P_{r/2}, C_{r/2})$, and using Floyd’s cycle finding algorithm [10] (or another cycle finding algorithm which requires little memory, such as [16]) to find a collision between them. In the case of constant memory, Floyd’s algorithm finds such a collision (which produces a

⁵ The idea of constructing two step functions was first proposed in [8].

key suggestion which complies with the pairs $(P_1, C_1), \dots, (P_{r/2}, C_{r/2})$) in $2^{(r/4)n}$ time. If $M = 2^{mn}$ memory is given, this step can be speeded up by incorporating Hellman's time-memory tradeoff techniques [5], that allow to find 2^{mn} collisions simultaneously in $2^{(r/4+m/2)n}$ time. In both cases, after $2^{(r/2)n}$ partial collisions are found, it is expected that one of them passes the condition of the second step, which means that it is the desired key suggestion. The time complexity of the algorithm is $T = 2^{(r/4+m/2)n} \cdot 2^{(r/2-m)n} = 2^{(3r/4-m/2)n}$, which leads to the tradeoff curve $T^2 M = 2^{(3/2)rn}$.

4.2 The Dissect and Collide Algorithm

In this section we present the Dissect & Collide (*DC*) algorithm, which uses dissection to enhance the PCS algorithm.

The basic idea behind the *DC* algorithm is that it is possible to fix several intermediate values after $r/2$ rounds, that is, $(X_1^{r/2}, \dots, X_u^{r/2})$, and construct step functions \tilde{F}^{upper} and \tilde{F}^{lower} in such a way that all the keys they suggest partially encrypt P_i to $X_i^{r/2}$ and partially decrypt C_i to $X_i^{r/2}$, for all $i \leq u$. This is achieved by incorporating an attack on $E^{[1\dots r/2]}$ with $(P_1, X_1^{r/2}), \dots, (P_u, X_u^{r/2})$ as the plaintext/ciphertext pairs into the function F^{upper} , and similarly with $E^{[r/2+1\dots r]}$ and F^{lower} . As a result, a partial collision which complies with the pairs $(P_1, C_1), \dots, (P_{r/2}, C_{r/2})$ can be found at the smaller “cost” of finding a collision which complies only with $(P_{u+1}, C_{u+1}), \dots, (P_{r/2}, C_{r/2})$. It should be noted that this gain could be diminished by the “cost” of the new step function \tilde{F} , that is higher than the “cost” of the simpler step function F . However, we show that if the efficient dissection algorithms presented in Section 3 are used to attack the subciphers $E^{[1\dots r/2]}$ and $E^{[r/2+1\dots r]}$, the gain is bigger than the loss, and the resulting *DC* algorithm is faster than the PCS algorithm (for the same amount of memory).

A Basic Example: Applying DC to 8-encryption. As the idea of the *DC* algorithm is somewhat involved, we illustrate it by considering the simple case ($r = 8, m = 1$). In the case of 8-encryption, the goal of the first step in the PCS algorithm is to find partial collisions which comply with the pairs $(P_1, C_1), \dots, (P_4, C_4)$. Given memory of 2^n , the average time PCS requires for finding each such collision is $2^{1.5n}$. The *DC* algorithm allows to achieve the same goal in 2^n time.

In the *DC* algorithm, we fix three intermediate values: (X_1^4, X_2^4, X_3^4) , and want to attack the subciphers $E^{[1\dots 4]}$ and $E^{[5\dots 8]}$. Recall that *Dissect*₂(4, 1) presented in Section 3 allows to retrieve all 2^n values of (k_1, k_2, k_3, k_4) which comply with the pairs $(P_1, X_1^4), (P_2, X_2^4), (P_3, X_3^4)$ in time 2^{2n} and memory 2^n . Furthermore, given a fixed value X_1^2 , there is a single value of (k_1, k_2, k_3, k_4) (on average) which complies with the three plaintext/ciphertext pairs and the X_1^2 value, and this value can be found in time 2^n (since the *Dissect*₂(4, 1) algorithm starts with guessing the value X_1^2 and then performs only 2^n operations for each guess).

The algorithm works as follows:

1. Given the plaintexts (P_1, P_2, P_3, P_4) and their corresponding ciphertexts (C_1, C_2, C_3, C_4) , for each guess of (X_1^4, X_2^4, X_3^4) :
2. (a) Define the step functions \tilde{F}^{upper} and \tilde{F}^{lower} by:

$$\tilde{F}^{upper} : X_1^2 \mapsto X_4^4 \quad \text{and} \quad \tilde{F}^{lower} : X_1^6 \mapsto X_4^4.$$

In order to compute the step function \tilde{F}^{upper} , apply *Dissect*₂(4, 1) to $E^{[1\dots 4]}$ with the plaintext/ciphertext pairs $(P_1, X_1^4), (P_2, X_2^4), (P_3, X_3^4)$ and the intermediate value X_1^2 to obtain a unique value of the keys (k_1, k_2, k_3, k_4) . Then, partially encrypt P_4 through $E^{[1\dots 4]}$ with these keys to obtain $\tilde{F}^{upper}(X_1^2) = X_4^4$. The function \tilde{F}^{lower} is computed similarly.

- (b) Find a collision between the functions \tilde{F}^{upper} and \tilde{F}^{lower} using a variant of Floyd's cycle finding algorithm which exploits the $M = 2^n$ available amount of memory.
- (c) Check whether the keys $(k_1, \dots, k_4, k_5, \dots, k_8)$ suggested by the partial collision, encrypt (P_5, \dots, P_8) to (C_5, \dots, C_8) . If not, return to Step 2(a). After 2^n partial collisions are examined and discarded, return to Step 1, and pick a different guess for (X_1^4, X_2^4, X_3^4) .

By the properties of the algorithm *Dissect*₂(4, 1) mentioned above, each step of the functions \tilde{F} can be performed in 2^n time and memory. By the construction of the step functions, each suggested key (k_1, \dots, k_4) (or (k_5, \dots, k_8)) encrypts (P_1, P_2, P_3) to (X_1^4, X_2^4, X_3^4) (or decrypts (C_1, C_2, C_3) to (X_1^4, X_2^4, X_3^4) , respectively), and hence, each collision between \tilde{F}^{upper} and \tilde{F}^{lower} yields a suggestion of $(k_1, \dots, k_4, k_5, \dots, k_8)$ which complies with the pairs $(P_1, C_1), \dots, (P_4, C_4)$. Finally, since the step functions are from n bits to n bits, collision between them can be found instantly given 2^n memory. Therefore, the time required for finding a partial collision is 2^n , and thus, the total running time of the algorithm is $2^{4n} \cdot 2^n = 2^{5n}$. We note that while our *DC* algorithm outperforms the respective PCS algorithm (whose time complexity is $2^{5.5n}$), it has the same performance as the *Dissect*₄(8, 1) algorithm presented in Section 3. However, as we will show in the sequel, for larger values of r , the *DC* algorithms outperform the *Dissect* algorithms significantly.

The General Algorithms *DC*(r, m). Now we are ready to give a formal definition of the class $DC(r, m)$ of algorithms, applicable to r -encryption (for an even r)⁶, given memory of 2^{mn} . An algorithm $A \in DC(r, m)$ is specified by a number u , $1 \leq u \leq r/2$, and two sets I^{upper} and I^{lower} of intermediate locations in the subciphers $E^{[1\dots r/2]}$ and $E^{[r/2+1\dots r]}$, respectively, such that $|I^{upper}| = |I^{lower}| = r/2 - u$.

⁶ We note that for sake of simplicity, we discuss in this section only even values of r . An easy (but probably non-optimal) way to use these algorithms for an odd value of r is to guess the value of the key k_r , and for each guess, to apply the algorithms described in this section to $E^{[1\dots r-1]}$.

In the algorithm, the adversary fixes u intermediate values $(X_1^{r/2}, \dots, X_u^{r/2})$. Then, she defines the step functions \tilde{F}^{upper} and \tilde{F}^{lower} by:

$$\tilde{F}^{upper} : I^{upper} \mapsto (X_{u+1}^{r/2}, \dots, X_{r/2}^{r/2}) \quad \text{and} \quad \tilde{F}^{lower} : I^{lower} \mapsto (X_{u+1}^{r/2}, \dots, X_{r/2}^{r/2}).$$

The step function \tilde{F}^{upper} is computed by applying a dissection attack to $E^{[1\dots r/2]}$ with the plaintext/ciphertext pairs $(P_1, X_1^{r/2}), \dots, (P_u, X_u^{r/2})$ and the intermediate values contained in I^{upper} to retrieve a unique value of the keys $(k_1, \dots, k_{r/2})$, and then partially encrypting (P_{u+1}, \dots, P_r) to obtain $(X_{u+1}^{r/2}, \dots, X_{r/2}^{r/2})$. The step function \tilde{F}^{lower} is computed in a similar way, with respect to $E^{[r/2+1\dots r]}$ and the set I^{lower} . Then, a variant of Floyd's cycle finding algorithm which exploits the 2^{mn} amount of available memory is used to find a collision between \tilde{F}^{upper} and \tilde{F}^{lower} , which yields a suggestion of $(k_1, \dots, k_{r/2}, k_{r/2+1}, \dots, k_r)$ which complies with the plaintext/ciphertext pairs $(P_1, C_1), \dots, (P_{r/2}, C_{r/2})$.

Denote the time complexity of each application of \tilde{F} by $S = 2^{sn}$. An easy computation shows that the overall time complexity of the algorithm $DC(r, m)$ is:

$$2^{(r/2)n} \cdot 2^{((r/2-u-m)/2)n} \cdot 2^{sn} = 2^{((3/4)r-(u+m-2s)/2)n}. \quad (1)$$

As the time complexity of the PCS algorithm with memory 2^{mn} is $2^{((3/4)r-m/2)n}$, the multiplicative gain of the DC algorithm is $2^{(u/2-s)n}$. In particular, for the specific $DC(8, 1)$ algorithm described above for 8-encryption, we have $s = 1$, and thus, the advantage is indeed $2^{(3/2-1)n} = 2^{n/2}$ (i.e., the gain is $1/2$), as mentioned above. In the sequel, we denote the parameters $I^{upper}, I^{lower}, u, s$ which specify a $DC(r, m)$ algorithm A and determine its time complexity by $I^{upper}(A), I^{lower}(A), u(A)$, and $s(A)$, respectively.

We conclude this section by mentioning a difficulty in the implementation of the DC algorithm. Unlike the PCS algorithm where the output of the step functions F is always uniquely defined, in DC the functions \tilde{F} return no output for some of the inputs. This happens since the number of keys $(k_1, \dots, k_{r/2})$ which comply with the u plaintext/ciphertext values $(P_1, X_1^{r/2}), \dots, (P_u, X_u^{r/2})$ and the $r/2 - u$ fixed intermediate values contained in I^{upper} , is distributed according to the distribution $Poisson(1)$, and in particular, equals to zero for an $1/e$ fraction of the inputs. This difficulty can be resolved by introducing *flavors* into the step function \tilde{F} , which alter the function in a deterministic way when it fails to produce output. The exact modification is described in the extended version of this paper [4].

4.3 The Gain of the Dissect and Collide Algorithm over the PCS Algorithm

In this section we consider several natural extensions of the basic $DC(8, 1)$ algorithm presented in Section 4.2. We use these extensions to show that the gain of the DC algorithms over the PCS algorithm is monotone non-decreasing with r and is lower bounded by $2^{(\lfloor \sqrt{2r} \rfloor / 8)n}$ for any $r \geq 8$.

Before we present the extensions of the basic *DC* algorithm, we would like to define formally the notion of *gain* in the non-deterministic setting. As the best previously known algorithm in this setting is the PCS algorithm, whose time complexity given 2^{mn} memory is $2^{((3/4)r-m/2)n}$, we define the gain with respect to it.

Definition 2. *The gain of a probabilistic algorithm A for r -encryption whose time and memory complexities are T and $M = 2^{mn}$, respectively, is defined as*

$$\text{Gain}_{ND}(A) = (3/4)r - m/2 - (\log T)/n.$$

The maximal gain amongst all probabilistic DC algorithms for r -encryption which require 2^{mn} memory, is denoted by $\text{Gain}_{ND}(r, m)$.

Note that it follows from Equation (1) that if $A \in DC(r, m)$, then

$$\text{Gain}_{ND}(A) = u(A)/2 - s(A). \quad (2)$$

Monotonicity of the Gain. The most basic extension of the basic *DC* algorithm is to preserve the gain when additional “rounds” are added. While in the deterministic case, such an extension can be obtained trivially by guessing several keys and applying the previous algorithm, in our setting this approach leads to a decrease of the gain by $1/2$ for each two added rounds (as the complexity of the PCS algorithm is increased by a factor of $2^{3n/2}$ when r is increased by 2). However, the gain can be preserved in another way, as shown in the following lemma.

Lemma 1. *Assume that an algorithm $A \in DC(r', m)$ has gain ℓ . Then there exists an algorithm $B \in DC(r' + 2, m)$ whose gain is also equal to ℓ .*

Due to space restrictions, the proof of the lemma is presented in the extended version of this paper [4]. Here we only note that the algorithm B is constructed from A by choosing $I^{upper}(B) = I^{upper}(A) \cup \{X_1^{r'/2}\}$, and similarly for $I^{lower}(B)$.

Lemma 1 implies that the gain of the *DC* algorithms is monotone non-decreasing with r , and in particular, that $\text{Gain}_{ND}(r, 1) \geq 1/2$, for any even $r \geq 8$.

An Analogue of the *LogLayer* Algorithm. The next natural extension of the basic *DC* algorithm is an analogue of the *LogLayer* algorithm presented in Section 3.3. Recall that the LogLayer_r algorithm, applicable when r is a power of 2, consists of guessing the set of intermediate values:

$$I_0 = \{X_1^2, X_1^4, \dots, X_1^{r-2}, X_2^4, X_2^8, \dots, X_2^{r-4}, X_3^8, \dots, X_3^{r-8}, \dots, X_{\log r-1}^{r/2}\},$$

and applying a recursive sequence of meet in the middle attacks on 2-encryption. Using this algorithm, we can define the algorithm $LL_r \in DC(2r, 1)$, by specifying $I^{upper}(LL_r) = I_0$, and $I^{lower}(LL_r)$ in a similar way. Since $|I_0| = r - \log r - 1$,

we have $u(LL_r) = r - (r - \log r - 1) = \log r + 1$. It follows from the structure of the LogLayer_r algorithm that given the values in I_0 , it can compute the keys (k_1, \dots, k_r) in time and memory of 2^n . Hence, we have $s(LL_r) = 1$. By Equation (2), it follows that $\text{Gain}(LL_r) = (\log r + 1)/2 - 1 = (\log r - 1)/2$.

The basic algorithm for 8-encryption is the special case LL_4 of this algorithm. The next two values of r also yield interesting algorithms: LL_8 yields gain of 1 for $(r = 16, m = 1)$, which amounts to an attack on 16-encryption with $(T = 2^{10.5n}, M = 2^n)$, and LL_{16} yields gain of 1.5 for $(r = 32, m = 1)$, which amounts to an attack on 32-encryption with $(T = 2^{22n}, M = 2^n)$. Both attacks outperform the *Dissect* attacks and are the best known attacks on 16-encryption and on 32-encryption, respectively.

An Analogue of the Square_r Algorithm: The logarithmic asymptotic gain of the LL sequence can be significantly outperformed by an analogue of the Square_r algorithm, presented in Section 3.3. Recall that the Square_r algorithm, applicable when $r = (r')^2$ is a perfect square, starts by guessing the set of $(r' - 1)^2$ intermediate encryption values:

$$I_1 = \{X_1^{r'}, \dots, X_{r'-1}^{r'}, X_1^{2r'}, \dots, X_{r'-1}^{2r'}, \dots, X_1^{r'(r'-1)}, \dots, X_{r'-1}^{r'(r'-1)}\},$$

and then performs a two-layer attack, which amounts to $r' + 1$ separate attacks on r' -encryption. Using this algorithm, we can define the algorithm $Sq_r \in DC(2r, 1)$, by specifying $I^{upper}(Sq_r) = I_0$, and $I^{lower}(Sq_r)$ in a similar way. Since $|I_0| = (r' - 1)^2$, we have $u(Sq_r) = r - (r' - 1)^2 = 2r' - 1$. The step complexity $s(Sq_r)$ is the time complexity required for attacking r' -encryption without fixed intermediate values. Hence, by Equation (2),

$$\text{Gain}(Sq_r) = r' - 1/2 - f_1(r'),$$

where $2^{f_1(r)n}$ is the time complexity of the best possible attack on r -encryption with 2^n memory.

The basic algorithm for 8-encryption is the special case Sq_2 of this algorithm. Since for small values of r' , the best known attacks on r' -encryption are obtained by the dissection attacks presented in Section 3.4, the next elements of the sequence Sq_r which increase the gain, correspond to the next elements of the sequence $Magic_1 = \{1, 2, 4, 7, 11, 16, \dots\}$ described in Section 3.4. They lead to gains of 1.5, 2.5, and 3.5 for $r = 32, 98$, and $r = 242$, respectively. For large values of r , the PCS algorithm outperforms the *Dissect* algorithms, and using it we obtain:

$$\text{Gain}(Sq_r) \geq r' - 1/2 - ((3/4)r' - 1/2) = r'/4 = \sqrt{2r}/8.$$

This shows that the asymptotic gain of the *DC* algorithms is at least $\sqrt{2r}/8$.

We note that as for $r' \geq 16$, the *DC* algorithm outperforms both the *Dissect* and the PCS algorithms, we can use it instead of PCS in the attacks on r' -encryption in order to increase the gain for large values of r . However, as the gain of *DC* over PCS for r' -encryption is only of order $O(\sqrt{r'}) = O(r^{1/4})$, the addition to the overall gain of Sq_r is negligible.

Two-Layer DC Algorithms. A natural extension of the Sq_r algorithm is the class of *two-layer DC* algorithms. Assume that $r = 2r_1 \cdot r_2$, and that there exist algorithms A_1, A_2 for r_1 -encryption and for r_2 -encryption, respectively, which perform in time 2^{sn} and memory 2^n given sets of intermediate values I_1^{upper} and I_2^{upper} , respectively.

Then we can define an algorithm $A \in DC(r, 1)$ whose step function is computed by a two-layer algorithm: First, $E^{[1\dots r/2]}$ is divided into r_2 subciphers of r_1 rounds each, and the algorithm A_1 is used to attack each of them separately and compute 2^n possible suggestions for each set of r_1 consecutive keys. Then, each r_1 -round encryption is considered as a single encryption with 2^n possible keys, and the algorithm A_2 is used to attack the resulting r_2 -encryption. The set $I^{upper}(A)$ is chosen such that both A_1 and A_2 algorithms perform in time 2^s . Formally, if we denote $u_1 = |I_1^{upper}|$, then the set $I^{upper}(A)$ consists of r_2 “copies” of the set I_1^{upper} , $r_1 - 1 - u_1$ intermediate values after each r_1 rounds, and one copy of the set I_2^{upper} . The set $I^{lower}(A)$ is defined similarly. Hence,

$$\begin{aligned} u(A) &= r/2 - |I^{upper}(A)| = r/2 - (r_2 \cdot u_1 + (r_2 - 1)(r_1 - 1 - u_1) + u_2) = \\ &= r_2 + r_1 - u_1 - u_2 - 1. \end{aligned}$$

As $s(A) = s$, we have $Gain_{ND}(A) = (r_2 + r_1 - u_1 - u_2 - 1)/2 - s$.

Note that the algorithm Sq_r is actually a two-layer DC algorithm, with $r_1 = r_2 = r'$ and $I_1^{upper} = I_2^{upper} = \emptyset$. It turns out that for all $8 \leq r \leq 128$, the maximal gains are obtained by two-layer DC algorithms where r_1, r_2 are chosen from the sequence $Magic_1$ presented in Section 3.4, and A_1, A_2 are the respective *Dissect* algorithms. The cases of $r = 8, 16, 32$ presented above are obtained with $r_1 = 4$ and $r_2 = 1, 2, 4$ (respectively), and the next numbers of rounds in which the gain increases are $r = 56, 88, 128$, obtained for $r_1 = 4$ and $r_2 = 7, 11, 16$, respectively. The continuation of the “non-deterministic magic sequence” is, however, more complicated. For example, the two-layer algorithm for $r = 176$ with $(r_1 = 4, r_2 = 22)$ has the same gain as the algorithm with $(r_1 = 4, r_2 = 16)$, and the next increase of the gain occurs only for $r = 224$, and is obtained by a two-layer algorithm with $(r_1 = 7, r_2 = 16)$. For larger values of r , more complex algorithms, such as a three-layer algorithm with $r_1 = r_2 = r_3 = 7$ for 686-encryption, outperform the two-layer algorithms. We leave the analysis of the whole magic sequence to the full version of the paper, and conclude that the minimal numbers of rounds for which the gain equals $0.5, 1, 1.5, \dots$ are:

$$Magic_1^{ND} = \{8, 16, 32, 56, 88, 128, \dots\}.$$

Finally, we note that two-layer DC algorithms can be applied also for $m > 1$, and can be used to show that the first numbers of rounds for which $Gain_{ND}(r, m) = 0.5, 1, 1.5, 2, \dots$ are:

$$\begin{aligned} Magic_m^{ND} = \{ &8m, 8m + 8, 8m + 16, \dots, 16m, 16m + 16, 16m + 32, \dots, 32m, \\ &32m + 24, 32m + 48, \dots, 56m, \dots \}. \end{aligned}$$

The full analysis of the case $m > 1$ will appear in the full version of the paper.

5 Applications

In this section, we apply our new dissection algorithms to several well known bicomposite search problems. As described in the introduction, we can represent such a problem as an $r \times r$ execution matrix which is treated as a multiple-encryption scheme with r rounds. In the case of knapsacks, we are allowed to choose any constant value of r when n grows to infinity in order to optimize the value of t for any given m . In other cases, r is restricted to a specific value or to a set of values. For example, in the special case of Rubik's cube, we know that a 20-move solution exists for any reachable state, and thus it does not make sense to choose $r > 20$. Such constraints can limit the choice of parameters for our algorithms, and thus we may not be able to exploit the available memory as efficiently as in the case of knapsacks.

Since the analysis of our multiple encryption algorithms assumes the randomness of the underlying block ciphers, we have to justify this assumption for each reduction we consider. For example, in the case of knapsacks with n generators, strong randomness assumptions are common practice whenever n is sufficiently large (e.g., see [1,7]), and we can use the same assumptions when we consider subproblems with n/r generators for any constant r .

5.1 Applications to Knapsacks

The knapsack problem is a well-known problem that has been studied for many years. For more than 30 years, the best known algorithm for knapsacks was the Schroeppel-Shamir algorithm [17], which requires $2^{n/2}$ time and $2^{n/4}$ memory. Surprisingly, in 2010, Howgrave-Graham and Joux [7] showed how to solve the knapsack problem in time much better than $2^{n/2}$, by using the associativity and commutativity properties of addition. This result was further improved by Becker, Coron and Joux in [1]. In addition to their basic results, Howgrave-Graham and Joux [7] also described reduced-memory algorithms, and in particular [1] described a memoryless attack which requires only $2^{0.72n}$ time. All these new attacks are heuristic in a sense that they may fail to find a solution even when it exists, and thus they cannot be used in order to prove the nonexistence of solutions. In addition to these heuristic algorithms, Becker, Coron and Joux [1] also considered deterministic algorithms that never err, and described a straight-line time-memory tradeoff curve, but this curve was only valid in the range $1/16 \leq m \leq 1/4$.

In this section, we show how to use our generic dissection techniques in order to find deterministic algorithms for the knapsack problem which are better than the deterministic tradeoff curve described in [1] over the whole range of $1/16 < m < 1/4$. In addition, we can expand our tradeoff curve in a continuous way for any smaller value of $m \leq 1/4$. By combining our generic deterministic and non-deterministic algorithms, we can get a new curve which is better than the

best knapsack-specific algorithms described in [7] and [1] for the large interval of (approximately) $1/100 \leq m < 1/6$.⁷

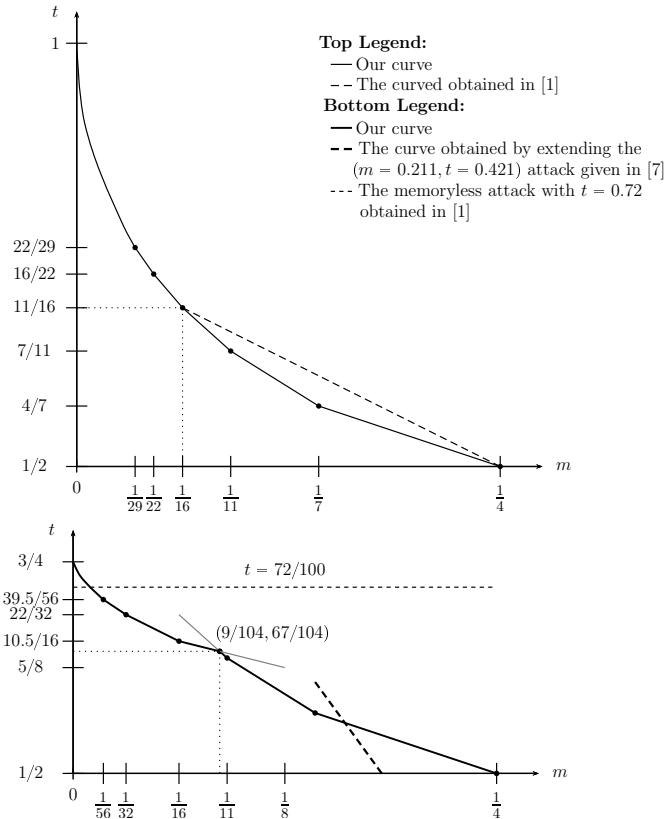
The formal reduction of the knapsack problem to r -round encryption (for any r) is given in the extended version of this paper [4], but it is not required in order to understand the rest of this paper. Given $M = 2^{mn}$ memory, our goal is to solve the knapsack problem by applying the reduction with a value of r which optimizes the time complexity of our multiple encryption algorithm. Formally, for any r , we apply the multiple encryption algorithm with an effective block size reduced by a factor of r , i.e., $n^* = n/r$. By equating $M = 2^{mn}$ with $M = 2^{n(m^*r)}$, we can see that the effective memory unit increases by the same ratio, i.e., $m^* = mr$. We denote by $f(r, n^*, m^*)$ the running time of our multiple encryption algorithm on an r -round block cipher with a block size of n^* bits and $M^* = 2^{m^*n^*}$ available memory, given r plaintext-ciphertext pairs. Using this notation, we would like to find r that minimizes $f(r, n^*, m^*) = f(r, n/r, mr)$. We call such a value of r an optimal value.

We note that the deterministic algorithms applied in [7] and [1] for $1/16 \leq m \leq 1/4$ implicitly perform a reduction to multiple encryption with the fixed parameters $r = 4$ and $r = 16$. In fact, for the case of knapsacks and these choices of r , these algorithms are closely related to our square algorithms (described in Section 3.3). However, as we now show, we can get a better tradeoff curve by using other choices of r .

Time-Memory Tradeoff Curves for Knapsacks. Using our multiple encryption algorithms, we construct time-memory tradeoff curves for knapsacks: we start with deterministic algorithms and consider first the case of $1/m \in \{1, 2, 4, 7, 11, 16, \dots\}$, which is the “magic sequence” constructed in Section 3.4. In order to simplify our notation, we denote the j 'th element of this sequence by b_j , starting from $j = 0$. In the case of $1/m = b_j$ for $j \geq 2$, we simply choose $r = 1/m = b_j$ and run the algorithm with $n^* = n/m$ and $m^* = m/m = 1$. For example, in case $m = 1/4$, we run the 4-round multiple encryption with $m^* = 1$, for which the time complexity is $T = 2^{2n^*} = 2^{2(n/4)} = 2^{n/2}$, or $t = 1/2$. In case $m = 1/7$, we run the 7-round dissection algorithm with $m^* = 1$, for which the time complexity is $T = 2^{4n^*} = 2^{4n/7}$, or $t = 4/7$. In the extended version of this paper [4], we show that in the case of $1/m \in \{4, 7, 11, 16, \dots\}$, our choice of r is indeed optimal. Thus, we obtain a sequence of optimal points on the deterministic time-memory tradeoff curve. In order to obtain an optimal continuous curve for $0 < m \leq 1/4$, we need to use our algorithms for integral $m^* \geq 2$. As described in the extended version of this paper [4], these algorithms enable us to connect in a straight line any two consecutive time-memory tradeoff points for $1/m \in \{4, 7, 11, 16, \dots\}$, and obtain a continuous curve (as shown on the left diagram of Figure 1).

⁷ We note that since our algorithms do not efficiently exploit more than $2^{n/4}$ memory, our tradeoff curves are only defined for $m \leq 0.25$. This should be contrasted with the non-deterministic algorithms of [7] and [1], whose main results solve the problem in time less than $2^{n/3}$ with about $2^{n/3}$ memory.

For non-deterministic algorithms, we use the same approach, and consider first the “magic sequence” constructed in Section 4 for $1/m \in \{16, 32, 56, \dots\}$. We choose $r = 1/m$ and the corresponding values of t ($1/10.5, 1/22, 1/39.5, \dots$). Similarly to the deterministic case, we can use our non-deterministic algorithms for integral $m^* \geq 2$ in order to obtain a continuous curve (as shown on the right diagram of Figure 1). The full details of how to connect consecutive points on the curve will be given in the full version of this paper.



On the top: A comparison between time-memory tradeoff curves obtained with deterministic algorithms. Our curve (defined for $m \leq 1/4$) is strictly better than the curve obtained in [1] (defined only for $1/16 \leq m \leq 1/4$) for any $1/16 < m < 1/4$.

On the bottom: A comparison between general time-memory tradeoff curves. Our general time-memory tradeoff curve is better than the attacks of [1] and [7] in the interval of (approximately) $1/100 \leq m < 1/6$.

Fig. 1. Time-Memory Tradeoff Curves for Knapsack

5.2 Improving Rebound Attacks on Hash Functions

Another application of our new techniques can significantly improve rebound attacks [13] on hash functions. An important procedure in such attacks is to match input/output differences through an S-box layer (or a generalized S-box layer). More precisely, the adversary is given a list L_A of input differences and a list L_B of output differences, and has to find all the input/output difference pairs that can happen through the S-box layer. A series of matching algorithms were recently developed, optimizing and improving various rebound attacks [15].

Our dissection algorithms can be applied for this problem as well, replacing the gradual matching or parallel matching presented at Crypto 2011 by [15]. For example, we can improve the rebound attack on Luffa using a variant of our *Dissect₂(4, 1)* algorithm. As described in the extended version of this paper [4], we can reduce the memory complexity of the matching algorithm from 2^{102} to only 2^{66} without affecting the time complexity of the attack (which remains at 2^{104}).

5.3 Applications to Relational Databases

As a final example of the versatility of our algorithm, we note that in some cases, it may be possible to use the dissection technique to speed up the processing of queries in relational databases. The problem of composing block ciphers can be viewed as the problem of computing the *join* of several databases, where each database contains all the possible plaintext/ciphertext pairs, and the join operation equates the previous ciphertext with the next plaintext. When intermediate joined databases blow up in size but the final database is quite small, it may be better to use the dissection technique which guesses some middle values and splits the computation into smaller independent parts. More details about this potential application will be given in the full version of this paper.

6 Summary

In this paper we introduced the new dissection technique which can be applied to a broad class of problems which have a bicomposite structure. We used this technique to obtain improved complexities for several well studied problems such as the cryptanalysis of multiple-encryption schemes and the solution of hard knapsacks. The main open problem in this area is to either improve our techniques or to prove their optimality. In particular, we conjecture (but can not prove) that any attack on multiple-encryption schemes should have a time complexity which is at least the square root of the total number of possible keys.

References

1. Becker, A., Coron, J.-S., Joux, A.: Improved Generic Algorithms for Hard Knapsacks. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 364–385. Springer, Heidelberg (2011)

2. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
3. Dinur, I., Dunkelman, O., Shamir, A.: Improved Attacks on Full GOST. In: Fast Software Encryption 2012. LNCS (to appear, 2012); Available as IACR ePrint report 2011/558
4. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems. Cryptology ePrint Archive, Report 2012/217 (2012)
5. Hellman, M.E.: A Cryptanalytic Time-Memory Tradeoff. IEEE Transactions on Information Theory 26(4), 401–406 (1980)
6. Fiat, A., Moses, S., Shamir, A., Shimshoni, I., Tardos, G.: Planning and Learning in Permutation Groups. In: Foundations of Computer Science 1989, pp. 274–279. IEEE Computer Society (1989)
7. Howgrave-Graham, N., Joux, A.: New Generic Algorithms for Hard Knapsacks. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 235–256. Springer, Heidelberg (2010)
8. Quisquater, J.-J., Delescaillie, J.-P.: How Easy Is Collision Search. New Results and Applications to DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 408–413. Springer, Heidelberg (1990)
9. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
10. Knuth, D.: The Art of Computer Programming, 2nd edn., vol. 2, p. 7. Addison-Wesley (1981)
11. Korf, R.E.: Finding Optimal Solutions to Rubik’s Cube Using Pattern Databases. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 1997, IAAI 1997, pp. 700–705. The MIT Press (1997)
12. Lucks, S.: Attacking Triple Encryption. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 239–253. Springer, Heidelberg (1998)
13. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
14. Merkle, R.C., Hellman, M.E.: On the Security of Multiple Encryption. Commun. ACM 24(7), 465–467 (1981)
15. Naya-Plasencia, M.: How to Improve Rebound Attacks. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 188–205. Springer, Heidelberg (2011)
16. Nivasch, G.: Cycle Detection Using a Stack. Inf. Process. Lett. 90(3), 135–140 (2004)
17. Schroeppel, R., Shamir, A.: A $T=O(2^{n/2})$, $S=O(2^{n/4})$ Algorithm for Certain NP-Complete Problems. SIAM J. Comput. 10(3), 456–464 (1981)
18. van Oorschot, P.C., Wiener, M.J.: Improving Implementable Meet-in-the-Middle Attacks by Orders of Magnitude. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 229–236. Springer, Heidelberg (1996)

Resistance against Iterated Attacks by Decorrelation Revisited^{*,**}

Aslı Bay, Atefeh Mashatan, and Serge Vaudenay

EPFL, Switzerland

{asli.bay,atefeh.mashatan,serge.vaudenay}@epfl.ch

Abstract. Iterated attacks are comprised of iterating adversaries who can make d plaintext queries, in each iteration to compute a bit, and are trying to distinguish between a random cipher C and the ideal random cipher C^* based on all bits. In EUROCRYPT '99, Vaudenay showed that a $2d$ -decorrelated cipher resists to iterated attacks of order d when iterations make almost no common queries. Then, he first asked what the necessary conditions are for a cipher to resist a non-adaptive iterated attack of order d . Secondly, he speculated that repeating a plaintext query in different iterations does not provide any advantage to a non-adaptive distinguisher. We close here these two long-standing open problems.

We show that, in order to resist non-adaptive iterated attacks of order d , decorrelation of order $2d - 1$ is not sufficient. We do this by providing a counterexample consisting of a cipher decorrelated to the order $2d - 1$ and a successful non-adaptive iterated attack of order d against it.

Moreover, we prove that the aforementioned claim is wrong by showing that a higher probability of having a common query between different iterations can translate to a high advantage of the adversary in distinguishing C from C^* . We provide a counterintuitive example consisting of a cipher decorrelated to the order $2d$ which can be broken by an iterated attack of order 1 having a high probability of common queries.

1 Introduction

Unlike asymmetric cryptography, in which the security of a cryptosystem is provably reduced to a mathematical problem and guaranteed by an intractability assumption, the focus in symmetric cryptography is often statistical cryptanalysis and, in the absence of a successful attack, a cryptosystem is believed to be secure. For instance, once the crypto community has spent enough time for scrutinizing a block cipher and has found no successful attacks against its full round version, the block cipher is believed to be secure. However, a different approach against block cipher cryptanalysis was pioneered by Nyberg [Nyb91] where she

* This work was supported in part by the European Commission through the ICT program under contract ICT-2007-216646 ECRYPT II.

** This work was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the SNF under grant number 5005-67322.

formalizes the notion of strength against differential cryptanalysis. Her work is followed by Chabaud and Vaudenay [CV94] formalizing the notion of strength against linear cryptanalysis.

Decorrelation Theory, introduced by Vaudenay [Vau99b, Vau03], encapsulates the techniques that guarantee the *provable* resistance of block ciphers against a wide range of statistical cryptanalysis, including the seminal differential and linear attacks, as well as their variants, for example the boomerang attack, truncated differential attacks, and impossible differential attacks. The beauty of this theory is that it can even guarantee resistance against some not-yet-discovered attacks that meet a certain broad criteria in the model presented by Luby and Rackoff [LR85, LR86]. They prove the security of Feistel schemes by assuming that the round function is random. However, their approach needs a very long secret key and is not suitable in practice. Carter and Wegman [CW79, CW81], on the other hand, use derandomization techniques for sampling pairwise independent numbers, which has inspired the notion of decorrelation in that it measures the pseudorandomness with smaller keys and examines its effects against the adversaries.

It is worth mentioning that *perfect* decorrelation of order d is equivalent to d -wise independence [Lub86]. Moreover, decorrelation of order d is also referred to as *almost d*-wise independence [NN90, AGM02]. Furthermore, the concept of decorrelation is somewhat related to the notion of pseudorandom functions and pseudorandom permutations except that we do not limit the time-complexity of the distinguisher, but only the number of queries are restricted.

The adversaries considered here can query d plaintexts and receive their corresponding ciphertexts, but are unlimited in terms of computational power. When these plaintext/ciphertext pairs are chosen randomly and independently from each other, we are dealing with d -limited *non-adaptive* adversaries, as opposed to d -limited *adaptive* adversaries. These adversaries give rise to distinguishers of order d , whether adaptive or otherwise, who are trying to distinguish between a random cipher C and the ideal random cipher C^* .

Several block ciphers have been proposed whose security is proven by decorrelation techniques, see for example DFC [PV98], NUT (n -Universal Transformation) families of block ciphers [Vau03]. Using similar techniques, Baignères and Finiasz propose two provably secure block ciphers to use in practice called the block cipher C [BF06a] and KFC [BF06b]. Decorrelation Theory has been used in other results as well, see for instance [Vau98b, Vau00, Vau99a, Vau99b, Vau98a, BV05].

Vaudenay [Vau03] shows how differential and linear attacks fit in the d -limited adversarial model by introducing *iterated* attacks, which are simply constructed by iterating a d -limited distinguisher (see Fig. 1). Linear and differential cryptanalysis can be formulated as non-adaptive iterated attacks of order 1 and order 2, respectively, and the boomerang attack is an adaptive (chosen plaintext and ciphertext) iterated attack of order 4. Moreover, he computes a bound on the advantage of the d -limited adversaries by decorrelation techniques in the Luby-Rackoff model. This result is expressed in the following theorem. Let C denote a

random cipher, i.e., the encryption based on a key which is modeled by a random variable, and C^* be the ideal random cipher, i.e., a uniformly distributed permutation. Moreover, $[C]^d$ is the d -wise distribution matrix of C (see Definition 3) and $\|\cdot\|_\infty$ is a matrix-norm (see Definition 4).

Theorem 1. [Vau03] Let C be a cipher on a message space of cardinality M such that $\|[C]^{2d} - [C^*]^{2d}\|_\infty \leq \varepsilon$, for some given $d \leq M/2$, where C^* is the ideal random cipher. Let us consider a non-adaptive iterated distinguisher of order d between C and C^* with n iterations. We assume that a set of d plaintexts is generated in each iteration in an independent way and following the same distribution. Moreover, we define δ as the probability that two sets drawn with this distribution have a nonempty intersection. Then, we bound the advantage of the adversary as $\text{Adv}_{\mathcal{A}_{\text{NAI}(d)}} \leq 5\sqrt[3]{\left(2\delta + \frac{5d^2}{2M} + \frac{3\varepsilon}{2}\right)n^2} + n\varepsilon$.

In this paper, we focus on the above result and address an open problem and disprove a claim that arise from Theorem 1. This theorem shows that, in order to resist a non-adaptive iterated attack of order d with seldom common queries, it is *sufficient* for a cipher to have the decorrelation of order $2d$. However, whether or not this is a necessary condition has not been addressed. Moreover, the bound given in the theorem can be interpreted to imply that, perhaps, a high probability δ of having a common query increases the bound of the attack. Despite this hint, Vaudenay in his EUROCRYPT '99 paper [Vau99b] speculates that having the same query to the oracle does not provide any advantage, but whether or not this is true has been left open. We will settle both of these open questions.

Firstly, we show that the decorrelation of order $2d - 1$ is not sufficient. We do this by proposing a counterexample consisting of a 3-round Feistel cipher which is decorrelated to the order $2d - 1$ and, *yet*, we are able to mount a successful non-adaptive iterated distinguisher of order d against it. Secondly, we propose another set of counterexamples where a higher probability of having common queries surprisingly increases the advantage of the distinguisher. In particular, we show that there is an iterated distinguisher of order 1 on a $2d$ -decorrelated cipher when the probability of having at least one query in common in any two iterations is high, which is counterintuitive. The rest of this paper is organized as follows. Section 2 summarizes some background. We dedicate Section 3 to our main contribution and address the aforementioned open problems.

2 Preliminaries

In this paper, F denotes a random function (or equivalently a function set up with a random key) from \mathcal{M}_1 to \mathcal{M}_2 and F^* denotes the ideal random function from \mathcal{M}_1 to \mathcal{M}_2 , that is, a function drawn uniformly at random among all $|\mathcal{M}_2|^{\mid\mathcal{M}_1\mid}$ functions on the given sets. Similarly, C denotes a random cipher (or equivalently, the encryption function set up with a random key) over \mathcal{M}_1 and C^* denotes the ideal random cipher over \mathcal{M}_1 , that is, a permutation drawn uniformly at random among all $|\mathcal{M}_1|!$ permutations. We use the following standard

notations: $|S|$ denotes the cardinality of the set S ; \mathcal{M}^d is the set of all sequences of d tuples over the set \mathcal{M} ; $\text{GF}(q)$ is the finite field with q elements; $\text{GF}(q)[x]$ is the set of polynomials defined over $\text{GF}(q)$; $\mathbb{E}(X)$ denotes the expected value of the random variable X ; $V(X)$ is the variance of the random variable X ; and $\text{gcd}(p(x), q(x))$ denotes the greatest common divisor of $p(x)$ and $q(x)$.

We consider the *Luby-Rackoff model* [LR85] in which an adversary \mathcal{A} is unbounded in terms of *computational power*. It is bounded to d number of plain-text/ciphertext queries to an oracle Ω implementing a random function. The goal of the adversary \mathcal{A} is to guess whether this function is drawn following the distribution of F (resp. C) or of F^* (resp. C^*). When queries are chosen randomly and at once, such an adversary is exactly a d -limited *non-adaptive* distinguisher. However, when queries are chosen depending on the answers to the previous queries, it is referred to as a d -limited *adaptive* distinguisher. In both distinguishers, the measure of success of \mathcal{A} is computed by means of the *advantage* of the adversary.

Definition 2. Let F_0 and F_1 be two random functions. The advantage of an adversary \mathcal{A} distinguishing F_0 from F_1 is defined by $\text{Adv}_{\mathcal{A}}(F_0, F_1) = |\Pr[\mathcal{A}(F_0) = 1] - \Pr[\mathcal{A}(F_1) = 1]|$.

Another measure is the *best advantage* of the distinguisher which is formulated as $\text{BestAdv}_{\zeta}(F_0, F_1) = \max_{\mathcal{A} \in \zeta} \text{Adv}_{\mathcal{A}}$. Here, the maximum is taken over adversaries in a class ζ . For instance, ζ can consist of all non-adaptive or all adaptive d -limited distinguishers, denoted by $\text{NA}(d)$ and $\text{A}(d)$, respectively, between F_0 and F_1 depending on \mathcal{A} being non-adaptive or adaptive.

Vaudenay also relates d -limited distinguishers with the two milestones of block cipher cryptanalysis, namely differential and linear cryptanalyses. These attacks are in fact members of a set of attacks called *iterated attacks* which includes many statistical attacks against block ciphers. Iterated attacks are basically constructed by iterating non-adaptive d -limited distinguishers and they are called non-adaptive iterated distinguishers of order d . We denote this distinguisher and its advantage by $\mathcal{A}_{\text{NAI}(d)}$ and $\text{Adv}_{\mathcal{A}_{\text{NAI}(d)}}$, respectively. Briefly, a function \mathcal{T} produces the binary outcome T_i of the d -limited distinguisher at iteration i . Another function Acc produces the final outcome based on (T_1, \dots, T_n) . The advantage of this Linear and differential cryptanalyses can be given as examples for non-adaptive iterated attacks of order 1 and 2, respectively. Boomerang attack is an adaptive iterated attack of order 4 (with chosen plaintexts and ciphertexts). Figure 1 gives a generic non-adaptive iterated distinguisher of order d with chosen plaintexts. We note that when referring to the advantage of an adversary in the rest of the paper, what we really mean is the best advantage.

The d -wise distribution matrix of a random function is defined next.

Definition 3. [Vau03] Let F be a random function from \mathcal{M}_1 to \mathcal{M}_2 . The d -wise distribution matrix $[F]^d$ of F is a $|\mathcal{M}_1|^d \times |\mathcal{M}_2|^d$ -matrix which is defined by $[F]_{(x_1, \dots, x_d), (y_1, \dots, y_d)}^d = \Pr_F[F(x_1) = y_1, \dots, F(x_d) = y_d]$, where $x = (x_1, \dots, x_d) \in \mathcal{M}_1^d$ and $y = (y_1, \dots, y_d) \in \mathcal{M}_2^d$.

Parameters: a complexity n , a distribution on X , a test \mathcal{T} , a set \mathcal{Acc}
Oracle: an oracle Ω implementing a permutation c

```

for  $i = 1$  to  $n$  do
    pick  $x = (x_1, \dots, x_d)$  at random
    get  $y = (c(x_1), \dots, c(x_d))$ 
    set  $T_i = 0$  or  $1$  such that  $T_i = \mathcal{T}(x, y)$ 
end for
if  $(T_1, \dots, T_n) \in \mathcal{Acc}$  then
    output  $1$ 
else
    output  $0$ 
end if
```

Fig. 1. A generic non-adaptive iterated distinguisher of order d

Afterwards, the *decorrelation of order d of a random function F* is computed by finding the distance $\mathcal{D}([F]^d, [F^*]^d)$ between its d -wise distribution matrix and the d -wise distribution matrix of the ideal random function F^* . The definition of \mathcal{D} indeed depends on whether the used distinguisher is adaptive or not. Moreover, if $\mathcal{D}([F]^d, [F^*]^d) = 0$, then F is a *perfect d -decorrelated function*. During the paper, we use a d -decorrelated function and a function decorrelated to the order d , interchangeably.

Definition 4. Let $M \in \mathbb{R}^{|\mathcal{M}_1|^d \times |\mathcal{M}_2|^d}$ be a matrix. Then, two matrix-norms are defined by $\|M\|_\infty = \max_{x_1, \dots, x_d} \sum_{y_1, \dots, y_d} |M_{(x_1, \dots, x_d), (y_1, \dots, y_d)}|$ and $\|M\|_A = \max_{x_1} \sum_{y_1} \dots \max_{x_d} \sum_{y_d} |M_{(x_1, \dots, x_d), (y_1, \dots, y_d)}|$.

Next, the advantage of the best distinguisher is computed.

Theorem 5 (Theorems 10 and 11 in [Vau03]). Let F and F^* be a random function and the ideal random function, respectively. The respective advantages of the best d -limited non-adaptive and adaptive distinguishers, $\mathcal{A}_{\text{NA}(d)}$ and $\mathcal{A}_{\text{A}(d)}$, are $\text{Adv}_{\mathcal{A}_{\text{NA}(d)}}(F, F^*) = \frac{1}{2}\|[F]^d - [F^*]^d\|_\infty$ and $\text{Adv}_{\mathcal{A}_{\text{A}(d)}}(F, F^*) = \frac{1}{2}\|[F]^d - [F^*]^d\|_A$.

Theorem 1 provides a bound for the advantage of a distinguisher against random permutations. We provide the following theorem for the case of random functions with a better bound for the advantage.

Theorem 6. Let F be a random function from \mathcal{M}_1 to \mathcal{M}_2 , where $d \leq |\mathcal{M}_1|/2$ and $|\mathcal{M}_2| = N$. Assume that F is decorrelated to the order $2d$ by $\|[F]^{2d} - [F^*]^{2d}\|_\infty \leq \varepsilon$, where F^* is the ideal random function. We consider a non-adaptive iterated distinguisher of order d between F and F^* with n iterations. We assume that a set of d plaintexts is generated in each iteration in an independent way and following the same distribution. Moreover, we define δ as the probability that two sets drawn with this distribution have a nonempty intersection. Then, we bound the advantage of the adversary as

$$\text{Adv}_{\mathcal{A}_{\text{NA}(d)}} \leq 5\sqrt[3]{\left(2\delta + \frac{3\varepsilon}{2}\right)n^2} + n\varepsilon.$$

The proof of this theorem can be found in Appendix A.

Theorem 7 (Theorem 21 in [Vau03] for $k = r = 3$). *Let F_1 , F_2 , and F_3 be three independent random functions over \mathcal{M}_1 such that $\|[F_i]^d - [F^*]^d\|_A \leq \varepsilon$ for $i \in \{1, 2, 3\}$, where F^* is the ideal random function. Consider a 3-round Feistel cipher C on \mathcal{M}_1^2 as in Fig. 2 having F_i 's as a round function in round i and the ideal cipher C^* . Then, we have $\|[C]^d - [C^*]^d\|_A \leq 3\varepsilon + 2d^2/\sqrt{M}$, where $M = |\mathcal{M}_1|^2$.*

The following results are useful for the rest of the paper.

Definition 8. *The trace $\text{Tr}(\beta)$ of an element $\beta \in \text{GF}(2^k)$, is defined as $\text{Tr}(\beta) = \beta + \beta^2 + \dots + \beta^{2^{k-1}}$.*

Note that it is well known that the trace is linear and the trace of an element of $\text{GF}(2^k)$ is either 0 or 1.

Lemma 9. *Hoeffding's bound [Hoe62]: Let X_1, X_2, \dots, X_n be independent random variables and $0 \leq X_i \leq 1$, for $i \in \{1, \dots, n\}$. Define $\bar{X} = \frac{1}{n} \sum_{i=0}^n X_i$ and let $\mu = \mathbb{E}(\bar{X})$. Then, for ε , $0 \leq \varepsilon \leq 1 - \mu$, we have $\Pr[\bar{X} \geq \mathbb{E}(\bar{X}) + \varepsilon] \leq e^{-2n\varepsilon^2}$ and $\Pr[\bar{X} \leq \mathbb{E}(\bar{X}) - \varepsilon] \leq e^{-2n\varepsilon^2}$. In addition, two-sided Hoeffding's bound is stated by $\Pr[|\bar{X} - \mathbb{E}(\bar{X})| \geq \varepsilon] \leq 2e^{-2n\varepsilon^2}$.*

3 Addressing the Two Open Problems

We deal with two open problems in Decorrelation Theory. In [Vau03], Vaudenay proposes Theorem 1 proving that the decorrelation of order $2d$ is sufficient for a cipher in order to resist a non-adaptive iterated attack of order d . We show here that the decorrelation of order $2d - 1$ is not sufficient by providing a counterexample. Secondly, the same theorem can be interpreted to imply that probability of having common queries increases the bound of the attack. To see the effect of this probability, we provide another counterexample showing that when this probability is high, the advantage of the distinguisher can be high.

We now provide a three round Feistel scheme C to be used in the following two subsections. This cipher C consists of three perfect κ -decorrelated functions F_1 , F_2 , and F_3 on $\mathcal{M}_1 = \text{GF}(q)$. Each F_i is defined by $F_i(x) = a_{\kappa-1}^i x^{\kappa-1} + a_{\kappa-2}^i x^{\kappa-2} + \dots + a_0^i$ over a finite field $\text{GF}(q)$, where $(a_{\kappa-1}^i, a_{\kappa-2}^i, \dots, a_0^i)$ is distributed uniformly at random over $\text{GF}(q)^\kappa$, for $i \in \{1, 2, 3\}$. According to Theorem 7, we have $\|[C]^\kappa - [C^*]^\kappa\|_A \leq 2\kappa^2/q$.

3.1 Decorrelation of Order $2d - 1$ Is NOT Sufficient

In this section, we are going to propose a counterexample on a 3-round Feistel cipher decorrelated to the order $2d - 1$. We are going to provide a successful non-adaptive iterated distinguisher of order d against this cipher showing that the decorrelation of order $2d - 1$ is not enough to resist a non-adaptive iterated distinguisher of order d .

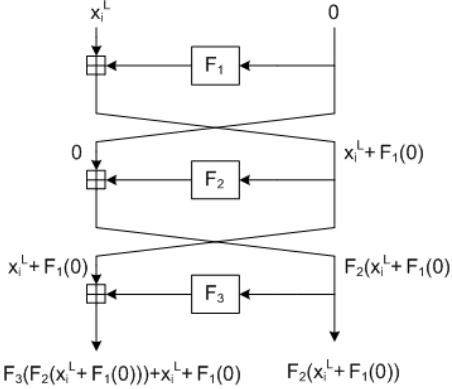


Fig. 2. The structure of the distinguisher for the 3-round Feistel cipher used in subsections 3.1 and 3.2

In our counterexample, we distinguish C for $\kappa = 2d - 1$. We show that C is not resistant to our non-adaptive iterated distinguisher of order d while $\| [C]^{2d-1} - [C^*]^{2d-1} \|_A \leq 2(2d - 1)^2/q$. First of all, we start with explaining the input distribution that adversary uses. Let (x_1, x_2, \dots, x_d) be the input tuple and (y_1, y_2, \dots, y_d) be the output tuple such that $C(x_i) = y_i$, where $1 \leq i \leq d$. We will pick plaintexts with specific properties. Every plaintext x_i can be written as $x_i = x_i^L \| x_i^R$, where x_i^L and x_i^R , both in $\text{GF}(q)$, are left and right halves of x_i . For each i , we let $x_i^R = 0$, i.e., $x_i = x_i^L \| 0$. Moreover, we choose a random c_1 and plaintexts $(x_1^L, x_2^L, \dots, x_d^L)$ satisfying $\prod_{i=1}^d x_i^L = c_0$ and

$$\sum_{i_1 \leq d} x_{i_1}^L = c_{d-1}, \quad \sum_{i_1 < i_2 \leq d} x_{i_1}^L x_{i_2}^L = c_{d-2}, \dots, \quad \sum_{i_1 < \dots < i_{d-1} \leq d} x_{i_1}^L x_{i_2}^L \cdots x_{i_{d-1}}^L = c_1,$$

where all c_i 's, except c_1 , are previously chosen constants and x_i^L 's are pairwise distinct. The left half is chosen by the algorithm is Fig. 3.

Given d and $c_0, c_1, \dots, c_{d-1} \in \text{GF}(q)$, this algorithm first constructs $h(x)$, where $h(x) = x^d - c_{d-1}x^{d-1} + \dots + (-1)^{d-2}c_2x^2 + (-1)^dc_0$. It picks a random c_1 from $\text{GF}(q)$ to construct $g(x)$ which is defined as $g(x) = h(x) + (-1)^{d-1}c_1x$. It checks if $g(x)$ divides $x^q - x$ in order to be sure that all roots are in $\text{GF}(q)$. Afterwards, it verifies that all roots are distinct. For this reason, it verifies that $g(x)$ and its derivative $g'(x)$ have no common divisors. Once these two conditions are satisfied, the algorithm outputs the roots of the polynomial g and gets the desired plaintext tuple.

The number of iterations in the algorithm to get the desired plaintext tuple is approximately $q^d / \binom{q}{d} \leq d!$, that is, one over the probability that a random monic polynomial of degree d has d distinct roots in $\text{GF}(q)$. To be more precise, since there are q possible irreducible factors of degree 1 in $\text{GF}(q)[x]$, we compute their d possible combinations in $\binom{q}{d}$ ways to construct polynomials of degree d and we divide it by the number of total monic polynomials of degree d which is q^d .

Input: $d, c_0, c_2, \dots, c_{d-1}, q$

Output: (x_1, \dots, x_d)

construct $h(x) = x^d - c_{d-1}x^{d-1} + \dots + (-1)^{d-2}c_2x^2 + (-1)^dc_0$

repeat

pick $c_1 \in \text{GF}(q)$ at random and construct $g(x) = h(x) + (-1)^{d-1}c_1x$

until $x^q \equiv x \pmod{g(x)}$ and $\text{gcd}(g(x), g'(x)) = 1$

find the roots (x_1, \dots, x_d) of $g(x)$ by using a factorization algorithm for polynomials

return (x_1, \dots, x_d)

Fig. 3. The algorithm to generate the left half of the plaintext tuples

Consider the encryption of each round when x_i 's are satisfying the above properties. After the first round, we have $0\|(x_i^L + F_1(0))$. Then, the output of the second round encryption is $(x_i^L + F_1(0))\|F_2(x_i^L + F_1(0))$. Finally, the corresponding ciphertext y_i will be $y_i = y_i^L\|y_i^R = (F_3(F_2(x_i^L + F_1(0))) + x_i^L + F_1(0))\|F_2(x_i^L + F_1(0))$. However, we will only be interested in the right part of the ciphertext, i.e., y_i^R , which can be seen as the output of a random polynomial function of degree at most $2d - 2$. More explicitly, since $y_i^R = F_2(x_i^L + F_1(0))$, we can write y_i^R as a function of x_i^L such that $F(x_i^L) = F_2(x_i^L + a_0^1)$. Obviously, each coefficient of the polynomial F is a function of coefficients of F_2 and the constant coefficient of F_1 , namely $f_i(a_0^1, a_{2d-2}^2, \dots, a_0^2)$. Since the coefficients of F depend on the coefficients of random functions F_1 and F_2 , F is also a random function.

Since we use the input distribution defined above, we can get some fixed bits by interpolating the right part of the output of the cipher, which is exactly the function F mentioned previously. In more detail, in every iteration we interpolate a polynomial r , which will appear in Equation 1. We expect that for the ideal random function F^* the constant coefficient of the polynomial r would be random, but for F it would be fixed. We prove this in Lemma 10. After formally writing this argument, by defining the test function \mathcal{T} and the acceptance set \mathcal{Acc} , we can distinguish the cipher from the ideal random cipher with only two iterations.

The distinguisher has d plaintext-ciphertext pairs in each iteration and F is a polynomial. Moreover, we know d points on F and we can use the underdetermined interpolation technique to determine F . We write F such that $F(x) = a_{2d-2}x^{2d-2} + a_{2d-3}x^{2d-3} + \dots + a_0$ over $\text{GF}(q)$, then we can determine F by

$$F(x) = r(x) + s(x)g(x). \quad (1)$$

Here, r is a unique polynomial of degree at most $d - 1$ which interpolates d given points, s is a polynomial of degree at most $d - 2$ over $\text{GF}(q)$ and g is a polynomial of degree d with the x_i^L 's as its roots $g(x) = (x - x_1^L) \cdots (x - x_d^L)$. Let $r(x) = r_{d-1}x^{d-1} + \dots + r_0$, $g(x) = x^d - c_{d-1}x^{d-1} + \dots + (-1)^{d-1}c_1 + (-1)^dc_0$, and $s(x) = s_{d-2}x^{d-2} + \dots + s_0$, where $r_i, c_j, s_k \in \text{GF}(q)$, $0 \leq i, j, k \leq d - 1$, and $0 \leq k \leq d - 2$. We note that $g(x)$ can be written as $g(x) = h(x) + (-1)^{d-1}c_1x$, where h is a fixed polynomial of degree d with zero coefficient for the term x .

Our aim is to get some fixed bits related to the function F in each iteration to have a distinguisher. The following lemma shows that, when the input distribution is picked as above, the constant coefficient r_0 of polynomial r is fixed in each iteration.

Lemma 10. *Let F be the polynomial of degree at most $2d - 2$ over $\text{GF}(q)$ as defined above. Let $x_1^L, \dots, x_d^L \in \text{GF}(q)$ be the left half of the plaintexts following the distribution above and $F(x_i^L) = y_i^R$, $0 \leq i \leq d$. Then, the constant coefficient r_0 of the polynomial r , which is obtained by the Lagrange interpolation of the given d points, is fixed in each iteration.*

Proof. We write $g(x) = h(x) + (-1)^{d-1}c_1x$ for (x_1^L, \dots, x_d^L) and for some $c_1 \in \text{GF}(q)$, where h is a fixed polynomial. Therefore, $F(x) = r(x) + s(x)g(x) = r(x) + s(x)(h(x) + (-1)^{d-1}c_1x)$ as in Equation 1. Moreover, $F(x) = r'(x) + s'(x)g'(x) = r'(x) + s'(x)(h(x) + (-1)^{d-1}c'_1x)$, for some $c'_1 \in \text{GF}(q)$. Hence,

$$\begin{aligned} & (r(x) + s(x)(h(x) + (-1)^{d-1}c_1x)) - (r'(x) + s'(x)(h(x) + (-1)^{d-1}c'_1x)) \\ &= \underbrace{r(x) - r'(x) + ((-1)^{d-1}(c_1s(x) - c'_1s'(x)))x}_{\text{Polynomial 1}} + \underbrace{(s(x) - s'(x))h(x)}_{\text{Polynomial 2}} = 0 \\ &\Rightarrow s(x) = s'(x). \end{aligned}$$

Polynomial 1 has degree at most $d - 1$ and Polynomial 2 has at least degree d (the degree of h), unless $s(x) = s'(x)$. Therefore, in order to have zero on the left side of the equation, $s(x) - s'(x)$ has to be zero. This shows that the polynomial s is a fixed polynomial, i.e., independent from the plaintext tuple that is queried. Therefore, we can write F as $F(x) = r(x) + s(x)(h(x) + (-1)^{d-1}c_1x)$, for fixed polynomials s and h and for some $c_1 \in \text{GF}(q)$. Hence, when $x = 0$, we have $F(0) = r(0) + s(0)h(0)$ which implies that $r_0 = a_0 - s_0h_0$ is always fixed. \square

We can now deduce that r_0 is always fixed and independent from the plaintext tuple due to the choice of plaintext tuple.

Now, we use Lemma 10 to construct a distinguisher between C and C^* . We denote the derived value of r_0 as a function $f((x_1, \dots, x_d), (y_1, \dots, y_d))$. Let D be a subset of distinguished values of $\text{GF}(q)$ with a given cardinality q/μ , where $\mu > 1$ is a positive divisor of q . Define the test function as

$$\mathcal{T}((x_1, \dots, x_d), (y_1, \dots, y_d)) = \begin{cases} 1, & \text{if } f((x_1, \dots, x_d), (y_1, \dots, y_d)) \in D, \\ 0, & \text{otherwise,} \end{cases}$$

and the acceptance set as

$$\mathcal{A}cc[t_1, \dots, t_n] = \begin{cases} 1, & \text{if } (t_1, \dots, t_n) \neq (0, \dots, 0), \\ 0, & \text{otherwise.} \end{cases}$$

All iterations will reply the same answer for the function F and a random answer for F^* . Let p (resp. p^*) be the probability that the distinguisher outputs 1 when

it is fed with F (resp. F^*). Hence, according to the acceptance set defined above, we get $p = \frac{1}{\mu}$ and $p^* = 1 - (1 - \frac{1}{\mu})^n$. If we consider $n = 2$, two iterations only, then the advantage of the distinguisher will be $|p - p^*| = |\frac{1}{\mu} - (1 - (1 - \frac{1}{\mu})^2)| = \frac{1}{\mu}(1 - \frac{1}{\mu})$ which is high. By this way, we can distinguish the cipher C from the ideal random cipher C^* by distinguishing the function F defining the right part of the output of the cipher C from the ideal random function F^* .

3.2 Assuming a Low δ Is NECESSARY

Theorem 1 shows that if a cipher is decorrelated to the order $2d$, then it resists to an iterated attack of order d . Moreover, it is speculated that a high probability δ of having a common query does not provide any advantage to the adversary. However, we give a counterintuitive example showing that there is an iterated distinguisher of order 1 on a $2d$ -decorrelated cipher when the probability of having at least one query in common in any two iterations is high. This shows that Theorem 1 is not universal and has a limit since δ may increase the bound of the distinguisher.

In our distinguisher, we use C depicted in Fig. 2 for $\kappa = 2d$ and $q = 2^k$. Note that Theorem 7 implies that $\|[C]^{2d} - [C^*]^{2d}\|_A \leq 8d^2/2^k$. We are going to prove that the random cipher C defined above is not resisting the iterated attack of order 1 when the set of plaintexts of adversary's choice is small. Let S be a set of plaintexts $S = \{x_1, x_2, \dots, x_{2d+2}\}$, where $x_i = x_i^L \| x_i^R$, x_L and x_R are left and right halves of x_i , respectively. These sets of plaintexts satisfy $x_i^R = 0$ and $\sum_{i=1}^{2d+2} (x_i^L)^j = 0$, $1 \leq j \leq 2d - 1$ and all x_i^L 's are pairwise distinct elements of $\text{GF}(2^k)$. The algorithm to generate the left part of the plaintexts in S is provided in Fig. 4.

This algorithm finds $p(x) = x^{2d+2} + ax^2 + bx + c$ with distinct roots in $\text{GF}(2^k)$, where $a, b, c \in \text{GF}(2^k)$. Note that $p(x)$ has roots of the form $\sum_{i=1}^{2d+2} (x_i^L)^j = 0$, $1 \leq j \leq 2d - 1$. This is proved in Appendix B for the case $n = 2d + 2$. The expected number of iterations in the algorithm can be computed heuristically as $2^{k(2d+2)} / \binom{2^k}{2d+2} \leq (2d+2)!$, that is, one over the probability that a random monic polynomial of degree $2d+2$ has $2d+2$ distinct roots in $\text{GF}(2^k)$. Since there are 2^k possible irreducible factors of degree 1 in $\text{GF}(2^k)[x]$, we compute their $2d+2$ possible combinations $\binom{2^k}{2d+2}$ to construct polynomials of degree $2d+2$ and we divide it by the total number of monic polynomials of degree $2d+2$ which is $2^{k(2d+2)}$. In each iteration, we pick one element of S at random. Since the adversary's choice of input set has $2d+2$ elements, we have $\delta = 1/(2d+2)$.

Like in the first counterexample, in order to distinguish this cipher C from the ideal random cipher C^* , we are going to take advantage of the right half of the output of the cipher. For this, at first we will show how to define the right part of the cipher to be a random function. When plaintexts x_i 's are satisfying the above properties, we have $y_i = y_i^L \| y_i^R = (F_3(F_2(x_i^L + F_1(0))) + x_i^L + F_1(0)) \| F_2(x_i^L + F_1(0))$. We only use the right part of the ciphertext which can be seen as an output of a random polynomial of degree at most $2d - 1$. In detail, since $y_i^R = F_2(x_i^L + F_1(0))$, y_i^R can be written as a function of x_i^L such that $F(x_i^L) = F_2(x_i^L +$

```

Input:  $k, d$ 
Output:  $(x_1, \dots, x_{2d+2})$ 
repeat
    pick  $a, b, c \in \text{GF}(2^k)$  at random and construct  $p(x) = x^{2d+2} + ax^2 + bx + c$ 
until  $x^{2^k} \equiv x \pmod{p(x)}$  and  $\gcd(p(x), p'(x)) = 1$ 
find the roots  $(x_1, \dots, x_{2d+2})$  of  $p(x)$  by using a factorization algorithm for polynomials
return  $(x_1, \dots, x_{2d+2})$ 

```

Fig. 4. The algorithm to generate the left half of the plaintexts in S

a_0^1). Obviously, each coefficient of the polynomial function F is a function of the coefficients of F_2 and the constant coefficient of F_1 , namely $f(a_0^1, a_{2d-1}^2, \dots, a_0^2)$. The fact that the coefficients of F are depending on the coefficients of random functions implies that F is also a random function. We then denote the function F as $F(x) = a_{2d-1}x^{2d-1} + a_{2d-2}x^{2d-2} + \dots + a_0$ over $\text{GF}(q)$.

The trick is that we distinguish the right half of the output of C which is in fact the polynomial F mentioned above. To explain how the distinguisher works briefly, when we consider that the plaintext space has the special form mentioned before, the right half of the cipher C which defines a polynomial F can be distinguished by using the trace. This is because, by the following Lemma, the sum of the trace of all elements in the set S behaves differently in this polynomial function F than in the ideal random function F^* allowing us to distinguish C from C^* . Now, we will propose this distinguishing property of the polynomial function F .

Lemma 11. *Let F be a random function and S be the input set defined as above. For $x_i = x_i^L \parallel 0 \in S$, $1 \leq i \leq 2d+2$, we have $\sum_{i=1}^{2d+2} \text{Tr}(F(x_i^L)) = 0$.*

Proof. Since $\sum_{i=1}^{2d+2} (x_i^L)^j = 0$, $1 \leq j \leq 2d-1$, we have

$$\begin{aligned} \sum_{i=1}^{2d+2} \text{Tr}(F(x_i^L)) &= \sum_{i=1}^{2d+2} \text{Tr}(a_{2d-1}(x_i^L)^{2d-1} + a_{2d-2}(x_i^L)^{2d-2} + \dots + a_0) = \\ &= \text{Tr}\left(a_{2d-1} \sum_{i=1}^{2d+2} (x_i^L)^{2d-1}\right) + \text{Tr}\left(a_{2d-2} \sum_{i=1}^{2d+2} (x_i^L)^{2d-2}\right) + \dots + \text{Tr}\left(a_0 \sum_{i=1}^{2d+2} (x_i^L)^0\right). \end{aligned}$$

Which is equal to 0 due the linearity of trace, the characteristic of this field being 2, and $\text{Tr}(0) = 0$. \square

We emphasize that Lemma 11 implies that there is an even number of $F(x_i^L)$'s which have $\text{Tr}(F(x_i^L)) = 1$ since $\sum_{i=1}^{2d+2} \text{Tr}(F(x_i^L)) = 0$. We use this property of F to distinguish the cipher C from the ideal random cipher C^* . In fact, like in the first counterexample, we distinguish the function F which is equivalent to distinguishing C .

Now, we explicitly explain how the iterated distinguisher of order 1 with n iterations works, where the input is distributed independently and identically

over the set S . We use the property of the polynomial function F which is stated in Lemma 11 in a way that in each iteration, we pick a plaintext x from S at random and compute the trace of $F(x^L)$, i.e., $t = \text{Tr}(F(x^L))$ since $F(x^L) = y^R$. Then, we compute the average $\bar{T} = \frac{1}{n}(t_1 + \dots + t_n)$, where t_i is the output of iteration i . We decide whether the oracle implements F (equivalently C) or F^* (equivalently C^*) by simply checking that the average value \bar{T} is in the specified set K which is determined according to the expected values of both \bar{T} and \bar{T}^* .

Lemma 12. *Assume that the plaintext set S has the above property. Then, depending on S , the expected value of \bar{T} takes any value from the set $S_1 = \{2m/(2d+2) | 0 \leq m \leq d+1\}$, and the expected value of \bar{T}^* takes any value from the set $S_2 = \{m/(2d+2) | 0 \leq m \leq 2d+2\}$.*

Proof. Assume that there are $2m$ number of x_i 's in S such that $F(x_i^L)$'s have $\text{Tr}(F(x_i^L)) = 1$ (from Lemma 11). Then, the number of x_i 's satisfying $\text{Tr}(F(x_i^L)) = 1$ in n iterations is expected to be $n(2m)/(2d+2)$, where $0 \leq m \leq d+1$. Therefore, the expected value of \bar{T} will be $2m/(2d+2)$, where $0 \leq m \leq d+1$. In a similar way, we can find the expected value of $\mathbb{E}(\bar{T}^*)$ for the ideal random function F^* . \square

Now, using Lemma 12, we define the acceptance set as

$$\text{Acc}[t_1, \dots, t_n] = \begin{cases} 1, & \text{if } \bar{T} \in K = \bigcup_{m=0}^{d+1} \left(\frac{2m}{2d+2} - \varepsilon, \frac{2m}{2d+2} + \varepsilon \right), \\ 0, & \text{otherwise.} \end{cases}$$

Typically, $\varepsilon = 1/(4d+4)$. Let p (resp. p^*) be the probability that the distinguisher outputs 1 when it is fed with F (resp. F^*). The following lemma states the bounds for both p and p^* .

Lemma 13. *We have $p \geq 1 - 2e^{-2n\varepsilon^2}$ and $p^* \leq \frac{1}{2} + e^{-2n\varepsilon^2}$.*

Proof. For the function F , according to the acceptance set defined previously, p is expressed by $p = \sum_{x \in S_1} \Pr[\mathbb{E}(\bar{T}) = x] \Pr[\bar{T} \in K | \mathbb{E}(\bar{T}) = x]$.

Since $\Pr[\bar{T} \in K | \mathbb{E}(\bar{T}) = x] \geq 1 - 2e^{-2n\varepsilon^2}$ by Hoeffding's bound from Lemma 9, we have $p \geq 1 - 2e^{-2n\varepsilon^2}$. Similarly, p^* is computed as $p^* = \sum_{x \in S_2} \Pr[\mathbb{E}(\bar{T}^*) = x] \Pr[\bar{T}^* \in K | \mathbb{E}(\bar{T}^*) = x]$. The computation of p^* is not straightforward, hence, we first compute the probability that each expected value of \bar{T}^* from S_2 occurs with probability $\Pr[\mathbb{E}(\bar{T}^*) = x] = \binom{2d+2}{x(2d+2)} 2^{-(2d+2)}$. In detail, we are picking $x(2d+2)$ places for 1's among $2d+2$ possible places and dividing the total number of possible choices which is 2^{2d+2} . Furthermore, the probabilities $\Pr[\bar{T}^* \in K | \mathbb{E}(\bar{T}^*) = x]$ are different according to the expected value of different \bar{T}^* . More explicitly, when $\mathbb{E}(\bar{T}^*) = 2m/(2d+2)$ for $0 \leq m \leq d+1$, we have $\Pr[\bar{T}^* \in K | \mathbb{E}(\bar{T}^*) = x] \leq 1$. Similarly, when $\mathbb{E}(\bar{T}^*) = (2m'+1)/(2d+2)$, $0 \leq m' \leq d$, we get $\Pr[\bar{T}^* \in K | \mathbb{E}(\bar{T}^*) = x] \leq 2e^{-2n\varepsilon^2}$ by Hoeffding's bound (Lemma 9). Then,

p^* can be computed as

$$\begin{aligned}
p^* &= \sum_{x \in \{2m/(2d+2) | 0 \leq m \leq d+1\}} \Pr[\mathbb{E}(\bar{T}^*) = x] \Pr[\bar{T}^* \in K | \mathbb{E}(\bar{T}^*) = x] \\
&+ \sum_{x \in \{(2m'+1)/(2d+2) | 0 \leq m' \leq d\}} \Pr[\mathbb{E}(\bar{T}^*) = x] \Pr[\bar{T}^* \in K | \mathbb{E}(\bar{T}^*) = x] \\
&\leq \sum_{x \in \{2m/(2d+2) | 0 \leq m \leq d+1\}} \binom{2d+2}{x(2d+2)} 2^{-(2d+2)} \\
&+ \sum_{x \in \{(2m'+1)/(2d+2) | 0 \leq m' \leq d\}} \binom{2d+2}{x(2d+2)} 2^{-(2d+2)} 2e^{-2n\varepsilon^2}.
\end{aligned}$$

Note that the sum of even and odd indices of binomial coefficients are $\sum_{i \geq 0} \binom{n}{2i} = 2^{n-1}$ and $\sum_{i \geq 0} \binom{n}{2i+1} = 2^{n-1}$, respectively. Hence, we have

$$\sum_{x \in \{2m/(2d+2) | 0 \leq m \leq d+1\}} \binom{2d+2}{x(2d+2)} 2^{-(2d+2)} = \frac{1}{2}.$$

Then, we get

$$\sum_{x \in \{(2m'+1)/(2d+2) | 0 \leq m' \leq d\}} \binom{2d+2}{x(2d+2)} 2^{-(2d+2)} 2e^{-2n\varepsilon^2} \leq \frac{1}{2} 2e^{-2n\varepsilon^2} = e^{-2n\varepsilon^2}.$$

Therefore, we get $p^* \leq \frac{1}{2} + e^{-2n\varepsilon^2}$. \square

Finally, the advantage of the distinguisher is computed as

$$|p - p^*| \geq \left| \left(1 - 2e^{-2n\varepsilon^2}\right) - \left(\frac{1}{2} + e^{-2n\varepsilon^2}\right) \right| = \left| \frac{1}{2} - 3e^{-2n\varepsilon^2} \right|.$$

When the distinguisher has a large number of iterations, we have $|p - p^*| \approx 1/2$ which is quite high. This way we manage to distinguish the cipher C from the ideal random cipher C^* . Hence, *in specific situations*, having common queries *can* increase the advantage. Essentially, if the images of $2d+2$ points sum to zero, by taking $\varepsilon = 1/(4d+4)$ and $n \approx \Omega(d^2)$ we obtain an efficient iterated distinguisher of order 1. This could be used to transform Integral/Square/Saturation attacks to this kind of distinguisher (with a squared number of inputs).

As a final remark, in Decorrelation Theory, Vaudenay considers block ciphers in the context of deterministic symmetric key encryption. Therefore, for some input distribution, the probability δ that two iterations have at least one query in common can be high. However, if we consider symmetric-key probabilistic encryption, then δ will always be small. This is because, in this scheme, the oracle picks the random coins, and even if the same plaintext is picked by the adversary, the random coins picked by the oracle for two plaintexts would be different which causes two different inputs to the encryption. This implies that high δ is not a threat when we consider the probabilistic encryption.

4 Conclusion and Future Work

We settled an open problem and disproved a claim, both of which are raised by the EUROCRYPT '99 work of Vaudenay in Decorrelation Theory. In particular, we proved that in order for a cipher C to resist a non-adaptive iterated attack of order d , it is not sufficient to have a decorrelation of order $2d - 1$. We showed this by providing a cipher decorrelated to the order $2d - 1$ and a successful non-adaptive iterated attack against it which has order d . Hence, we concluded that the minimal order of decorrelation to ensure resistance is $2d$. Furthermore, we illustrated that when the probability of having a common query between different iterations increases, the advantage of the distinguisher *can* increase.

Our counterexamples comprise of non-adaptive distinguishers. One could also investigate whether or not a similar result holds for the case of adaptive adversaries. Moreover, the adversaries we consider make plaintext queries and receive the corresponding ciphertexts. A different adversarial model can also be considered where the adversary can make ciphertext queries together with plaintext queries.

References

- [AGM02] Alon, N., Goldreich, O., Mansour, Y.: Almost k-wise independence versus k-wise independence. *Electronic Colloquium on Computational Complexity (ECCC)* 9(048) (2002)
- [BF06a] Baignères, T., Finiasz, M.: Dial C for Cipher. In: Biham, E., Youssef, A.M. (eds.) *SAC 2006*. LNCS, vol. 4356, pp. 76–95. Springer, Heidelberg (2007)
- [BF06b] Baignères, T., Finiasz, M.: KFC - The Krazy Feistel Cipher. In: Lai, X., Chen, K. (eds.) *ASIACRYPT 2006*. LNCS, vol. 4284, pp. 380–395. Springer, Heidelberg (2006)
- [BV05] Baignères, T., Vaudenay, S.: Proving the Security of AES Substitution-Permutation Network. In: Preneel, B., Tavares, S. (eds.) *SAC 2005*. LNCS, vol. 3897, pp. 65–81. Springer, Heidelberg (2006)
- [CV94] Chabaud, F., Vaudenay, S.: Links between Differential and Linear Cryptanalysis. In: De Santis, A. (ed.) *EUROCRYPT 1994*. LNCS, vol. 950, pp. 356–365. Springer, Heidelberg (1995)
- [CW79] Carter, L., Wegman, M.N.: Universal Classes of Hash Functions. *Journal of Computer and System Sciences* 18(2), 143–154 (1979)
- [CW81] Carter, L., Wegman, M.N.: New Hash Functions and Their Use in Authentication and Set Equality. *Journal of Computer and System Sciences* 22(3), 265–279 (1981)
- [Hoe62] Hoeffding, W.: Probability Inequalities For Sums of Bounded Random Variables (1962)
- [LR85] Luby, M., Rackoff, C.: How to Construct Pseudo-random Permutations from Pseudo-random Functions. In: Williams, H.C. (ed.) *CRYPTO 1985*. LNCS, vol. 218, p. 447. Springer, Heidelberg (1986)
- [LR86] Luby, M., Rackoff, C.: Pseudo-random Permutation Generators and Cryptographic Composition. In: Hartmanis, J. (ed.) *STOC*, pp. 356–363. ACM (1986)

- [Lub86] Luby, M.: A Simple Parallel Alogarithm for the Maxial Independent Set Problem. *SIAM J. Comput.* 15(4), 1036–1053 (1986)
- [NN90] Naor, J., Naor, M.: Small-bias probability spaces: Efficient constructions and applications. In: Ortiz, H. (ed.) STOC, pp. 213–223. ACM (1990)
- [Nyb91] Nyberg, K.: Perfect Nonlinear S-Boxes. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 378–386. Springer, Heidelberg (1991)
- [PV98] Poupard, G., Vaudenay, S.: Decorrelated Fast Cipher: An AES Candidate Well Suited for Low Cost Smart Card applications. In: Quisquater, J.-J., Schneier, B. (eds.) CARDIS 1998. LNCS, vol. 1820, pp. 254–264. Springer, Heidelberg (2000)
- [Vau98a] Vaudenay, S.: Feistel Ciphers with L_2 -Decorrelation. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 1–14. Springer, Heidelberg (1999)
- [Vau98b] Vaudenay, S.: Provable Security for Block Ciphers by Decorrelation. In: Morvan, M., Meinel, C., Krob, D. (eds.) STACS 1998. LNCS, vol. 1373, pp. 249–275. Springer, Heidelberg (1998)
- [Vau99a] Vaudenay, S.: On Probable Security for Conventional Cryptography. In: Song, J.S. (ed.) ICISC 1999. LNCS, vol. 1787, pp. 1–16. Springer, Heidelberg (2000)
- [Vau99b] Vaudenay, S.: Resistance Against General Iterated Attacks. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 255–271. Springer, Heidelberg (1999)
- [Vau00] Vaudenay, S.: Adaptive-Attack Norm for Decorrelation and Super-Pseudorandomness. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 49–61. Springer, Heidelberg (2000)
- [Vau03] Vaudenay, S.: Decorrelation: A Theory for Block Cipher Security. *J. Cryptology* 16(4), 249–286 (2003)

A The Proof of Theorem 6

This proof is exactly the same as the proof of Theorem 1 [Vau03] except for the computation of $V(T(F^*))$ which results in a tighter bound for the distinguisher. Given $F = f$ (resp. $F^* = f$), let $T(f)$ be the probability that test function \mathcal{T} outputs 1 when $(X, f(X))$ is its input, i.e., $T(f) = \mathbb{E}_X(\mathcal{T}(X, f(X)))$. Let p (resp. p^*) be the probability that the distinguisher outputs 1, i.e., $p = \Pr_F[(T_1(F), \dots, T_n(F)) \in \mathcal{A}cc]$, where $\mathcal{A}cc$ is the acceptance set and $T_i(F)$ (resp. $T_i(F^*)$) is the output of iteration i .

Since $T_i(F)$'s are all independent with the same expected value $T(F)$ which only depends on F , we get

$$p = \mathbb{E}_F \left(\sum_{(t_1, \dots, t_n) \in \mathcal{A}cc} T(F)^{t_1 + \dots + t_n} (1 - T(F))^{n - (t_1 + \dots + t_n)} \right).$$

Then, p can be rewritten as $p = \sum_{i=0}^n a_i \mathbb{E}_F(T(F))^i (1 - T(F))^{n-i}$ for some integers a_i such that $0 \leq a_i \leq \binom{n}{i}$. Therefore, the advantage $|p - p^*|$ is maximal when all a_i 's are either 0 or $\binom{n}{i}$ depending on the distributions $T(F)$ and $T(F^*)$. This implies that the acceptance set of the best distinguisher is of the form $\mathcal{A}cc = \{(t_1, \dots, t_n) | \sum_{i=1}^n t_i \in \mathcal{B}\}$ for some set $\mathcal{B} \subseteq \{0, \dots, n\}$. Therefore, we have $p = \mathbb{E}_F(s(T(F)))$ where $s(x) = \sum_{x \in \mathcal{B}} \binom{n}{i} x^i (1 - x)^{n-i}$.

Recall that $|s(T(F)) - s(T(F^*))| \leq 2n|T(F) - T(F^*)|$. In addition, we have $|\mathbb{E}_F(T(F)) - \mathbb{E}_{F^*}(T(F^*))| \leq \varepsilon/2$, $|\mathbb{E}_F(T^2(F)) - \mathbb{E}_{F^*}(T^2(F^*))| \leq \varepsilon/2$ and $|V(T(F)) - V(T(F^*))| \leq 3\varepsilon/2$. Then, the advantage of the distinguisher is $|p - p^*| = |\mathbb{E}(T(F)) - \mathbb{E}(T(F^*))| \leq \mathbb{E}(|T(F) - T(F^*)|)$. By applying Tchebichev's inequality for both $T(F)$ and $T(F^*)$ which is $\Pr[|T(F) - \mathbb{E}(T(F))| > \lambda] \leq V(T(F))/\lambda^2$ for any $\lambda > 0$ (same for $T(F^*)$), we get

$$|p - p^*| \leq 5 \left((2V(T(F^*)) + \frac{3\varepsilon}{2}) n^2 \right)^{\frac{1}{3}} + n\varepsilon, \quad (2)$$

when $\lambda = \left(\frac{2V(T(F^*)) + \frac{3\varepsilon}{2}}{n} \right)^{\frac{1}{3}}$. Up to this point, the proof was the same with the proof of Theorem 1. However, the bound for $V(T(F^*))$ is different from the bound for $V(T(C^*))$, where C^* is the ideal random cipher. We get $V(T(F^*))$ to be equal to $\sum_{(x,y),(x',y') \in \mathcal{T}} \Pr[X = x] \Pr[X = x'] \left(\Pr_{F^*}[(x, x') \xrightarrow{F^*} (y, y')] - \Pr_{F^*}[x \xrightarrow{F^*} y] \Pr_{F^*}[x' \xrightarrow{F^*} y'] \right)$.

In order to bound this sum, we divide pairs (x, x') into two groups of pairs such that the first group has no common queries, i.e., $\forall i, j \ x_i \neq x'_j$, but the second one has. As a remark, we assume that the adversary does not pick the same query in a single iteration, i.e., $x_i \neq x_j$, when $i \neq j$. Since all x_i are distinct in $x = (x_1, \dots, x_d)$, then $[F^*]_{(x_1, \dots, x_d), (y_1, \dots, y_d)}^d = \prod_{i=1}^d \Pr[F^*(x_i) = y_i] = N^{-d}$. When inputs x and x' have no common

queries, then $[F^*]_{(x_1, \dots, x_d, x'_1, \dots, x'_d), (y_1, \dots, y_d, y'_1, \dots, y'_d)}^{2d} = \prod_{i=1}^d \Pr[F^*(x_i) = y_i] \prod_{i=1}^d \Pr[F^*(x'_i) = y'_i] = N^{-2d}$. Therefore, when input tuples x and x' have no common queries, the sum will be 0. Otherwise, when the plaintext tuples x and x' have common queries with probability δ , then the sum over all these plaintext tuples will be less than δ . Hence, we have $V(T(F^*)) \leq \delta$. When we substitute δ for $V(T(F^*))$ in Inequality 2, we get $|p - p^*| \leq 5\sqrt[3]{(2\delta + \frac{3\varepsilon}{2})n^2 + n\varepsilon}$.

B A Required Lemma for Subsection 3.2

Lemma 14. *Let f be a polynomial of the form $f(x) = x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-2}x^2 + a_{n-1}x + a_n$ over $\text{GF}(2^k)$ and x_1, x_2, \dots, x_n be its roots. If $a_1 = a_2 = \dots = a_{n-3} = 0$, then its roots satisfy $s_k = \sum_{i=1}^n x_i^j = 0$, where $1 \leq j \leq n-3$ and $n \geq 4$.*

Proof. First, we recall the *Newton formulas*. Let f be a polynomial of the form $f(x) = x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-2}x^2 + a_{n-1}x + a_n$ with the roots x_1, x_2, \dots, x_n so that $f(x) = (x - x_1)(x - x_2) \cdots (x - x_n)$ over a ring R . Then, we define the elementary symmetric functions of the roots as $\sum_{1 \leq i \leq n} x_i = -a_1, \sum_{1 \leq i < j \leq n} x_i x_j = a_2, \sum_{1 \leq i < j < k \leq n} x_i x_j x_k = -a_3, \dots$, and $x_1 x_2 \cdots x_n = (-1)^n a_n$.

In addition, k^{th} power sums of the roots is defined as $s_k = \sum_{1 \leq i \leq n} x_i^k$. Then, *Newton formulas* give recursion relations between a_i 's and s_i 's as $s_1 + a_1 = 0, s_2 + a_1 s_1 + 2a_2 = 0, \dots, s_n + a_1 s_{n-1} + a_2 s_{n-2} + \dots + n a_n = 0, s_{n+1} + a_1 s_{n-1} + a_2 s_{n-2} + \dots + s_1 a_n = 0$.

Note that Newton formulas are valid over finite fields. Now, if we assume that $a_1 = a_2 = \dots = a_{n-3} = 0$, then according to the Newton formulas given above we will show that $s_k = 0$ for $1 \leq k \leq n-3$. Since $s_1 + a_1 = 0$ and $a_1 = 0$, we have $s_1 = 0$. By induction, assume that $s_1 = s_2 = \dots = s_{k-1} = 0$ and $a_1 = a_2 = \dots = a_k = 0$, then we have $s_k = -(a_1 s_{k-1} + a_2 s_{k-2} + \dots + k a_k) = 0$. Therefore, the polynomial $f(x) = x^n + a_{n-2}x^2 + a_{n-1}x + a_n$ satisfies $s_k = \sum_{i=1}^n x_i^j = 0$, where $1 \leq j \leq n-3$ and $n \geq 4$. \square

Secure Identity-Based Encryption in the Quantum Random Oracle Model*

Mark Zhandry

Stanford University, USA

Abstract. We give the first proof of security for an identity-based encryption scheme in the *quantum* random oracle model. This is the first proof of security for *any* scheme in this model that requires no additional assumptions. Our techniques are quite general and we use them to obtain security proofs for two random oracle hierarchical identity-based encryption schemes and a random oracle signature scheme, all of which have previously resisted quantum security proofs, even using additional assumptions. We also explain how to remove the extra assumptions from prior quantum random oracle model proofs. We accomplish these results by developing new tools for arguing that quantum algorithms cannot distinguish between two oracle distributions. Using a particular class of oracle distributions, so called *semi-constant* distributions, we argue that the aforementioned cryptosystems are secure against quantum adversaries.

Keywords: Quantum, Random Oracle, IBE, Signatures.

1 Introduction

While quantum computation is not yet viable, Shor [Sho97] showed that when fully realized, quantum computers will break most of the cryptosystems used today, namely those based on the difficulty factoring and the discrete log problem. This has sparked the field of post-quantum cryptography, the search for classical systems secure against quantum adversaries. To be secure in this setting, a system must have an underlying difficult problem for quantum computers, as well as a security reduction showing how to solve this problem using a quantum adversary that breaks the system. Problems based on lattices have so far resisted quantum attacks, and there is a considerable amount of literature on lattice-based cryptosystems. However, random oracle security proofs for these constructions typically only consider classical adversaries, thus failing to show security in the quantum world.

The random oracle model [BR93] is one area where many classical proofs lack quantum equivalents. This model is of interest because random oracle schemes tend to be more efficient than their standard model counterparts. Consequently, the most efficient lattice-based schemes are often constructed in the random oracle model. For example, Gentry, Peikert, and Vaikuntanathan [GPV08]

* Full version available at <http://eprint.iacr.org/2012/076/>

show how to construct signatures and identity-based encryption (IBE). Cash et al. [CHKP10] and Agrawal, Boneh, and Boyen [ABB10] give hierarchical IBE schemes both in the standard model and the random oracle model, with the random oracle constructions being more efficient than the corresponding standard model schemes. Gordon, Katz, and Vaikuntanathan [GKV10] give a group signature scheme, while Boneh and Freeman [BF11] demonstrate a homomorphic signature scheme, all in the random oracle model.

All of these papers prove security against adversaries with classical access to the random oracle. However, when the scheme is instantiated, the random oracle is replaced with a hash function H , which a quantum adversary may evaluate on a quantum superposition of inputs. To model this ability, it is necessary to allow a quantum adversary to make quantum queries to the random oracle. We call this model the quantum random oracle model.

Proving security in the quantum random oracle model presents many challenges. Among them is the difficulty of efficiently simulating the random oracle. In the classical case, the random oracle is simulated on the fly, only generating randomness as needed. In the quantum setting, the adversary can query the oracle on an exponential superposition of inputs. Therefore, to make even the first query look random to the adversary, it would seem that we would need exponential randomness.

In addition to this difficulty, there are classical random oracle techniques that do not make sense if the adversary has quantum access to the random oracle. One such technique is that of Bellare and Rogaway [BR93] for proving the security of the Full Domain Hash signature scheme. In this technique, the reduction algorithm is given a challenge c to solve, and randomly guesses which oracle query the adversary will use to break the scheme. The algorithm embeds c into the response for this query, and if both the guess is correct and the adversary breaks the scheme, then the algorithm will be able to solve c .

In the quantum setting, this argument no longer applies because each oracle query might be over a superposition of exponentially many inputs. We could choose a random query and plug c into all outputs for that query, but this will not look like a random oracle to the adversary. Alternatively, we could choose one oracle input and plug c into the corresponding output for all queries. However, our chance of guessing correctly will then be exponentially small.

1.1 Our Contributions

We resolve some of the issues outlined above by giving a quantum analog of the technique of Bellare and Rogaway [BR93] and demonstrating how to simulate a random oracle without any additional computational assumptions. Specifically, we:

- Describe new ways to argue that quantum algorithms cannot distinguish between two distributions of oracles.
- Apply these techniques to a new type of distribution of oracles, which we call *semi-constant* distributions, showing that they cannot be distinguished from random oracles.

- Use our results on semi-constant distributions to prove that the random oracle IBE scheme of Gentry et al. [GPV08] is secure against quantum adversaries. The basic idea is to plug the challenge c into a small fraction of inputs to the oracle, making the oracle seen by the adversary a semi-constant distribution. The adversary thus behaves as though the oracle is random. If the adversary happens to use any of the inputs in this fraction, we are able to solve c .
- Show that this technique is general by applying it to the random oracle hierarchical IBE schemes of Cash et al. [CHKP10] and Agrawal et al. [ABB10].
- Prove that the generic Full Domain Hash signature scheme [BR93] is secure against quantum adversaries, though this remains a theoretical result until a trapdoor permutation is found that is secure against quantum adversaries.
- Use our techniques and k -wise independent functions to solve the problem of simulating quantum random oracles, thus making the above results unconditional.

1.2 Related Work

Quantum random oracles have been used in several prior works. For example, Bennett et al. [BBBV97] prove several quantum complexity results, including a proof that quantum computers cannot solve all of NP, relative to a random oracle. In a cryptographic setting, quantum random oracles have been used by Aaronson [Aar09] to construct quantum money and by Brassard and Salvail [BS08] and Brassard et al. [BHK⁺11] to construct quantum analogs of Merkle’s Puzzles.

Boneh et al. [BDF⁺11] give a random oracle scheme that is secure against quantum adversaries with classical access to the oracle, but is insecure once the adversary has quantum access to the oracle. Despite this result, they show that there are circumstances in which a classical random oracle security reduction can also be used in the quantum setting. However, their proof techniques do not apply to the schemes analyzed in this paper. Additionally, their proof simulates the random oracle using a pseudorandom function (PRF) that is secure against quantum adversaries, hence making their results conditional on the existence of a so-called quantum-secure PRF. We note that Zhandry [Zha12b] shows that such PRFs can be built from quantum-secure pseudorandom generators, which can, in turn, be built from lattices.

There has also been progress toward converting classical security proofs into quantum proofs outside of the random oracle model. For instance, Unruh [Unr10] shows that classical statistical security in Canetti’s universal composability (UC) model implies quantum statistical security. Hallgren, Smith, and Song [HSS11] give a two-party protocol that is quantum computationally secure in the UC.

2 Preliminaries

A function $\epsilon(n)$ is negligible if it is non-negative and smaller than any inverse polynomial. That is, for any polynomial $p(n)$, $\epsilon(n) < 1/p(n)$ for all sufficiently large n .

A probabilistic polynomial time (PPT) algorithm is a classical randomized algorithm that runs in time polynomial in the size of its input. We also call such algorithms efficient.

2.1 Weight Assignments

A weight assignment on a set \mathcal{X} is a function $D : \mathcal{X} \rightarrow \mathbb{R}$ such that $\sum_{x \in \mathcal{X}} D(x) = 1$. We sometimes write $\Pr_D[\text{event}]$ to represent the sum of the weights of all outcomes consistent with that event. A distribution on X is a weight-assignment D such that $D(x) \geq 0$ for all $x \in \mathcal{X}$. If D is a distribution, way that x occurs with probability $D(x)$. Let $U_{\mathcal{X}}$ denote the uniform distribution over \mathcal{X} . That is, $U_{\mathcal{X}}(x) = 1/|\mathcal{X}|$. When the set \mathcal{X} is clear, we may omit the subscript.

We define the distance between two weight assignments D_1 and D_2 over a set \mathcal{X} as

$$|D_1 - D_2| = \sum_{x \in \mathcal{X}} |D_1(x) - D_2(x)|$$

If $|D_1 - D_2| \leq \epsilon$, we say that D_1 and D_2 are ϵ -close.

Given a set of weight assignments D_y over \mathcal{X} , indexed by $y \in \mathcal{Y}$, and a weight assignment D over \mathcal{Y} , we can define a weight assignment D' over \mathcal{X} where

$$D'(x) = \sum_{y \in \mathcal{Y}} D(y)D_y(x)$$

We write $D' = \sum_{y \in \mathcal{Y}} D(y)D_y$. Say each of the D_y s are actually distributions. Given two weight assignments D_1 and D_2 over \mathcal{Y} , define $D'_1 = \sum_{y \in \mathcal{Y}} D_1(y)D_y$ and $D'_2 = \sum_{y \in \mathcal{Y}} D_2(y)D_y$. Then:

$$\begin{aligned} |D'_1 - D'_2| &= \sum_{x \in \mathcal{X}} \left| \sum_{y \in \mathcal{Y}} (D_1(y) - D_2(y))D_y(x) \right| \\ &\leq \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} |(D_1(y) - D_2(y))D_y(x)| \\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} |D_1(y) - D_2(y)|D_y(x) \\ &= \sum_{y \in \mathcal{Y}} |D_1(y) - D_2(y)| = |D_1 - D_2| \end{aligned}$$

Consider the set of functions $H : \mathcal{X} \rightarrow \mathcal{Y}$ for sets \mathcal{X} and \mathcal{Y} , denoted by $\mathcal{H}_{\mathcal{X}, \mathcal{Y}}$. Consider a weight assignment D on $\mathcal{H}_{\mathcal{X}, \mathcal{Y}}$. Let $\mathcal{W} \subseteq \mathcal{X}$. We define the marginal weight assignment $D_{\mathcal{W}}$ of D on $\mathcal{H}_{\mathcal{W}, \mathcal{Y}}$ where the weight of a function $H_{\mathcal{W}} : \mathcal{W} \rightarrow \mathcal{Y}$ is equal to the sum of the weights of all $H \in \mathcal{H}_{\mathcal{X}, \mathcal{Y}}$ that agree with $H_{\mathcal{W}}$ on \mathcal{W} . In other words,

$$D_{\mathcal{W}}(H_{\mathcal{W}}) = \Pr_D[H(w) = H_{\mathcal{W}}(w) \forall w \in \mathcal{W}]$$

We call two weight assignments D_1 and D_2 on $\mathcal{H}_{\mathcal{X}, \mathcal{Y}}$ k -wise equivalent if for all $\mathcal{W} \subseteq \mathcal{X}$ of size k , the marginal weight assignments $D_{1, \mathcal{W}}$ and $D_{2, \mathcal{W}}$ (of D_1 and D_2) over $\mathcal{H}_{\mathcal{W}, \mathcal{Y}}$ are identical.

2.2 Quantum Computation

A quantum algorithm is an algorithm executed on a quantum computer that produces a classical output. Most of the background in quantum computation needed to understand this paper is for the proof of Theorem 3.1. Since this proof appears in the full version [Zha12a], we present the necessary background there. In the meantime, we recall a couple basic facts about quantum computation, and refer the reader to [NC00] for a more thorough discussion.

Fact 1. *Any classical computation can be implemented on a quantum computer.*

Fact 2. *Any function that has an efficient classical algorithm computing it can be implemented efficiently as a quantum-accessible oracle.*

Fact 3. *Given a quantum algorithm A with oracle access to an oracle O , each oracle O defines a probability distribution of the outputs of A . Hence, any weight assignment of oracles leads to a weight assignment of outputs of A , and if two weight assignments D_1 and D_2 are a distance ϵ apart, the weight assignment of the outputs of A under these distributions are a distance at most ϵ apart.*

2.3 Cryptographic Primitives

Here we briefly outline a few cryptographic primitives, and refer to the full version [Zha12a] for details. All primitives depend on a security parameter n .

An encryption scheme E is a triple of PPT algorithms $(E.\text{Gen}, E.\text{Enc}, E.\text{Dec})$, where $E.\text{Gen}(1^n)$ generates secret/public keys (sk, pk) , $E.\text{Enc}_{\text{pk}}$ encrypts a message, and $E.\text{Dec}_{\text{sk}}$ decrypts a ciphertext. We use the indistinguishability under chosen plaintext attack (IND-CPA) notion of security [GM84].

An identity-based encryption (IBE) scheme IBE is a 4-tuple of PPT algorithms $(\text{IBE}.\text{Gen}, \text{IBE}.\text{Extract}, \text{IBE}.\text{Enc}, \text{IBE}.\text{Dec})$ where $\text{IBE}.\text{Gen}(1^n)$ generates master secret/public keys (msk, mpk) , $\text{IBE}.\text{Extract}_{\text{msk}}$ generates secret keys for given identities, $\text{IBE}.\text{Enc}_{\text{mpk}}$ encrypts a message to an identity, and $\text{IBE}.\text{Dec}$ decrypts a ciphertext sent to an identity by using the corresponding secret key. We use the indistinguishability under chosen plaintext attack (IND-ID-CPA) notion of security [BF01].

A signature scheme $S = (S.\text{Gen}, S.\text{Sign}, S.\text{Ver})$ is a triple of PPT algorithms where $S.\text{Gen}(1^n)$ generates secret/public keys (sk, pk) , $S.\text{Sign}_{\text{sk}}$ signs a message, and $S.\text{Ver}_{\text{pk}}$ verifies a signature. We use the existential unforgeability under chosen message attack (UF-CMA) notion of security [GMR88].

A pre-image sampleable function (PSF) is a quadruple of algorithms $F = (F.\text{Gen}, F.\text{Sample}, f, f^{-1})$ where $F.\text{Gen}$ generates private/public keys (sk, pk) , f_{pk} is a function, $F.\text{Sample}$ samples x from a distribution D such that $f_{\text{pk}}(x)$ is uniform, and $f_{\text{sk}}^{-1}(y)$ samples from D conditioned on $f_{\text{pk}}(x) = y$. For security, we use the notion of one-wayness.

A trapdoor permutation (TDP) is a special case of a PSF where f is bijective and $F.\text{Sample}$ simply returns a random element in the domain of f_{pk} . Since $F.\text{Sample}$ is already defined, we omit it from the specification.

2.4 Random Oracle Model

In the Random Oracle Model, we assume the existence of a random function H , and give all parties oracle access to this function. If A makes queries to an oracle H , we denote this as A^H . If a system \mathcal{S} uses a random oracle in its specification, we denote this as \mathcal{S}^H . The algorithms comprising any cryptographic protocol can use H , as can the adversary. Thus we modify the security games for all cryptographic systems to allow the adversary to make random oracle queries.

When a random oracle scheme is implemented, some suitable hash function H is included in the specification. Any algorithm (adversary included) now replaces oracle queries with evaluations of this hash function. In the quantum setting, because a quantum algorithm can evaluate H on an arbitrary superposition, we must allow the quantum adversary to make quantum queries to the random oracle. We call this the quantum random oracle model.

3 Distinguishing Oracles with Quantum Queries

In this section, we give some tools for arguing that quantum algorithms cannot distinguish between two distributions of oracles. The trivial way to bound the ability of any quantum algorithm to distinguish two oracle distributions D_1 and D_2 is to bound the distance between D_1 and D_2 themselves. However, in many cases, the distributions we are interested in are in fact very far apart, so we need some new techniques for arguing indistinguishability. First, we show that k -wise equivalent distributions are indistinguishable, using techniques similar to that of Meyer and Pommersheim [MP11]:

Theorem 3.1. *Let A be a quantum algorithm making q quantum queries to an oracle $H : \mathcal{X} \rightarrow \mathcal{Y}$. If we draw H from some weight assignment D , then for every z , the quantity $\Pr_{H \leftarrow D}[A^H() = z]$ is a linear combination of the quantities $\Pr_{H \leftarrow D}[H(x_i) = r_i \forall i \in \{1, \dots, 2q\}]$ for all possible settings of the x_i and r_i .*

This is proved in the full version [Zha12a], and immediately implies two important facts:

- If two weight assignments on oracles, D_1 and D_2 , are $2q$ -wise equivalent, then any q query quantum algorithm behaves the same under both weight assignments, since for all $2q$ pairs (x_i, r_i) , $\Pr_{H \leftarrow D_1}[H(x_i) = r_i \forall i \in \{1, \dots, 2q\}] = \Pr_{H \leftarrow D_2}[H(x_i) = r_i \forall i \in \{1, \dots, 2q\}]$.
- If the quantities $\Pr_{H \leftarrow D}[H(x_i) = r_i \forall i \in \{1, \dots, 2q\}]$ are low-degree polynomials in some parameter λ , then so is $\Pr_{H \leftarrow D}[A^H() = z]$, since it is a linear combination of the $\Pr_{H \leftarrow D}[H(x_i) = r_i \forall i \in \{1, \dots, 2q\}]$. Moreover, $\Pr_{H \leftarrow D}[A^H() \in S]$ for any set S is also a low-degree polynomial.

We use the second fact to prove the following theorem:

Theorem 3.2. *Fix q , and let D_λ be a family of distributions on $\mathcal{H}_{\mathcal{X}, \mathcal{Y}}$ indexed by $\lambda \in [0, 1]$. Suppose there are integers d and Δ such that for every $2q$ pairs $(x_i, r_i) \in \mathcal{X} \times \mathcal{Y}$, the function $p(\lambda) = \Pr_{H \leftarrow D_\lambda}[H(x_i) = r_i \forall i \in \{1, \dots, 2q\}]$ satisfies:*

- p is a polynomial in λ of degree at most d .
- $p^{(i)}(0)$, the i th derivative of p at 0, is 0 for each $i \in \{1, \dots, \Delta - 1\}$.

Then any quantum algorithm A making q quantum queries can only distinguish D_λ from D_0 with probability at most $\frac{4^\Delta}{(2\Delta)!} \lambda^\Delta d^{2\Delta}$.

Proof. By the assumptions of the theorem, for any $2q$ pairs (x_i, r_i) , the quantity $\Pr_{H \leftarrow D_\lambda}[H(x_i) = r_i \forall i \in \{1, \dots, 2q\}]$ is a polynomial of degree d in λ with the first $\Delta - 1$ derivatives at 0 being 0. By Theorem 3.1, for a q -query quantum algorithm A , $\Pr_{H \leftarrow D_\lambda}[A^H() = z]$ is a linear combination of these values. Thus, for any z , $\Pr_{H \leftarrow D_\lambda}[A^H() = z]$ is also a polynomial in λ of degree d with the first $\Delta - 1$ derivatives at 0 being 0.

Now, suppose that A distinguishes D_λ from D_0 with probability $\epsilon(\lambda)$. That is

$$\sum_z \left| \Pr_{H \leftarrow D_\lambda}[A^H() = z] - \Pr_{H \leftarrow D_0}[A^H() = z] \right| = \epsilon(\lambda) .$$

Let \mathcal{Z}_λ be the set of z such that z is a more likely output under D_λ than D_0 . That is, $\Pr_{H \leftarrow D_\lambda}[A^H() = z] > \Pr_{H \leftarrow D_0}[A^H() = z]$. It is not difficult to show that

$$\Pr_{H \leftarrow D_\lambda}[A^H() \in \mathcal{Z}_\lambda] - \Pr_{H \leftarrow D_0}[A^H() \in \mathcal{Z}_\lambda] = \epsilon(\lambda)/2 .$$

Fix λ_0 , and consider the quantity

$$p_{\lambda_0}(\lambda) \equiv \Pr_{H \leftarrow D_\lambda}[A^H() \in \mathcal{Z}_{\lambda_0}] = \sum_{z \in \mathcal{Z}_{\lambda_0}} \Pr_{H \leftarrow D_\lambda}[A^H() = z] .$$

Then $p_\lambda(\lambda) - p_\lambda(0) = \epsilon(\lambda)/2$. Further, for each λ_0 , p_{λ_0} is a degree- d polynomial in λ such that $p_{\lambda_0}^{(i)}(0) = 0$ for $i \in \{1, \dots, \Delta - 1\}$. It also lies in the range $[0, 1]$ for all $\lambda \in [0, 1]$, so we can use an inequality by the Markov brothers [DS41] to bound the Δ -th derivative for all $\lambda \in [0, 1]$:

$$\left| p_{\lambda_0}^{(\Delta)}(\lambda) \right| \leq \frac{4^\Delta \Delta!}{2(2\Delta)!} d^{2\Delta}$$

Thus we can bound $p_{\lambda_0}(\lambda) \leq p_{\lambda_0}(0) + \frac{4^\Delta}{2(2\Delta)!} \lambda^\Delta d^{2\Delta}$. Setting $\lambda_0 = \lambda$, we get

$$\epsilon(\lambda) = 2(p_\lambda(\lambda) - p_\lambda(0)) \leq \frac{4^\Delta}{(2\Delta)!} \lambda^\Delta d^{2\Delta} .$$

□

We conclude this section with another general approach toward arguing that no quantum algorithm can distinguish between two oracle distributions D_1 and D_2 : construct a weight assignment $D_{1,2}$ which is $2q$ -wise equivalent to D_1 (so that the behaviors of A under D_1 and $D_{1,2}$ are the same) and that is a distance at most ϵ from D_2 (so that the behaviors of A under $D_{1,2}$ and D_2 are ϵ -close). Then the behaviors of A under D_1 and D_2 are also ϵ -close. We formalize this latter idea with a new pseudometric on oracle distributions:

Definition 3.3. Let D_1 and D_2 be two weight assignments on oracles. Fix k . Then define the pseudometric $|D_1 - D_2|_{(k)}$ as the minimum of $|D_{1,2} - D_2|$ over all weight assignments $D_{1,2}$ that are k -wise equivalent to D_1 .

Theorem 3.4. The minimum $D_{1,2}$ in Definition 3.3 exists, and $|D_1 - D_2|_{(k)}$ defines a pseudometric on weight assignments on oracles.

Proof. If D is k -wise equivalent to D_1 and a distance ϵ from D_2 , we say that D is a witness to $|D_1 - D_2|_{(k)} \leq \epsilon$. If $\epsilon = |D_1 - D_2|_{(k)}$, we call D an optimal witness.

First, we argue that an optimal witness exists: We see that D_1 itself is a witness to $|D_1 - D_2|_{(k)} \leq |D_1 - D_2|$. Thus to find $D_{1,2}$, we are minimizing $|D_{1,2} - D_2|$ (a continuous function in $D_{1,2}$) over the set of assignments $D_{1,2}$ that are k -wise similar to D_1 (a closed set) and at most $|D_1 - D_2|$ away from D_2 (a compact set). Thus we are minimizing over a compact set, so the minimum is actually obtained.

We now show that $|D_1 - D_2|_{(k)}$ satisfies the properties of a pseudometric:

- $|D_1 - D_1|_{(k)} = 0$, since D_1 is a witness to $|D_1 - D_1|_{(k)} \leq 0$, and $|D_1 - D_2|_{(k)}$ is non-negative for all weight assignments D_1 and D_2 .
- $|D_1 - D_2|_{(k)} = |D_2 - D_1|_{(k)}$. Let $D_{1,2}$ be an optimal witness for $|D_1 - D_2|_{(k)}$. Then define $D_{2,1} = D_1 - D_{1,2} + D_2$. The sums of all weights add to 1, so this is a weight assignment. Further, it is easy to see that $|D_{2,1} - D_1| = |D_2 - D_{1,2}| = |D_1 - D_2|_{(k)}$. Further, since D_1 is k -wise equivalent to $D_{1,2}$, when looking at the marginal weight assignment over any k inputs, D_1 and $D_{1,2}$ cancel, leaving $D_{2,1}$ to be k -wise equivalent to D_2 . Thus, $D_{2,1}$ is a witness to $|D_2 - D_1|_{(k)} \leq |D_1 - D_2|_{(k)}$. We can perform the same argument in reverse to get the other inequality.
- $|D_1 - D_3|_{(k)} \leq |D_1 - D_2|_{(k)} + |D_2 - D_3|_{(k)}$. Let $D_{1,2}$ and $D_{2,3}$ be the optimal witnesses for $|D_1 - D_2|_{(k)}$ and $|D_2 - D_3|_{(k)}$. Define $D_{1,3} = D_{1,2} - D_2 + D_{2,3}$. It is easy to see that D_3 is a weight assignment and that $|D_{1,3} - D_3| \leq |D_{1,2} - D_2| + |D_{2,3} - D_3|$. It is also not difficult to show that, since $D_{2,3}$ is k -wise equivalent to D_2 , $D_{1,3}$ must be k -wise equivalent to $D_{1,2}$, which is k -wise equivalent to D_1 . Hence $D_{1,3}$ is a witness for the identity in question. \square

Observe that $|D_1 - D_2|_{(k)} = 0$ if and only if D_1 is k -wise equivalent to D_2 . Thus, we can rephrase Theorem 3.1 as follows: if $|D_1 - D_2|_{(2q)} = 0$, then no quantum algorithm making q quantum queries can distinguish the oracle weight assignments D_1 and D_2 . We now generalize this to $|D_1 - D_2|_{(2q)} \neq 0$:

Corollary 3.5. Let D_1 and D_2 be two oracle distributions. Then the output weight assignment of any q -query quantum algorithm A under D_1 and D_2 are $|D_1 - D_2|_{(2q)}$ -close.

Proof. Let $D_{1,2}$ be an optimal witness for $|D_1 - D_2|_{(2q)}$. $D_{1,2}$ is $2q$ -wise equivalent to D_1 , so by Theorem 3.1, the behaviors of the A under D_1 and $D_{1,2}$ are identical. The behavior of A under D_2 is $|D_{1,2} - D_2| = |D_1 - D_2|_{(2q)}$ -close to that under $D_{1,2}$, and hence D_1 . \square

We note that a slightly weaker version of Theorem 3.2 can also be proved using the above ideas. Roughly, we pick some constants λ_i and find functions $a_i(\lambda)$ such that

$$p(\lambda) = \left(1 - \sum_{i=0}^{d-\Delta} a_i(\lambda) \right) p(0) + \sum_{i=0}^{d-\Delta} a_i(\lambda) p(\lambda_i)$$

for any polynomial p of degree d such that $p^{(i)}(0) = 0$ for $i \in \{1, \dots, \Delta - 1\}$. The $a_i(\lambda)$ are basically modified Lagrange interpolating polynomials. Next, we define a new weight assignment

$$D'_\lambda = \left(1 - \sum_{i=0}^{d-\Delta} a_i(\lambda) \right) D_0 + \sum_{i=0}^{d-\Delta} a_i(\lambda) D_{\lambda_i} .$$

By defining D'_λ in this way, D'_λ turns out to be $2q$ -wise equivalent to D_λ . Because D_{λ_i} are all distributions, the distance between D'_λ and D_0 is at most $2 \sum_{i=0}^{d-\Delta} |a_i(\lambda)|$, thus showing that $|D_\lambda - D_0|_{(2q)}$ also satisfies this bound. By picking the right values for λ_i , we can get this bound to be small. We can then use Corollary 3.5 to bound the distance between the output distributions of the q -query quantum algorithm A .

4 Semi-constant Distributions

In this section, we define a class of distributions on oracles in $\mathcal{H}_{\mathcal{X}, \mathcal{Y}}$ called semi-constant distributions. Our motivation for these distributions is to support quantum Full Domain Hash-type arguments, where a random value is inserted into a small but significant fraction of oracle inputs. The following definition captures the essence of the this idea:

Definition 4.1. Define SC_λ , the semi-constant distribution, as the distribution over $\mathcal{H}_{\mathcal{X}, \mathcal{Y}}$ resulting from the following process:

- First, pick a random element y from \mathcal{Y} .
- For each $x \in \mathcal{X}$, do one of the following:
 - With probability λ , set $H(x) = y$. We call x a distinguished input to H .
 - Otherwise, set $H(x)$ to be a random element in \mathcal{Y} .

First, notice that SC_0 is the uniform distribution, since in this case, the probability that an input is distinguished is 0. We bound the ability of a quantum algorithm to distinguish between SC_λ and $\text{SC}_0 = U$ using the ideas of Section 3.

Lemma 4.2. Fix k . For each k pairs (x_i, r_i) , the probability $\Pr_{H \leftarrow \text{SC}_\lambda}[H(x_i) = r_i \forall i \in \{1, \dots, k\}]$ is a degree- k polynomial in λ whose first derivative is 0 at $\lambda = 0$.

This, proved in the full version [Zha12a], shows that for any q_H , we can set $d = k = 2q_H$ and $\Delta = 2$ in the assumptions of Theorem 3.2. We immediately get:

Corollary 4.3. *The distribution of outputs of a quantum algorithm making q_H queries to an oracle drawn from SC_λ is at most a distance $\frac{8}{3}q_H^4\lambda^2$ away from the case when the oracle is drawn from the uniform distribution.*

We note that using standard quantum query results, it is possible to prove this corollary for a bound that is first-order in λ . However, we will see in Section 5 that we need a bound that is second-order in λ for our security results.

We do not know if this bound is tight. In the full version [Zha12a], we adapt the collision search algorithm of Brassard et al. [BHT97] to SC_λ , obtaining a distinguishing probability of $\Omega(q_H^3\lambda^2)$.

5 Quantum Security Arguments

In this section, we explore the random oracle proof technique of Bellare and Rogaway [BR93]. In this technique, we plug a challenge c into a randomly chosen hash query, and hope that the adversary uses c to break the system. If so, we can use the adversary to solve the underlying problem.

In the quantum setting, the straight-forward application of this technique breaks down. The adversary can now query the hash function on a superposition of exponentially many inputs. If we plug c to one of those inputs, the probability that the adversary uses c is exponentially small.

We instead plug c into some small but significant fraction of the inputs, so that the oracle is distributed according to SC_λ . As we have shown, a quantum algorithm cannot detect that this oracle is not random, but now the adversary uses c with significant probability.

5.1 Identity-Based Encryption from Lattices

Here we prove the security of the IBE scheme from Gentry et al. [GPV08]. Their scheme is constructed from an encryption scheme $E = (E.\text{Gen}, E.\text{Enc}, E.\text{Dec})$, for which there exists a trapdoor allowing the computation of secret keys from public keys.

More specifically, let $F = (F.\text{Gen}, F.\text{Sample}, f, f^{-1})$ be a pre-image sampleable function (PSF). Suppose $E.\text{Gen}(1^n)$ works as follows: generate $(\text{msk}, \text{mpk}) \leftarrow F.\text{Gen}(1^n)$. Then, sample $\text{sk} \leftarrow F.\text{Sample}(1^n)$, and compute $\text{pk} = f_{\text{mpk}}(\text{sk})$. Output $(\text{sk}, (\text{pk}, \text{mpk}))$.

Gentry et al. give such an encryption scheme based on the hardness of lattice problems. Their security reduction treats the adversary as a black box, so their proof holds in the quantum setting. They then prove the security of the IBE scheme $\text{IBE} = (\text{IBE}.\text{Gen} = F.\text{Gen}, \text{IBE}.\text{Extract}^H, \text{IBE}.\text{Enc}^H, \text{IBE}.\text{Dec})$ where H maps identities to public keys of E and:

- $\text{IBE}.\text{Extract}_{\text{msk}}(\text{id})^H = f_{\text{msk}}^{-1}(H(\text{id}))$
- $\text{IBE}.\text{Enc}_{\text{mpk}}(\text{id}, m)^H = E.\text{Enc}_{H(\text{id})}(m)$
- $\text{IBE}.\text{Dec}_{\text{sk}_{\text{id}}}(c) = E.\text{Dec}_{\text{sk}_{\text{id}}}(c)$

Theorem 5.1. Let E and F be as above, and suppose that E is quantum IND-CPA-secure. If we model H as a random oracle, then IBE is quantum IND-ID-CPA-secure.

Proof. Let A be a quantum adversary making q_H hash queries, q_E extract queries queries, that breaks IBE with advantage ϵ .

Let \mathbf{Game}_0 be the standard attack game for IBE : the challenger generates $(\mathsf{msk}, \mathsf{mpk})$ from $\mathsf{IBE}.\mathsf{Gen}$, and sends mpk to the adversary. The adversary can make (classical) extraction queries on identities id_i , to which the challenger responds with $\mathsf{IBE}.\mathsf{Extract}_{\mathsf{msk}}^H(\mathsf{id}_i)$, and (quantum) hash queries to the random oracle H . A then produces an identity id^* , along with messages m_0 and m_1 . The challenger chooses a random bit b , and responds with $\mathsf{IBE}.\mathsf{Enc}_{\mathsf{mpk}}(\mathsf{id}, m_b)$. A is allowed to make more extraction and hash queries, and produces a bit b' . A wins if $b' = b$ and for all i , $\mathsf{id}_i \neq \mathsf{id}^*$. By definition, this happens with probability $\frac{1}{2} + \epsilon$.

Assume without loss of generality that A never asks for the secret key corresponding to id^* . Let A' be the following algorithm that makes quantum queries to an oracle H , and simulates the interaction between A and the challenger: generate $(\mathsf{msk}, \mathsf{mpk})$ from $\mathsf{IBE}.\mathsf{Gen}$, send mpk to A . Run A , forwarding all hash queries to H , and answering extraction queries using $\mathsf{IBE}.\mathsf{Extract}^H$. Similarly, answer the challenge query by choosing a random b and encrypting m_b to the identity id generated by A . The output of A' is (c, id^*, Q) where $c = b \oplus b'$ and Q is the list of extraction queries made by A . We can now think of \mathbf{Game}_0 as follows: run A' with a random oracle H to obtain (c, id^*, Q) . Report that the game is won if and only if $c = 0$. The number of queries to H made by A' is q_H for direct queries, q_E for queries through the extract algorithm, and 1 for the encryption of m_b , for a total of $q_H + q_E + 1$ queries.

Let $\lambda \in (0, 1)$ to be chosen later. Let \mathcal{M} be the identity space. Define \mathbf{Game}_1 as follows: first, pick a subset \mathcal{X} of \mathcal{M} where each $\mathsf{id} \in \mathcal{M}$ is put in \mathcal{X} with independent probability λ . Run A' as before to obtain (c, id^*, Q) . However, now we decide if the game is won as follows: if $\mathsf{id}^* \in \mathcal{X}$ and $Q \cap \mathcal{X} = \emptyset$, then the game is won if $c = 0$. Otherwise, flip a random coin to decide if the game is won. By our assumption that A never asks for a secret key for the challenge identity, we must have that $\mathsf{id}^* \notin Q$. Therefore, every identity in Q , as well as id^* , has an independent probability λ of being in \mathcal{X} . Thus, the conditions to not flip a coin are met with probability at least $\lambda(1 - \lambda q_E)$, independent of the probability that $c = 0$. Thus, \mathbf{Game}_1 is won with probability at least $\frac{1}{2} + \lambda(1 - \lambda q_E)\epsilon \geq \frac{1}{2} + \lambda\epsilon - q_E\lambda^2$.

Let \mathbf{Game}_2 be \mathbf{Game}_1 , except that we choose a random pk in the public key space of E , and set $H(x) = \mathsf{pk}$ for all x in \mathcal{X} , and choose $H(x)$ randomly for all other x . H is now distributed according to SC_λ .

Claim. There exists a polynomial $\ell(\cdot, \cdot)$ such that A' wins \mathbf{Game}_2 with probability at least

$$\frac{1}{2} + \lambda\epsilon - \frac{\ell(q_H, q_E)}{4}\lambda^2 .$$

We defer the precise proof to the full version [Zha12a], and instead give an informal justification: by Corollary 4.3, the output distribution of A' in Game_2 is at most a distance $\frac{8}{3}(q_H + q_E + 1)^4\lambda^2$ from that of Game_1 . Thus, we would expect A' to win Game_2 with probability at least $\frac{1}{2} + \lambda\epsilon - (q_E + \frac{8}{3}(q_H + q_E + 1)^4)\lambda^2$, which has the desired form. Of course, deciding if A' wins Game_2 depends on the set of distinguished inputs \mathcal{X} , so a more careful analysis is required to justify the claim.

Let us now restate Game_2 in terms of A : pick \mathcal{X} and H as above, and perform the standard attack game for IBE using H as the random oracle. When A produces b' , first check that no extraction queries were in \mathcal{X} and that the identity id^* produced by A is in \mathcal{X} . If so, A wins if and only if $b = b'$. Otherwise, if the check fails, flip a coin to decide if A wins. We can now make the following observation: once an extraction query in \mathcal{X} is made or the challenge query with $\text{id}^* \notin \mathcal{X}$ is made, we can abort running A , since we no longer need A to determine if the game is won.

Now we describe an adversary B that breaks E . Assume B has quantum access to two random oracles O_1 and O_2 . O_1 maps identities to randomness used by Sample . O_2 maps identities to bits, outputting 1 with probability λ . In Section 6, we will show how to efficiently simulate these oracles. On input (mpk, pk) , B works as follows:

- Send mpk to A and simulate A , playing the role of challenger to A .
- Construct a (quantum) oracle H such that

$$H(\text{id}) = \begin{cases} \text{pk} & \text{if } O_2(\text{id}) = 1 \\ f_{\text{mpk}}(\text{Sample}(\cdot; O_1(\text{id}))) & \text{otherwise} \end{cases}$$

where $\text{Sample}(\cdot; r)$ means run Sample with randomness r .

- When A asks for the secret key for id_i , compute $O_2(\text{id}_i)$. If the result is 1, output a random bit and abort. Otherwise, respond with $\text{sk}_i = \text{Sample}(\cdot; O_1(\text{id}_i))$.
- When A produces the challenge query (id^*, m_0, m_1) , check if $O_2(\text{id}^*) = 1$. If so, send (m_0, m_1) to B 's challenger. Otherwise, output a random bit and abort.
- When the challenger responds with a ciphertext, send it to A .
- When A outputs a bit b' , output the same bit.

Let \mathcal{X} be the set of identities id such that $O_2(\text{id}) = 1$. We can then see that the abort conditions are equivalent to Game_2 . For each extract query id_i that succeeds, we have that $O_2(\text{id}_i) = 0$, so

$$f_{\text{mpk}}(\text{sk}_i) = f_{\text{mpk}}(\text{Sample}(\cdot; O_1(\text{id}_i))) = H(\text{id}_i)$$

Thus sk_i is a correct secret key for id_i , and it is distributed correctly (since it is a random pre-image of $H(\text{id}_i)$). If B does not abort on the challenge query, it means $O_2(\text{id}^*) = 1$, so $H(\text{id}^*) = \text{pk}$. Also, we can see that H is distributed according to SC_λ . Therefore, if B aborts, we win with probability $\frac{1}{2}$, and if we do not abort, we win if and only if A wins. Thus, the view of A as a subroutine

of B is the same as in Game₂, and B wins with the same probability that A does. Therefore, B wins with probability at least $\frac{1}{2} + \lambda\epsilon - \frac{\ell(q_H, q_E)}{4}\lambda^2$.

The advantage of B is at least $\lambda\epsilon - \ell(q_H, q_E)\lambda^2/4$. We can now choose λ : the maximum advantage occurs when $\lambda = 2\epsilon/\ell(q_H, q_E)$, which gives

$$Adv_B \geq \frac{\epsilon^2}{\ell(q_H, q_E)}$$

Thus, if A had non-negligible advantage, so does B . \square

5.2 Hierarchical IBE from Lattices

In this section, we show the general idea behind adapting the techniques above to proving the security of the hierarchical identity-based encryption (HIBE) schemes of Agrawal et al. [ABB10] and Cash et al. [CHKP10].

In a HIBE scheme, identities are structured as a tree, with the identity of any node containing the identity of its parent as a proper prefix. Any node on the tree can produce private keys for any nodes in the subtree rooted at that node. We allow an adversary to adaptively take control of an arbitrary number of nodes in the tree (and thus the subtrees rooted there). An HIBE scheme is secure if the adversary cannot decrypt messages encrypted to an identity id^* of the adversary's choice but not under its control.

In [ABB10], the random oracle scheme has an oracle H that maps identities to some random quantities. The reduction has the following high-level structure:

- Guess which level w of the tree contains the identity id^*
- For each level i , generate some random quantities R_i .
- For each level i , simulate a separate random oracle for identities at that level. For i , guess which query number q_i will contain the hash of the level- i parent of id^* .
- Answer the j th random oracle query at level i on $\text{id}_{i,j}$ as follows: if $j = q_i$, output R_i . Otherwise, output a random value.
- Answer secret key queries on id in some special way, but fail if for all prefixes id_i of id , $\text{id}_i = \text{id}_{i,q_i}$. That is, fail if $H(\text{id}_i) = R_i$.
- When the adversary generates the identity id^* , we succeed if both the adversary succeeds and if id^* is at level w and all prefixes id_i^* of id^* satisfy $\text{id}_i^* = \text{id}_{i,q_i}$.

We now show how to prove security by repeatedly applying the arguments of Theorem 5.1. Basically, we iterate over levels i , and insert R_i into a λ_i fraction of identities at that level. In iteration i , we say the adversary wins if it won in the previous iteration, the level- i prefix of the chosen identity id^* is in the λ_i fraction of distinguished identities (that is, $H(\text{id}_i^*) = R_i$), and no signature query is. Let $\ell = \ell(q_H, q_E)$, where $\ell(\cdot, \cdot)$ is the polynomial from Theorem 5.1. If the iteration i advantage is ϵ_i , then using the same techniques as in Theorem 5.1, we can set λ_i so that

$$\epsilon_i \geq \frac{\epsilon_{i-1}^2}{\ell}$$

We will say that in iteration 0, the adversary wins if it normally would win and we guessed which level id^* belonged to correctly. That is, $\epsilon_0 = \epsilon/d$, where ϵ is the adversary's advantage in the standard game. This gives us a total advantage after iteration d of at least

$$\frac{(\epsilon/d)^{2^d}}{\ell^{(2^d-1)}} = \ell \left(\frac{\epsilon}{d\ell} \right)^{2^d}$$

Notice that the dependence on d is doubly-exponential, whereas in the classical proof it was singly exponential. Thus, for the same security parameters, this proof only works for much smaller depth than the classical proof.

These techniques apply as well to the random oracle HIBE of Cash et al. [CHKP10], though their reduction is a bit more complicated, as there is a second random oracle G which needs to be handled in a similar way.

5.3 Signatures from Trapdoor Permutations

Here we discuss the security of the Full Domain Hash (FDH) signature scheme:

Definition 5.2 (FDH Signatures). Let $F = (F.\text{Gen}, f, f^{-1})$ be a trapdoor permutation, and a hash function H that maps messages to images of f , let $S^H = (S.\text{Gen} = F.\text{Gen}, S.\text{Sign}^H, S.\text{Ver}^H)$ where:

- $S.\text{Sign}_{\text{sk}}^H(m) = f_{\text{sk}}^{-1}(H(m))$
- $S.\text{Ver}_{\text{pk}}^H(m, \sigma) = \begin{cases} \text{accept} & \text{if } f_{\text{pk}}(\sigma) = H(m) \\ \text{reject} & \text{otherwise} \end{cases}$

We now state the main theorem of this section:

Theorem 5.3. Let F be a quantum one-way trapdoor permutation. If we model H as a random oracle, then S is quantum UF-CMA-secure.

The proof of this theorem is very similar to that of Theorem 5.1, and appears in the full version [Zha12a].

6 Simulating Random Oracles

In this last section, we explain how to efficiently simulate random oracles. All quantum random oracle proofs to date require the reduction algorithm to have quantum access to a random oracle, but a truly random oracle requires exponentially many bits of randomness to construct. We show that this is not a problem:

Theorem 6.1. Any quantum algorithm A making quantum queries to random oracles O_i can be efficiently simulated by a quantum algorithm B , which has the same output distribution, but makes no queries.

Proof. We construct an algorithm B which simulates A , and answers queries to oracle O_i with evaluations of efficient functions f_i . Boneh et al. [BDF⁺11] use pseudorandom functions (PRF) for the f_i . At first glance, this seems like the only option, as we need a function f_i that cannot be distinguished from random.

Notice, however, that f_i need not be secure against all adversaries, just the adversary we are simulating. We know that our adversary makes q_i queries to oracle O_i , so it suffices to have f_i be PRFs secure for up to q_i quantum queries. In the classical setting, q_i -wise independent functions (functions that are q_i -wise equivalent to a random function) serve as *perfectly* secure PRFs for up to q_i classical queries. We could hope that something similar holds in the quantum world: indeed, according to Theorem 3.1, if f_i is $2q_i$ -wise equivalent to a random function, then the behavior of our adversary is the same when the oracle is random and when it is f_i . Thus if the f_i are $2q_i$ -wise independent, algorithm A , as a subroutine of B , behaves identically to the case where A is given truly random oracles. Hence, the output distribution of B is identical to that of A .

Efficient constructions of k -wise independent functions have been known for some time [Jof74, KM94], and they have been used extensively in the derandomization literature [Lub85, ABI86, KM93]. One common approach to construct a k -wise independent function f from \mathcal{X} to \mathcal{Y} is to assume that $N = |\mathcal{Y}|$ is a prime power and interpret \mathcal{Y} as the field \mathbb{F}_N . Then define a matrix C with the following properties:

- The entries are elements in \mathbb{F}_N .
- There are $|\mathcal{X}|$ rows and some small number r of columns.
- Each subset of k rows is linearly independent.

One such example is the Vandermonde matrix, which is used by Alon et al. [ABI86]. To define the function f , we then pick a random vector v in \mathbb{F}_N^r . $f(x)$ is then the x th element of the vector Cv . Since any k rows of C are linearly independent, any k elements of Cv are independent, and hence f is k -wise independent. The key to making this efficient is that to compute $f(x)$, we only need the x th row of C , which we can compute on the fly. \square

Hence, we can simulate O_1 from Theorem 5.1. To simulate O_2 , which outputs a bit such that $O_2(x) = 1$ with probability λ , approximate λ by some rational number a/b where b is a prime power, and construct a k -wise independent function f' with range $\mathcal{Y} = \{1, \dots, b\}$. Then set

$$f(x) = \begin{cases} 1 & \text{if } f'(x) \leq a \\ 0 & \text{otherwise} \end{cases}$$

7 Conclusion

We have shown how to adapt certain classical random oracle arguments to the quantum random oracle model. Specifically, we gave quantum security proofs for the IBE scheme of Gentry et al. [GPV08] and the Full Domain Hash signature scheme. We achieved this by defining a distribution of oracles, called

semi-constant distributions, and showing that such oracles cannot be distinguished from a random oracle by a quantum adversary. We also show how these techniques can be applied to the random oracle HIBE schemes of Cash et al. [CHKP10] and Agrawal et al. [ABB10]. Lastly, we have shown how to remove the need for quantum-secure pseudorandom functions from prior work.

Although we have made progress toward converting classical random oracle proofs into quantum proofs, there is still much work to be done. First, reductions using our technique result in an algorithm whose advantage is the square of the advantage of the underlying adversary, divided by some other factors. While improving the reduction to first-order in the adversary's advantage would still be of interest for the IBE scheme and Full Domain Hash, it would not affect the qualitative security statements. However, for the HIBE schemes, this second-order behavior results in an exponential weakening of the security relative to the classical reduction for deep identity trees. Therefore, to get the same qualitative security statements as in the classical world, our technique needs to be refined to make the reduction first-order in ϵ .

Further work lies in proving the security of other Random Oracle Model schemes. For example, the construction of signatures from identification protocols by Fiat and Shamir [FS87], though similar to the proofs in this paper, still needs a quantum proof. The difficulty in the Fiat-Shamir reduction is that the plugging step initiates the underlying identification protocol, and there is no obvious quantum analog for this strategy. Also, different types of security arguments, such as Fujisaki and Okamoto's generic conversion of weakly secure encryption schemes into a CCA-secure encryption scheme [FO99], still lack a quantum proof of security.

Acknowledgments. We would like to thank Dan Boneh for his guidance and many insightful discussions. This work was supported by NSF and DARPA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

References

- [Aar09] Aaronson, S.: Quantum Copy-Protection and Quantum Money. In: Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC (2009)
- [ABB10] Agrawal, S., Boneh, D., Boyen, X.: Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010)
- [ABI86] Alon, N., Babai, L., Itai, A.: A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. Journal of Algorithms 7(4), 567–583 (1986)
- [BBBV97] Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.: Strengths and Weaknesses of Quantum Computing. SIAM Journal on Computing 26, 1510–1523 (1997)

- [BDF⁺11] Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random Oracles in a Quantum World. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (2011)
- [BF01] Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
- [BF11] Boneh, D., Freeman, D.M.: Homomorphic Signatures for Polynomial Functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
- [BHK⁺11] Brassard, G., Høyer, P., Kalach, K., Kaplan, M., Laplante, S., Salvail, L.: Merkle Puzzles in a Quantum World. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 391–410. Springer, Heidelberg (2011)
- [BHT97] Brassard, G., Høyer, P., Tapp, A.: Quantum Algorithm for the Collision Problem. ACM SIGACT News (Cryptology Column) 28, 14–19 (1997)
- [BR93] Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS), pp. 62–73. ACM (November 1993)
- [BS08] Brassard, G., Salvail, L.: Quantum Merkle Puzzles. In: Second International Conference on Quantum, Nano and Micro Technologies (ICQNM 2008), pp. 76–79 (February 2008)
- [CHKP10] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
- [DS41] Duffin, R.J., Schaffer, A.C.: A Refinement of an Inequality of the Brothers Markoff. Trans. Amer. Math. Soc. 44(3), 289–297 (1941)
- [FO99] Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
- [FS87] Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
- [GKV10] Dov Gordon, S., Katz, J., Vaikuntanathan, V.: A Group Signature Scheme from Lattice Assumptions. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 395–412. Springer, Heidelberg (2010)
- [GM84] Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and System Sciences, 270–299 (1984)
- [GMR88] Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
- [GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), p. 197 (2008)
- [HSS11] Hallgren, S., Smith, A., Song, F.: Classical Cryptographic Protocols in a Quantum World. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 411–428. Springer, Heidelberg (2011)
- [Jof74] Joffe, A.: On a Set of Almost Deterministic k-Independent Random Variables. The Annals of Probability 2(1), 161–162 (1974)

- [KM93] Koller, D., Megiddo, N.: Constructing Small Sample Spaces Satisfying Given Constraints. In: Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC), pp. 268–277. ACM (1993)
- [KM94] Karloff, H., Mansour, Y.: On Construction of k -Wise Independent Random Variables. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC), vol. 17, pp. 564–573 (1994)
- [Lub85] Luby, M.: A Simple Parallel Algorithm for the Maximal Independent Set Problem. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC), pp. 1–10. ACM (1985)
- [MP11] Meyer, D., Pommersheim, J.: On the Uselessness of Quantum Queries. *Theoretical Computer Science*, 1–12 (March 2011)
- [NC00] Nielsen, M.A., Chuang, I.: Quantum Computation and Quantum Information. *American Journal of Physics* 70(5), 558 (2000)
- [Sho97] Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing* 26(5), 1484–1509 (1997)
- [Unr10] Unruh, D.: Universally Composable Quantum Multi-Party Computation. In: Gilbert, H. (ed.) *EUROCRYPT 2010*. LNCS, vol. 6110, pp. 486–505. Springer, Heidelberg (2010)
- [Zha12a] Zhandry, M.: Secure Identity-Based Encryption in the Quantum Random Oracle Model (February 2012); Full version available at the Cryptology ePrint Archives, <http://eprint.iacr.org/2012/076/>
- [Zha12b] Zhandry, M.: How to Construct Quantum Random Functions (April 2012); Full version available at the Cryptology ePrint Archives, <http://eprint.iacr.org/2012/182/>

Quantum to Classical Randomness Extractors*

Mario Berta¹, Omar Fawzi², and Stephanie Wehner³

¹ Institute for Theoretical Physics, ETH Zurich
bertha@phys.ethz.ch

² School of Computer Science, McGill University
ofawzi@cs.mcgill.ca

³ Centre for Quantum Technologies, National University of Singapore
wehner@nus.edu.sg

Abstract. The goal of randomness extraction is to distill (almost) perfect randomness from a weak source of randomness. When the source outputs a classical string X , many extractor constructions are known. Yet, when considering a physical randomness source, X is itself ultimately the result of a measurement on an underlying quantum system. When characterizing the power of a source to supply randomness it is hence a natural question to ask, how much *classical* randomness we can extract from a *quantum* system. To tackle this question we here take on the study of *quantum-to-classical randomness extractors* (QC-extractors).

We provide constructions of QC-extractors based on measurements in a full set of mutually unbiased bases (MUBs), and certain single qubit measurements. The latter are particularly appealing since they are not only easy to implement, but appear throughout quantum cryptography. We proceed to prove an upper bound on the maximum amount of randomness that we could hope to extract from any quantum state. Some of our QC-extractors almost match this bound. We show two applications of our results.

First, we show that any QC-extractor gives rise to entropic uncertainty relations with respect to quantum side information. Such relations were previously only known for two measurements. In particular, we obtain strong relations in terms of the von Neumann (Shannon) entropy as well as the min-entropy for measurements in (almost) unitary 2-designs, a full set of MUBs, and single qubit measurements in three MUBs each.

Second, we finally resolve the central open question in the noisy-storage model [Wehner et al., PRL 100, 220502 (2008)] by linking security to the quantum capacity of the adversary's storage device. More precisely, we show that any two-party cryptographic primitive can be implemented securely as long as the adversary's storage device has sufficiently low quantum capacity. Our protocol does not need any quantum storage to implement, and is technologically feasible using present-day technology.

Keywords: randomness extractors, randomness expansion, entropic uncertainty relations, mutually unbiased bases, quantum side information, two-party quantum cryptography, noisy-storage model.

* A full version with complete proofs can be found online arXiv:1111.2026.

1 Introduction

Randomness is an essential resource for information theory, cryptography, and computation. However, most sources of randomness exhibit only weak forms of unpredictability. The goal of randomness extraction is to convert such weak randomness into (almost) uniform random bits. Classically, a weakly random source simply outputs a string X where the ‘amount’ of randomness is measured in terms of the probability of guessing the value of X ahead of time. That is, it is measured in terms of the min-entropy $H_{\min}(X) = -\log P_{\text{guess}}(X)$. To convert X to perfect randomness, one applies a function Ext that takes X , together with a shorter string R of perfect randomness (the *seed*) to an output string $K = \text{Ext}(X, R)$. The use of a seed is thereby necessary to ensure that the extractor works for all sources X about which we know only the min-entropy, but no additional details of the source. Much work has been invested into showing that particular classes of functions have the property that K is indeed very close to uniform as long as the min-entropy of the source $H_{\min}(X)$ is large enough (see [37] for a survey).

Yet, for most applications this is not quite enough, and we want an even stronger statement. In particular, imagine that we hold some quantum system E containing *side information* about X that increases our guessing probability to $P_{\text{guess}}(X|E)$. For example, such side information could come from an earlier application of an extractor to the same source. Intuitively, one would not talk about randomness if e.g., the output is uniformly distributed, but identical to an earlier output. In a cryptographic setting, side information can also be gathered by an adversary during the course of the protocol. We thus ask that the output is perfectly random even with respect to such side information, i.e., uniform *and* independent of E . Classically, it is known that extractors are indeed robust against classical side information [23], yielding a uniform output K , whenever the min-entropy about X *given access to side information* E ($H_{\min}(X|E) = -\log P_{\text{guess}}(X|E)$) is sufficiently high. Especially with respect to cryptographic applications, we thereby again want extractors that work for any source X of sufficiently high entropy $H_{\min}(X|E)$ without any additional assumptions about the source.

Recently, it has been recognized that since the underlying world is not classical, E may in fact hold *quantum* side information about X [21,35]. That this adds substantial difficulty to the problem was emphasized in [17] where it was shown that there are in fact situations where using the same extractor gives a uniform output K if E is classical, but is entirely predictable when E is quantum. Positive results were obtained in [35,23,34,42], eventually culminating in [39,12], proving that a wide class of classical extractors (with relatively short seed) yield a uniform output, as long as $H_{\min}(X|E)$ is sufficiently large.

Yet, in a fully quantum world we might ask ourselves: where does X itself come from? How can we hope to harness weak sources to obtain a *surplus* of classical randomness? Indeed, for any physical source hoping to create fresh randomness, X is the result of a measurement on a quantum system A . That is, we can view the source as consisting of in fact two processes: First, a *quantum* source emits a

state ρ_A . Second, a measurement takes place yielding the classical string X . Note that quantum mechanics does allow many different measurements on ρ_A , and hence the question arises whether all such measurements are equally powerful at yielding a (weakly) random classical string X , or whether some are more useful to us than others. As such, it becomes clear that when trying to study our ability to extract randomness from any physical source, it is natural to ask how much randomness we can obtain from ρ_A itself, rather than a classical distribution X that might be the outcome of a particular measurement.

The problem of extracting randomness from X alone is further complicated by the fact that it is typically very hard to bound $H_{\min}(X|E)$, when X is the result of quantum measurements on A , *even* if we know stringent bounds on the *quantum* correlations between A and E to begin with. When E is trivial, entropic uncertainty relations [45] give such bounds when we are willing to average over a few randomly chosen measurements. A crude bound on $H_{\min}(X|E)$ can then be obtained by assuming that the size of E is limited. But even classically, it is easy to see that there exist scenarios where bounding the adversaries' knowledge simply by his memory size yields very weak bounds [24]. Another approach to bounding $H_{\min}(X|E)$, common in e.g., Quantum Key Distribution (QKD), is possible in the case when randomness is extracted from a state ρ_{ABE} where measurements are made on both A and B to obtain an estimate of $H_{\min}(X|E)$ where X is obtained from A alone [41,8,31,9]. Part of the state is thereby consumed during the estimation process, which itself requires randomness. It is nevertheless possible to have an overall gain in randomness. For example, it is known that if measurements¹ between systems A and B lead to a so-called Bell inequality violation, then E knows little about X [8,31,9]. Clearly, making such an estimate is only possible in a special setting where the states have a particular form ρ_{ABE} , and we are given access to B and A .

1.1 Quantum to Classical Extractors

This leads us to study *quantum-to-classical randomness extractors* (QC-extractors). Our goal is to answer the following question: how can we extract *classical* randomness from a physical source ρ_{AE} by performing measurements on the *quantum* state ρ_A ? In analogy to classical extractors, we thereby want to obtain randomness from the source given only a minimal guarantee about its randomness - i.e., like min-entropy $H_{\min}(X|E)$ for classical sources. It is important to note that unlike the classical world, quantum mechanics does allow for the creation of true randomness *if* we are given full control of the source and can prepare any state ρ_A at will.² However, we want our extractors to work for *any* unknown source as long as it has sufficiently high entropy.

¹ That satisfy the no-signalling condition.

² For example, we could prepare the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and measure it in the computational basis, yielding a truly random outcome. Yet, this would correspond to controlling and knowing details of the source.

As opposed to classical-to-classical extractors (CC-extractors) given by functions $\text{Ext}(\cdot, R)$ mapping the outcome of the randomness source to a string K , a QC-extractor is described by projective measurements whose outcomes correspond to a classical string K . That is, a QC-extractor is a set of measurements $\{\mathcal{M}_{A \rightarrow K}^1, \dots, \mathcal{M}_{A \rightarrow K}^L\}$, where the random seed R determines the measurement $\mathcal{M}_{A \rightarrow K}^R$ that we will perform (see Section 3 for a detailed explanation and a formal definition).³

When talking about quantum states ρ_{AE} , what is the relevant measure of how weak or strong a source is? To gain some intuition on what the relevant measure should be, consider the case where ρ_{AE} is the maximally entangled state between A and E . Intuitively, this is the strongest quantum correlation that can exist between two systems. It is not hard to see that if we measure A in *any* basis to obtain some outcome X , and later communicate the choice of basis to an adversary holding E , then the adversary can guess X perfectly. Intuitively, we would thus expect that the relevant measure of how weak a quantum source is with respect to E involves a measure of the amount of entanglement between A and E . It turns out that the conditional min-entropy $H_{\min}(A|E)$ is exactly such a measure [22], and we find that it is indeed the quantity that determines how many classical random bits we can hope to extract from A . That this is rather analogous to the classical case is very appealing. However, unlike for classical A , $H_{\min}(A|E)$ can be *negative* if A is quantum (see below).

Note that in a quantum setting, we could also consider a quantum-to-quantum extractor (QQ-extractor). That is, an extractor in which we do not measure but merely ask that the resulting state is quantumly fully random (i.e., maximally mixed) and uncorrelated from E . Clearly, any QQ-extractor also forms a QC-extractor since any subsequent measurement on the maximally mixed state has a uniform distribution over outcomes. As such a QQ-extractor is stronger than a QC-extractor since for the latter we only require the output state to be close to uniform *after* performing a measurement.⁴ Constructions for such extractors are indeed well known in quantum information theory as a consequence of a notion known as ‘decoupling’, which plays a central role in quantum information theory (see [18,13,14] and references therein). In general, a map that transforms a state ρ_{AE} into a state that is close to a product state $\sigma_A \otimes \rho_E$ is a decoupling map. Decoupling processes thereby typically take the form of choosing a random unitary from a set $\{U_1, \dots, U_L\}$, applying this unitary to the system $A = A_1 A_2$ and tracing out (i.e., ignoring) the system A_2 . For certain classes of unitaries such as (almost) unitary 2-designs [14,38] (see below) the resulting state $\rho_{A_1 E}$ is close to maximally mixed on A_1 and uncorrelated from E , whenever $H_{\min}(A|E)$

³ For quantum information theorists, note that one can of course performing successive measurements - however, recall that we are interested in how much randomness we can obtain from an unknown source using a single measurement. The latter is furthermore motivated by experimental situations where successive measurements are typically very hard to implement.

⁴ In quantum mechanics, it is possible to obtain a uniform distribution over outcomes even if the state was not maximally mixed. E.g., consider measuring the pure state $|0\rangle\langle 0|$ in the Fourier basis.

is sufficiently large. Measurements consisting of applying such a unitary, followed by a measurement on A_1 thus also yield QC-extractors.⁵

The authors of [2] also proposed a definition of quantum extractors that is indeed somewhat similar to a QQ-extractor, however without any side information E . Our definitions (see Section 3) impose two important requirements not present in [2, Definition 5.1]. Firstly, we require the output of the extractor to be unpredictable for any, possibly quantum, adversary with access to side information E provided $H_{\min}(A|E)$ is large enough. Secondly, we consider *strong* extractors so that even given the seed R , the output of the extractor cannot be predicted. This allows us to employ our extractor for cryptographic purposes. It also means that the output K *together with* R are jointly close to uniform, meaning that we have effectively created *more* almost perfect randomness than we invested in the seed.

QC-extractors. We give two novel constructions of QC-extractors.⁶ The first one involves a full set of mutually unbiased bases (MUBs) and pair-wise independent permutations (Theorem 3). This construction is more appealing than unitary 2-designs because it is combinatorially simpler to describe and computationally more efficient, while having the same output size.

Our second construction (Theorem 4) is composed of unitaries acting on single qudits followed by some measurements in the computational basis. We also refer to these as *bitwise* QC-extractors. An appealing feature of the measurements defined by these unitaries is that they can be implemented *with current technology*. In addition to computational efficiency, the fact that the unitaries act on a single qubit is often a desirable property for the design of cryptographic protocols in which the creation of randomness is not the only requirement for security. Our example application below (see also Section 5) illustrates this.

Finally, we also prove in Proposition 1 that the maximum amount of randomness one can hope to extract is roughly $n + H_{\min}(A|E)$, where n denotes the input size. This upper bound can indeed be almost achieved by means of, e.g., our full set of MUBs QC-extractor. We also establish basic upper and lower bounds on the seed size for QC-extractors (see Table 1).

The technique we use to prove that our constructions are QC-extractors is to bound the distance between the output of the extractor and the desired output in Hilbert-Schmidt norm (cf. [38]). For the full set of MUBs, this distance can even be computed *exactly*. We use the fact that the set of all the MUB vectors forms a complex projective 2-design and that the set of permutations is pair-wise independent. For our second construction, the analysis uses similar ideas

⁵ For decoupling experts, note that the measurement map in a QC-extractor can be understood as a decoupling map. We would like to emphasize though that our QC-extractor results do not follow from previous work on decoupling, and our measurements have many nice properties not shared by unitaries used previously for decoupling.

⁶ That is, not following from results on QQ-extractors (i.e., from general decoupling theorems in quantum information theory).

in a more involved calculation. Our upper bound on the amount of extractable randomness follows from simple monotonicity properties of the min-entropy. The upper bound on the seed size follows from a non-explicit construction involving concentration-of-measure techniques.

1.2 Application to Entropic Uncertainty Relations

One of the fundamental ideas in quantum mechanics is the uncertainty principle. The security of essentially all quantum cryptographic protocols is founded on its existence. Intuitively, it states that even with complete knowledge about the quantum state ρ_A of a system A , it is impossible to predict the outcomes of all possible measurements on A with certainty. In an information theoretic context it is very natural to quantify this lack of knowledge in terms of entropic uncertainty relations (see [45] for a survey). Apart from their deep significance in the foundations of quantum mechanics, entropic uncertainty relations are crucial tools in quantum information theory and quantum cryptography. The most well-known relation is for two measurements $\mathcal{M}_{A \rightarrow K}^1, \mathcal{M}_{A \rightarrow K}^2$ and reads [27]

$$\frac{1}{2} \sum_{j=1}^2 H(K)_{\rho^j} \geq \log \frac{1}{c}, \quad (1)$$

where $H(K)_{\rho^j}$ denotes the Shannon entropy of the post-measurement probability distributions $\rho_K^j = \mathcal{M}_{A \rightarrow K}^j(\rho_A)$, and c measures the overlap between the measurements. Note that for any quantum state ρ_A and measurements for which $c \neq 1$, at least one of the entropies has to be greater than zero. In other words, it is impossible to predict the outcomes of both measurements with certainty. Uncertainty relations are thereby called *strong*, if $\log(1/c)$ is large.

Just as extractors can depend on side information E , it is important to realize that also uncertainty should in fact not be treated as an absolute, but with respect to the prior knowledge of an observer who has access to a quantum system E [46]. As an illustration, recall the example from above where ρ_{AE} is the maximally entangled state. In this case, for any measurement on A , there is a corresponding measurement on E that reproduces the measurement outcomes. I.e., there is no uncertainty in the outcome at all! In order to take into account possibly quantum information about A , one needs to prove new entropic uncertainty relations that would have an additional term quantifying the quantum side information. Unfortunately, up to this day, we only know such relations for *two* measurements [4,33,41]. Intuitively, uncertainty relations for two measurements are much easier to prove than relations for more measurements as in this case uncertainty coincides with another foundational notion in quantum information, complementarity. This notion is relevant when we perform two measurements in succession and was an essential ingredient in the proofs. However, it does not carry over to three or more measurements. Here, we prove the following results.

Uncertainty Relations with Quantum Side Information for More than Two Measurements. We show that any set of measurements forming a QC-extractor yields an entropic uncertainty relation *with respect to*

quantum side information. We thereby obtain relations both for the usual von Neumann (Shannon) entropy, as well as the min-entropy. The latter is relevant for cryptographic applications. This yields the first uncertainty relations with quantum side information for more than two measurements. From our QC-extractors, we obtain strong uncertainty relations for (almost) unitary 2-designs, measurements in a full set of mutually unbiased bases (MUBs) on the whole space, as well as on many single qudits. The latter are the measurements used e.g., in the six-state protocol of QKD, and are particularly relevant for applications in quantum cryptography.

Note that uncertainty relations in terms of the min-entropy effectively help us to bound $H_{\min}(X|ER)$, where R is the seed for the QC-extractor (see Section 4 for details). For example, for the full set of MUBs we prove that

$$H_{\min}(X|ER) \gtrsim \log |A| + H_{\min}(A|E), \quad (2)$$

where the output of the measurements is called X . Since $H_{\min}(A|E)$ is negative when A and E are entangled, one obtains less uncertainty in this case (as expected when considering the example of a maximally entangled state given above). Of course, given such a bound, we could in turn apply a CC-extractor to the weakly random string X to obtain a uniform K . This underscores the beautiful relation between the concept of randomness extraction from a quantum state, and the notion of uncertainty relations with side information in quantum physics. From a QC-extractor, we obtain uncertainty relations. In turn, from any measurements inducing strong uncertainty relations *plus* a CC-extractor, we obtain a QC-extractor.⁷

1.3 Application to Cryptography

Our second application is to proving security in the noisy-storage model. Unfortunately, it turns out that even quantum communication does not enable us to solve two-party cryptographic problems between two parties that do not trust each other [25]. Such problems include e.g., the well-known primitives bit commitment and oblivious transfer [26,7,30,5], of which merely very weak variants are possible. How can this be when quantum communication offers such great advantages when it comes to distributing encryption keys? Intuitively, the security proof of QKD is considerably simplified by the fact that Alice and Bob do trust each other, and can collaborate to check for any eavesdropping activity. For example, as mentioned above, when Alice and Bob share a state ρ_{ABE} , where the eavesdropper holds E , they can use up part of the state to obtain an estimate of $H_{\min}(X|E)$, where X is a measurement outcome of the remaining part of Alice's system.

Yet, since two-party cryptographic protocols are a central part of modern cryptography, one is willing to make assumptions on how powerful the adversary can be in order to obtain security. Classically, these assumptions typically

⁷ Note that measurements plus a classical post-processing effectively forms a new, larger, set of measurements.

consist of two parts. First, one assumes that a particular problem requires a lot of computational resources to solve in some precise complexity-theoretic sense. Second, one assumes that the adversary does indeed have insufficient computational resources. However, we might instead ask whether there are other, more *physical* assumptions that enable us to solve such tasks?

Classically, it is possible to obtain security, when we are willing to assume that the adversary's *classical* memory is limited in size [29,6]. Yet, apart from the fact that classical storage is by now cheap and plentiful, the beautiful idea of assuming a limited classical storage has one rather crucial caveat: *any* classical protocol in which the honest players need to store n classical bits to execute the protocol can be broken by an adversary who is able to store more than $O(n^2)$ bits [15]. Motivated by this unsatisfactory gap, it was thus suggested to assume that the attacker's *quantum* storage was bounded [11,10], or, more generally, noisy [44,36,24]. The central assumption of the so-called *noisy-storage model* is that during waiting times Δt introduced in the protocol, the adversary can only keep quantum information in his quantum storage device \mathcal{F} . Otherwise, the attacker may be all powerful. In particular, he can store an unlimited amount of classical information, and perform computations 'instantaneously'. The latter implies that the attacker could encode his quantum information into an arbitrarily complicated error correcting code to protect it from any noise in \mathcal{F} (see Section 5 for details). Of particular interest are thereby quantum memories consisting of N 'memory cells', each of which undergoes some noise described by a channel \mathcal{N} . That is, the memory device is of the form $\mathcal{F} = \mathcal{N}^{\otimes N}$. Note that the bounded storage model is a special case, where each memory cell is just one qubit, and \mathcal{N} is the identity channel. To relate the number of transmitted qubits n to the size of the storage device one typically chooses the *storage rate* ν such that $N = \nu \cdot n$. We follow this convention here to ease comparison with earlier work.

Since its inception [44], it was clear that security in the noisy-storage model should be related to the question of how much information the adversary can send through his noisy storage device. That is, the capacity of \mathcal{F} to transmit quantum information. Initial progress was made in [24] where security was linked to the storage device's ability to transmit *classical* information and shown against fully general attacks.⁸ Further progress was made only very recently, linking the security to the so-called entanglement cost of the storage device [3], which lies between its classical and quantum capacities.

Security and the Quantum Capacity. Here, we finally resolve the question of linking security in the noisy-storage model to the quantum capacity of the storage device. More precisely, we show that any two-party cryptographic primitive can be implemented securely under the assumption that

⁸ Before [24], security was only shown under the additional assumption that the adversary attacks each qubit individually [44]. Whereas this may sound similar to problems in QKD, note that the setting is entirely different when proving security between two mutually distrustful parties, and security in QKD does not imply security in this model.

the adversary is restricted to using a quantum storage device of the form $\mathcal{F} = \mathcal{N}^{\otimes \nu \cdot n}$ by means of a protocol transmitting n qubits whenever

$$\nu \cdot \mathcal{Q}(\mathcal{N}) < 1 , \text{ and } 2 - \log(3) \lesssim \nu \cdot \gamma^Q(\mathcal{N}, 1/\nu) , \quad (3)$$

where $\mathcal{Q}(\mathcal{N})$ is the quantum capacity of the channel \mathcal{N} and $\gamma^Q(\mathcal{N}, 1/\nu)$ is the so-called strong converse parameter of \mathcal{N} for sending information through \mathcal{F} at rate $R = 1/\nu$. Note that the second condition actually *does* favor small ν , since $\gamma^Q(\mathcal{N}, 1/\nu)$ is large whenever the rate $R = 1/\nu$ is large. A similar statement can be obtained for general channels \mathcal{F} (see Section 5 for details).

We prove our result by showing the security of a simple quantum protocol for the cryptographic primitive *weak string erasure* [24], which is known to be universal for two-party secure computation [24]. To this end, we employ the bitwise QC-extractor for measurements of single qubits, each in one of three MUBs, known from the six-state protocol in QKD.

2 Preliminaries

In this section, we briefly recall the definitions and notations we need. More details can be found in the full version.

In quantum mechanics, a system such as Alice's or Bob's labs are described mathematically by *Hilbert spaces*, denoted by A, B, C, \dots . Here, we follow the usual convention in quantum cryptography and assume that all *Hilbert spaces* are finite-dimensional. We write $|A|$ for the dimension of A . The set of linear operators on A is denoted by $\mathcal{L}(A)$. A *quantum state* ρ_A is an operator $\rho_A \in \mathcal{S}(A)$, where $\mathcal{S}(A) = \{\sigma_A \in \mathcal{L}(A) \mid \sigma_A \geq 0, \text{tr}(\sigma_A) = 1\}$. For a bipartite system $A = A_1 A_2$, we define the *measurement map* $\mathcal{T}_{A \rightarrow A_1} : \mathcal{L}(A) \rightarrow \mathcal{L}(A_1)$,

$$\mathcal{T}(\cdot)_{A \rightarrow A_1} = \sum_{a_1 a_2} \langle a_1 a_2 | (\cdot) | a_1 a_2 \rangle |a_1\rangle \langle a_1| , \quad (4)$$

where $\{|a_1\rangle\}, \{|a_2\rangle\}$ are (standard) orthonormal bases of A_1, A_2 respectively. When applying a unitary transformation U_j followed by the measurement map $\mathcal{T}_{A \rightarrow A_1}$, we obtain new measurements which we denote by $\mathcal{M}_{A \rightarrow K_1}^j$. Here the relabeling $A_1 \rightarrow K_1$ accounts for the fact that the output system is classical.

The *conditional min-entropy* of a state $\rho_{AB} \in \mathcal{S}(AB)$ is defined as

$$H_{\min}(A|B)_\rho = \max_{\sigma_B \in \mathcal{S}(B)} H_{\min}(A|B)_{\rho|\sigma} \quad (5)$$

$$\text{with } H_{\min}(A|B)_{\rho|\sigma} = \max \left\{ \lambda \in \mathbb{R} : 2^{-\lambda} \cdot \mathbb{I}_A \otimes \sigma_B \geq \rho_{AB} \right\} .$$

The smoothed version is defined by $H_{\min}^\varepsilon(A|B)_\rho = \max_{\tilde{\rho}_{AB} \in \mathcal{B}^\varepsilon(\rho_{AB})} H_{\min}(A|B)_{\tilde{\rho}}$, where $\mathcal{B}^\varepsilon(\rho)$ is the set of states at a distance at most ε from ρ . We use the purified distance as the distance measure [40].

3 Quantum to Classical Randomness Extractors

To understand the definition of quantum extractors, it is convenient to see a classical extractor as a family of (deterministic) permutations acting on the possible values of the source. This family of permutations should satisfy the following property: for any probability distribution on input bit strings with high min-entropy, applying a typical permutation from the family to the input induces an almost uniform probability distribution on a prefix of the output. We define a quantum to classical extractor in a similar way by allowing the operations performed to be general unitary transformations followed by a measurement in the computational basis.

Definition 1. Let $A = A_1 A_2$ with $n = \log |A|$. Let $\mathcal{T}_{A \rightarrow A_1}$ be the measurement map defined in Equation (4).

For $k \in [-n, n]$ and $\varepsilon \in [0, 1]$, a (k, ε) -QC-extractor is a set $\{U_1, \dots, U_L\}$ of unitary transformations on A such that for all states $\rho_{AE} \in \mathcal{S}(AE)$ with $H_{\min}(A|E)_\rho \geq k$, we have

$$\frac{1}{L} \sum_{i=1}^L \left\| \mathcal{T}_{A \rightarrow A_1} \left((U_i \otimes \mathbb{I}_E) \rho_{AE} (U_i^\dagger \otimes \mathbb{I}_E) \right) - \frac{\mathbb{I}_{A_1}}{|A_1|} \otimes \rho_E \right\|_1 \leq \varepsilon. \quad (6)$$

Observe that Definition 1 only allows a specific form of measurements obtained by applying a unitary transformation followed by a measurement in the computational basis of A_1 . The reason we use this definition is that we want the output of the extractor to be determined by the source and the choice of the seed. In the quantum setting, a natural way of translating this requirement is by imposing that an adversary holding a system that is maximally entangled with the source can perfectly predict the output. This condition is satisfied by the form of measurements dictated by Definition 1. Allowing generalized measurements (POVMs) already (implicitly) allows the use of randomness for free. Note also, that in the case where the system E is trivial, a $(0, \varepsilon)$ -QC-extractor is the same as an ε -metric uncertainty relation [16].

3.1 Examples and Limitations of QC-Extractors

The following is immediate using a general decoupling result from [13,14].

Theorem 1 (Unitary 2-designs are QC-extractors). Let $A = A_1 A_2$ with $n = \log |A|$. For all $k \in [-n, n]$ and all $\varepsilon > 0$, a unitary 2-design $\{U_1, \dots, U_L\}$ on A is a (k, ε) -CQ-extractor with output size

$$\log |A_1| = \min(n, n + k - 2 \log(1/\varepsilon)). \quad (7)$$

The following theorem shows that choosing unitaries at random defines a QC-extractor with high probability. The seed size in this case is of the same order as the output size of the extractor. We expect that a much smaller seed size would be sufficient.

Theorem 2 (Random unitaries are QC-extractors). Let $A = A_1 A_2$ with $n = \log |A|$ and $\mathcal{T}_{A \rightarrow A_1}$ be the measurement map defined in Equation (4). Let $\varepsilon > 0$, c be a sufficiently large constant, and

$$\log |A_1| \leq n + k - 4 \log(1/\varepsilon) - c \text{ and } \log L \geq \log |A_1| + \log n + 4 \log(1/\varepsilon) + c. \quad (8)$$

Then, choosing L unitaries $\{U_1, \dots, U_L\}$ independently according to the Haar measure defines a (k, ε) -QC-extractor with high probability.

The proof uses one-shot decoupling techniques [14, 38, 13] combined with an operator Chernoff bound [1]. We now give some limitations on the output size and seed size of QC-extractors. The following proposition shows that even if we are looking for a QC-extractor that works for a particular state ρ_{AE} , the output size is at most roughly $n + H_{\min}(A|E)_\rho$, where n denotes the size of the input.

Proposition 1 (Upper bound on the output size). Let $A = A_1 A_2$, $\rho_{AE} \in \mathcal{S}(AE)$, $\{U_1, \dots, U_L\}$ a set of unitaries on A , and $\mathcal{T}_{A \rightarrow A_1}$ defined as in Equation (4), such that, $\frac{1}{L} \sum_{i=1}^L \left\| \mathcal{T}_{A \rightarrow A_1} \left(U_i \rho_{AE} U_i^\dagger \right) - \frac{\mathbb{I}_{A_1}}{|A_1|} \otimes \rho_E \right\|_1 \leq \varepsilon$. Then,

$$\log |A_1| \leq \log |A| + H_{\min}^{\sqrt{\varepsilon}}(A|E)_\rho.$$

The proof uses monotonicity properties of the min-entropy. Concerning the seed size, a simple argument shows that the number of unitaries of a QC-extractor has to be at least about $1/\varepsilon$. It is interesting to observe that in the case where the system E is trivial (or classical), this bound is almost tight. In fact, it was shown in [16] that in this case, there exists QC-extractors with $L = O(\log(1/\varepsilon)\varepsilon^{-2})$ unitaries. This is a difference with classical extractors for which the number of possible values of the seed is at least $\Omega((n-k)\varepsilon^{-2})$ [32].

3.2 Full Set of Mutually Unbiased Bases (MUBs)

We saw that unitary 2-designs define QC-extractors. As unitary 2-designs also define QQ-extractors, it is natural to expect that we can build smaller and simpler sets of unitaries if we are only interested in extracting random classical bits. In fact, in this section, we construct simpler sets of unitaries that define a QC-extractor. Two ingredients are used: a full set of mutually unbiased bases and a family of pair-wise independent permutations.

A set of unitaries $\{U_1, \dots, U_L\}$ acting on A is said to define *mutually unbiased bases* if for all elements $|a\rangle, |a'\rangle$ of the computational basis of A , we have $|\langle a' | U_j U_i^\dagger | a \rangle|^2 \leq |A|^{-1}$ for all $i \neq j$. In other words, a state described by a vector $U_i^\dagger |a\rangle$ of the basis i gives a uniformly distributed outcome when measured in basis j for $i \neq j$. For example the two bases, sometimes called computational and Hadamard bases (used in most quantum cryptographic protocols), are mutually unbiased. There can be at most $|A| + 1$ mutually unbiased bases for A . Constructions of full sets of $|A| + 1$ MUBs are known in prime power dimensions [47].

A family \mathcal{P} of permutations of a set X is *pair-wise independent* if for all $x_1 \neq x_2$ and $y_1 \neq y_2$, and if π is uniformly distributed over \mathcal{P} , $\Pr\{\pi(x_1) = y_1, \pi(x_2) = y_2\} = \frac{1}{|X|(|X|-1)}$. If X has a field structure, i.e., if $|X|$ is a prime power, it is simple to see that the family $\mathcal{P} = \{x \mapsto a \cdot x + b : a \in X^*, b \in X\}$ is pair-wise independent. In the following, permutations of basis elements of a Hilbert space A should be seen as a unitary transformation on A . In this section and the following \mathcal{P} denotes this set of pair-wise independent permutations.

Theorem 3. *Let $\{U_1, \dots, U_{|A|+1}\}$ define a full set of mutually unbiased bases and \mathcal{P} be a family of pair-wise independent permutations. Then the set $\{PU_i : P \in \mathcal{P}, i \in [|A|+1]\}$ defines a (k, ε) -QC-extractor provided $\log |A_1| \leq n + k - 2\log(1/\varepsilon)$. The number of unitaries of this extractor is $L = (|A|+1)|\mathcal{P}| = (|A|+1)|A|(|A|-1)$.*

The idea of the proof is to bound the trace norm by the Hilbert-Schmidt (or L_2 -) norm of some well-chosen operator. This term is then computed exactly using the fact that the set of all the MUB vectors form a *complex projective* 2-design and the fact that the set of permutations is pair-wise independent.

3.3 Bitwise QC-Extractor

The unitaries we construct in this section are even simpler. They are composed of unitaries V acting on single qudits followed by permutations P of the computational basis elements. Note that this means that the measurements defined by these unitaries can be implemented with current technology. As the measurement \mathcal{T} commutes with the permutations P , we can first apply V , then measure in the computational basis and finally apply the permutation to the (classical) outcome of the measurement. In addition to the computational efficiency, the fact that the unitaries act on single qudits, is often a desirable property for the design of cryptographic protocols. In particular, the application to the noisy storage model that we present in Section 5 does make use of this fact.

Let $d \geq 2$ be a prime power so that there exists a complete set of mutually unbiased bases in dimension d . We represent such a set of bases by a set of unitary transformations $\{V_0, V_1, \dots, V_d\}$ mapping these bases to the standard basis. For example, for the qubit space ($d = 2$), we can choose

$$V_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad V_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad V_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & -1 \end{pmatrix}. \quad (9)$$

We define the set $\mathcal{V}_{d,n}$ of unitary transformations on n qudits by $\mathcal{V}_{d,n} := \{V = V_{u_1} \otimes \dots \otimes V_{u_n} | u_i \in \{0, \dots, d\}\}$. As in the previous section, \mathcal{P} denotes a family of pair-wise independent functions.

Theorem 4. *The set $\{PV : P \in \mathcal{P}, V \in \mathcal{V}_{d,n}\}$ is a (k, ε) -extractor provided $\log |A_1| \leq (\log(d+1)-1)n + \min\{0, k\} - 4\log(1/\varepsilon) - 7$. The number of unitaries of this extractor is $L = (d+1)^n d^n (d^n - 1)$.*

The analysis uses the same technique as in the proof of Theorem 3. The main difference is that we were not able to express the L_2 -norm exactly in terms of the conditional min-entropy $H_{\min}(A|E)_\rho$. We use some additional inequalities, which account for the slightly more complicated expression we obtain.

4 Application to Entropic Uncertainty Relations with Quantum Side Information

The first application of our result is to entropic uncertainty relations *with quantum side information*. It is not hard to prove than any set of unitaries $\{U_j\}_j$ that form a QC-extractor define measurements that satisfy entropic uncertainty relations with quantum side information. The measurement $\mathcal{M}_{A \rightarrow K}^j$ is defined by first performing the unitary U_j followed by a measurement in the standard basis. We denote the post-measurement state by

$$\rho_{KEJ} = \frac{1}{L} \sum_{j=1}^L \mathcal{M}_{A \rightarrow K}^j(\rho_{AE}) \otimes |j\rangle\langle j|_J , \quad (10)$$

where the classical register J tells us which measurement $\mathcal{M}_{A \rightarrow K}^j$ was applied. Here, we only state the most important uncertainty relations that are obtained from our constructions of QC-extractors. We refer the reader to the full version for a more detailed treatment.

Corollary 1. *Let $d \geq 2$ be a prime power and $\mathcal{M}_{A \rightarrow K}^j$ be the measurements defined by the unitaries $\{V_j\}_j = \mathcal{V}_{d,n}$ as defined in Theorem 4. For all $\varepsilon, \delta' > 0, \delta \geq 0$ such that $\varepsilon^2 > 2(\delta + \delta')$ and any state ρ_{AE} , we have*

$$\begin{aligned} H_{\min}^\varepsilon(K|EJ)_\rho &\geq n \cdot (\log(d+1) - 1) + \min \left\{ 0, H_{\min}^\delta(A|E)_\rho - \log \left(\frac{2}{\delta'^2} + \frac{1}{1-2\delta} \right) \right\} \\ &\quad - \log \left(\frac{1}{(\varepsilon^2/2 - 2(\delta + \delta'))^2} \right) - 2, \end{aligned}$$

where ρ_{KEJ} is defined in (10).

Concerning uncertainty relations for the von Neumann (Shannon) entropy, we would mainly like to point to the following proposition.

Proposition 2. *Let $\rho_{AE} \in \mathcal{S}(AE)$ with A an n -qudit system, i.e., $|A| = d^n$. Then, the measurements given by the single qudit unitaries as defined in Section 3.3, give rise to the entropic uncertainty relation*

$$\frac{1}{L} \sum_{j=1}^L H(K|E)_{\rho^j} \geq n \cdot (\log(d+1) - 1) + \min \{0, H(A|E)_\rho\} . \quad (11)$$

Note that the conditional von Neumann entropy on the rhs can become negative and quantifies the entanglement present in the initial state.

Previously, uncertainty relations with quantum side information were only known for two measurements [4,33,41].

5 Applications to Security in the Noisy-Storage Model

As a second application, we solve the long standing question of relating the security of cryptographic protocols in the noisy-storage model [44,36,43,24] to the quantum capacity. We will state our main theorem and an example - a more gentle explanation and the protocol can be found in the full version. In [24] it was shown that bit commitment and oblivious transfer, and hence any two-party secure computation [20], can be implemented securely given access to a simpler primitive called *weak string erasure (WSE)*. It is hence enough to prove the security of WSE, and we will follow this approach here.

Informally, weak string erasure achieves the following task - a formal definition [24,28] can be found in the full version. WSE takes no inputs from either Alice and Bob. Alice receives as output a randomly chosen string $X^n = X_1, \dots, X_n \in \{0, 1\}^n$. Bob receives a randomly chosen subset $\mathcal{I} \in [n]$ and the substring $X_{\mathcal{I}}$ of X^n . Randomly chosen thereby means that each index $i \in [n]$ has some fixed probability p of being in \mathcal{I} . Originally, $p = 1/2$ [24], but any probability $0 < p < 1$ allows for the implementation of oblivious transfer [28]. The security requirements of weak string erasure are that Alice does not learn \mathcal{I} , and Bob's min-entropy given all of his information B is bounded as $H_{\min}(X|B) \geq \lambda n$ for some parameter $\lambda > 0$. To summarize all relevant parameters, we thereby speak of a $(n, \lambda, \varepsilon, p)$ -WSE scheme.

For simplicity, we here include the general statement in terms of the channel fidelity F_c , and refer to the full version for an expression in terms of the strong converse parameter γ^Q . The channel fidelity is an important concept in quantum information as it is used as a measure of success of how well a channel can transmit quantum information. Very roughly, we can determine the maximum size of a quantum state that can be transmitted over \mathcal{F} with an error of at most ε , by computing the maximum size of an input system A such that $\max_{\mathcal{D}, \mathcal{E}} F_c(\mathcal{D} \circ (\mathcal{F} \otimes \mathcal{I}_M) \circ \mathcal{E}) \geq 1 - \varepsilon$ where \mathcal{E} and \mathcal{D} are encoding and decoding maps respectively, and M is a system allowing for free feed forward classical communication. This yields the ε -error quantum capacity.

Theorem 5. *Let Bob's storage device be given by \mathcal{F} . For any choice of constant parameters $\varepsilon, \delta' > 0$, Protocol 1 implements an $(n, \lambda, \varepsilon, 1/3)$ -WSE with*

$$\lambda = \log(3) - 1 - \frac{1}{n} \left(\max_{\mathcal{D}, \mathcal{E}} \log 2^n F_c(\mathcal{D} \circ (\mathcal{F} \otimes \mathcal{I}_M) \circ \mathcal{E}) + \kappa + \xi + 1 \right), \quad (12)$$

where $\kappa = \log(2/\delta'^2 + 1)$ and $\xi = \log\left(1/\left(\varepsilon^2/2 - \delta'\right)^2\right)$.

To get some intuition about the parameters above, we consider the example of bounded, noise-free, storage. The quantum capacity of the one qubit identity channel $\mathcal{N} = \mathcal{I}_2$ is simply $Q_{\rightarrow}(\mathcal{I}_2) = 1$. When Bob can store $\nu \cdot n$ qubits, i.e., his storage device is of the form $\mathcal{F} = \mathcal{I}_2^{\otimes \nu n}$ then security for any two-party protocol is possible if $\nu \lesssim \log(3) - 1 \approx 0.585$.

It should be noted that the parameters obtained here for the case of bounded storage are slightly worse than what was obtained in [28] where security was shown to be possible for $\nu < 2/3$ instead of $\nu \lesssim 0.585$. This is due to the fact that the lower bound 0.585 in our uncertainty relation stems from an expression involving the *collision* entropy rather than the Shannon entropy. We emphasize, however, that for the practically relevant regime of $n \lesssim 10^6$ our exact bound is still better for the same error parameters. Our result resolves the long standing question of relating security to the quantum capacity, and opens the door for improved results on strong converse parameters for *any* kind of storage device to be applied immediately to obtain security parameters.

6 Discussion and Outlook

Motivated by the problem of using physical resources to extract true classical randomness, we introduced the concept of quantum-to-classical randomness extractors. We emphasize that these QC-extractors also work against quantum side information. We showed that for a QC-extractor to distill randomness from a quantum state ρ_{AE} , the relevant quantity to bound is the conditional min-entropy $H_{\min}(A|E)_{\rho}$. This is in formal analogy with classical-to-classical extractors, in which case the relevant quantity is $H_{\min}(X|E)_{\rho}$.

We proceeded by showing various properties of QC-extractors and giving several examples for QC-extractors. Table 1 gives a comparison between our results on QC-extractors and known results about CC-extractors (holding against quantum side information as well). It is eye-catching that there is a vast difference between the upper and lower bounds for the seed size of QC-extractors. We were only able to show the existence of QC-extractors with seed length roughly the output size m , but we believe that it should be possible to find QC-extractors with much smaller seeds, say $O(\text{polylog}(n))$ bits long, where n is the input size. However, different techniques are needed to address this question.

It is interesting to note that our results do indeed lend further justification to use Bell tests to certify randomness created by measuring a quantum system [8,31,9]. Note that for a tripartite pure state ρ_{ABE} where we want to

Table 1. Known bounds on the seed size and output size in terms of (qu)bits for different kinds of (k, ε) -randomness extractors. n refers to the number of input (qu)bits, m the number of output (qu)bits and k the min-entropy of the input $H_{\min}(A|E)$. Note that for QC-extractors, k can be as small as $-n$. Additive absolute constants are omitted. The symbol (NE) denotes non-explicit constructions.

		CC-extractors	QC-extractors
Seed	Low. bound	$\log(n - k) + 2 \log(1/\varepsilon)$ [32]	$\log(1/\varepsilon)$
	Upp. bounds	$c \cdot \log(n/\varepsilon)$ [12]	$m + \log n + 4 \log(1/\varepsilon)$ [Th 2] (NE) $3n$ [Th 3]
Output	Upp. bound	$k - 2 \log(1/\varepsilon)$ [32]	$n + H_{\min}^{\sqrt{\varepsilon}}(A E)$ [Pr 1]
	Low. bound	$k - 2 \log(1/\varepsilon)$ [19,35,42]	$n + k - 2 \log(1/\varepsilon)$ [Th 3]

create classical randomness by means of QC-extractors on A , we have to find a lower bound on $H_{\min}(A|E)_\rho$. But by the duality relation for min/max-entropies we have $H_{\min}(A|B)_\rho = -H_{\max}(A|B)_\rho$ [40], where the latter denotes the max-entropy as introduced [22]. Since $H_{\max}(A|B)_\rho$ is again a measure for the entanglement between A and B , one basically only has to do entanglement witnessing (e.g., Bell tests consuming part of the state) to ensure that the QC-extractor method can work (i.e., that $H_{\min}(A|E)_\rho$ is large enough). Note that any method to certify such an estimate would do and we could also use different measurements during the estimation process and the final extraction step. It would be interesting to know, if by using a particular QC-extractor, one can gain more randomness than in [8,31,9].

As the first application, we showed that every QC-extractor gives rise to entropic uncertainty relations with quantum side information for the von Neumann (Shannon) entropy and the min-entropy. Here the seed size translates into the number of measurements in the uncertainty relation. Since it is in general difficult to obtain uncertainty relations for a small set of measurements (except for the special case of two), finding QC-extractors with a small seed size is also worth pursuing from the point of view of uncertainty relations.

As the second application, we used the bitwise QC-extractor from Section 3.3 to show that the security in the noisy storage model can be related to the strong converse rate of the quantum storage; a problem that attracted quite some attention over the last few years. Here one can also see the usefulness of *bitwise* QC-extractors for quantum cryptography. Indeed, any bitwise QC-extractor would yield a protocol for weak string erasure. Bitwise measurements have a very simple structure, and hence are implementable with current technology. In that respect, it would be interesting to see if a similar QC-extractor can also be proven for only two (complementary) measurements per qubit. This would give a protocol for weak string erasure using BB84 bases as in [24].

We expect that QC-extractors will have many more applications in quantum cryptography, e.g., quantum key distribution. One possible interesting application could be to prove the security of oblivious transfer when purifying the protocol of [24]. Yet, it would require additional concepts of ‘entanglement sampling’ which still elude us.

References

1. Ahlswede, R., Winter, A.: Strong converse for identification via quantum channels. *IEEE Trans. Inform. Theory* 48, 569–579 (2010); Addendum *ibid* 49, 346 (2003), arXiv:quant-ph/0012127v2
2. Ben-Aroya, A., Schwartz, O., Ta-Shma, A.: Quantum expanders: Motivation and construction. *Theory of Computing* 6, 47–79 (2010)
3. Berta, M., Brandao, F., Christandl, M., Wehner, S.: Entanglement cost of quantum channels (2011), arXiv:1108.5357
4. Berta, M., Christandl, M., Colbeck, R., Renes, J.M., Renner, R.: The uncertainty principle in the presence of quantum memory. *Nat. Phys.* 6, 659 (2010), arXiv:0909.0950v4

5. Buhrman, H., Christandl, M., Hayden, P., Lo, H.-K., Wehner, S.: Security of quantum bit string commitment depends on the information measure. *Phys. Rev. Lett.* 97, 250501 (2006), arXiv:quant-ph/0609237v2
6. Cachin, C., Maurer, U.M.: Unconditional Security against Memory-Bounded Adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 292–306. Springer, Heidelberg (1997)
7. Chau, H.F., Lo, H.-K.: Making an empty promise with a quantum computer. *Fortschritte der Physik* 46, 507–520 (1998); Republished in Braunstein, S. (ed.) Quantum Computing, where do we want to go tomorrow?, arXiv:quant-ph/9709053v2
8. Colbeck, R.: Quantum and relativistic protocols for secure multi-party computation. PhD thesis. University of Cambridge (2006), arXiv:0911.3814v2
9. Colbeck, R., Kent, A.: Private randomness expansion with untrusted devices. *J. Phys. A - Math. Gen.* 44, 095305 (2011), arXiv:1011.4474v3
10. Damgård, I.B., Fehr, S., Renner, R.S., Salvail, L., Schaffner, C.: A Tight High-Order Entropic Quantum Uncertainty Relation with Applications. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 360–378. Springer, Heidelberg (2007), arXiv:quant-ph/0612014v2
11. Damgård, I.B., Fehr, S., Salvail, L., Schaffner, C.: Cryptography in the Bounded-Quantum-Storage Model. In: Proc. IEEE FOCS, pp. 449–458 (2005), arXiv:quant-ph/0508222v2
12. De, A., Portmann, C., Vidick, T., Renner, R.: Trevisan’s extractor in the presence of quantum side information (2009), arXiv:0912.5514
13. Dupuis, F.: The Decoupling Approach to Quantum Information Theory. PhD thesis, Université de Montréal (2009), arXiv:1004.1641v1
14. Dupuis, F., Berta, M., Wullschleger, J., Renner, R.: The decoupling theorem (2010), arXiv:1012.6044v1
15. Dziembowski, S., Maurer, U.: On Generating the Initial Key in the Bounded-Storage Model. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 126–137. Springer, Heidelberg (2004)
16. Fawzi, O., Hayden, P., Sen, P.: From low-distortion norm embeddings to explicit uncertainty relations and efficient information locking. In: Proc. ACM STOC (2011), arXiv:1010.3007v3
17. Gavinsky, D., Kempe, J., Kerenidis, I., Raz, R., de Wolf, R.: Exponential separations for one-way quantum communication complexity, with applications to cryptography. In: Proc. ACM STOC, pp. 516–525. ACM (2007)
18. Hayden, P., Horodecki, M., Yard, J., Winter, A.: A decoupling approach to the quantum capacity. *Open. Syst. Inf. Dyn.* 15, 7–19 (2008), arXiv:quant-ph/0702005v1
19. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions. In: Proc. ACM STOC, pp. 12–24. ACM (1989)
20. Kilian, J.: Founding cryptography on oblivious transfer. In: Proc. ACM STOC, pp. 20–31 (1988)
21. König, R., Maurer, U., Renner, R.: On the power of quantum memory. *IEEE Trans. Inform. Theory* 51, 2391–2401 (2005), arXiv:quant-ph/0305154v3
22. König, R., Renner, R., Schaffner, C.: The operational meaning of min- and max-entropy. *IEEE Transactions on Information Theory* 55, 4674–4681 (2009), arXiv:0807.1338v1
23. König, R., Terhal, B.M.: The bounded-storage model in the presence of a quantum adversary. *IEEE Trans. Inform. Theory* 54, 749–762 (2008)

24. König, R., Wehner, S., Wullschleger, J.: Unconditional security from noisy quantum storage. *IEEE Trans. Inform. Theory* 58(3), 1962–1984 (2012), arXiv:0906.1030v3
25. Lo, H.-K.: Insecurity of quantum secure computations. *Phys. Rev. A* 56, 1154 (1997)
26. Lo, H.-K., Chau, H.F.: Is quantum bit commitment really possible? *Phys. Rev. Lett.* 78, 3410 (1997)
27. Maassen, H., Uffink, J.: Generalised entropic uncertainty relations. *Phys. Rev. Lett.* 60, 1103–1106 (1988)
28. Mandayam, P., Wehner, S.: Achieving the physical limits of the bounded-storage model. *Phys. Rev. A* 83, 022329 (2011), arXiv:1009.1596v2
29. Maurer, U.: Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptol.* 5, 53–66 (1992)
30. Mayers, D.: Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.* 78, 3414–3417 (1997)
31. Pironio, S., Acin, A., Massar, S., de la Giroday, A.B., Matsukevich, D.N., Maunz, P., Olmschenk, S., Hayes, D., Luo, L.: Random numbers certified by Bell’s theorem. *Nature* 464, 1021–1024 (2010), arXiv:0911.3427v3
32. Radhakrishnan, J., Ta-Shma, A.: Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Math.* 13, 2 (2000)
33. Renes, J.M., Boileau, J.-C.: Conjectured strong complementary information trade-off. *Phys. Rev. Lett.* 103, 020402 (2009), arXiv:0806.3984v2
34. Renner, R.: Security of quantum key distribution. *International Journal of Quantum Information* 6, 1 (2008), arXiv:quant-ph/0512258v2
35. Renner, R., König, R.: Universally Composable Privacy Amplification Against Quantum Adversaries. In: Kilian, J. (ed.) *TCC 2005. LNCS*, vol. 3378, pp. 407–425. Springer, Heidelberg (2005), arXiv:quant-ph/0403133v2
36. Schaffner, C., Terhal, B., Wehner, S.: Robust cryptography in the noisy-quantum-storage model. *Quantum Inf. Comput.* 9, 11 (2008), arXiv:0807.1333v3
37. Shaltiel, R.: Recent developments in explicit constructions of extractors. *Bulletin of the EATCS* 77, 67–95 (2002)
38. Szehr, O., Dupuis, F., Tomamichel, M., Renner, R.: Decoupling with unitary almost two-designs (2011), arXiv:1109.4348
39. Ta-Shma, A.: Short seed extractors against quantum storage. In: Proc. ACM STOC, pp. 401–408. ACM (2009)
40. Tomamichel, M., Colbeck, R., Renner, R.: Duality between smooth min- and max-entropies. *IEEE Trans. Inform. Theory* 56, 4674 (2010), arXiv:0907.5238v2
41. Tomamichel, M., Renner, R.: The uncertainty relation for smooth entropies. *Phys. Rev. Lett.* 106, 110506 (2011), arXiv:1009.2015v2
42. Tomamichel, M., Schaffner, C., Smith, A., Renner, R.: Leftover hashing against quantum side information. *IEEE Trans. Inform. Theory* 57(8), 5524–5535 (2011), arXiv:1002.2436v1
43. Wehner, S., Curty, M., Schaffner, C., Lo, H.-K.: Implementation of two-party protocols in the noisy-storage model. *Phys. Rev. A* 81, 052336 (2010), arXiv:0911.2302v2
44. Wehner, S., Schaffner, C., Terhal, B.: Cryptography from noisy storage. *Phys. Rev. Lett.* 100, 220502 (2008), arXiv:0711.2895v3
45. Wehner, S., Winter, A.: Entropic uncertainty relations - a survey. *New J. Phys.* 12, 025009 (2010), arXiv:0907.3704v1
46. Winter, A.: Quantum information: Coping with uncertainty. *Nat. Phys.* 6, 640 (2010)
47. Wootters, W.K., Fields, B.D.: Optimal state-determination by mutually unbiased measurements. *Ann. Physics* 191, 363–381 (1989)

Actively Secure Two-Party Evaluation of Any Quantum Operation

Frédéric Dupuis^{1,*}, Jesper Buus Nielsen^{2,**}, and Louis Salvail^{3,***}

¹ Institute for Theoretical Physics, ETH Zurich, Switzerland
dupuis@phys.ethz.ch

² Department of Computer Science, Aarhus University, Denmark
jbn@cs.au.dk

³ Université de Montréal (DIRO), QC, Canada
salvail@iro.umontreal.ca

Abstract. We provide the first two-party protocol allowing Alice and Bob to evaluate privately even against active adversaries any completely positive, trace-preserving map $\mathcal{F} \in L(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \rightarrow L(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$, given as a quantum circuit, upon their joint quantum input state $\rho_{\text{in}} \in D(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}})$. Our protocol leaks no more to any active adversary than an ideal functionality for \mathcal{F} provided Alice and Bob have the cryptographic resources for active secure two-party classical computation. Our protocol is constructed from the protocol for the same task secure against specious adversaries presented in [4].

1 Introduction

We provide the first active-secure two-party protocol for computing on quantum data. We look at a model where Alice and Bob hold an input ρ_{in} on registers \mathcal{A}_{in} and \mathcal{B}_{in} , where Alice holds register \mathcal{A}_{in} and Bob holds \mathcal{B}_{in} . They agree on a completely positive, trace-preserving (CPTP) map \mathcal{F} from registers $\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}$ to registers $\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}}$, and they want to compute $\rho_{\text{out}} = \mathcal{F}(\rho_{\text{in}})$ such that, at the end of the protocol, Alice is in possession of \mathcal{A}_{out} and Bob is in possession of \mathcal{B}_{out} . They want to do this in an actively secure manner. Our notion of active security is phrased via simulation, but intuitively it simply guarantees that any cheating Alice, even an infinitely powerful Alice, which might deviate from the protocol, can only affect the output of the protocol by replacing her own input and that she will at any point during the execution of the protocol only hold information which can be computed (efficiently) from either $\rho_{\text{in}}^{\mathcal{A}}$ or $\rho_{\text{out}}^{\mathcal{A}}$. The equivalent condition should hold for Bob.

* Supported by Canada's NSERC Postdoctoral Fellowship Program and by the Swiss National Science Foundation via the National Center of Competence QSIT.

** Partially supported by an European Research Council, Starting Grant, number 279447 and the Danish National Research Foundation and the National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation.

*** Supported by Canada's NSERC discovery grant, FREQUENCY(NSERC), the Quantum-Works networks(NSERC), and INTRIQ(FQRNT).

A simple example of such an \mathcal{F} is the quantum swap, where $\mathcal{A}_{\text{in}} = \mathcal{A}_{\text{out}} = \mathcal{A}$, $\mathcal{B}_{\text{in}} = \mathcal{B}_{\text{out}} = \mathcal{B}$, and $\rho_{\text{out}}^{\mathcal{A}} = \rho_{\text{in}}^{\mathcal{B}}$ and $\rho_{\text{out}}^{\mathcal{B}} = \rho_{\text{in}}^{\mathcal{A}}$. Securely implementing this unitary basically means to do an atomic swap of quantum states, which was shown impossible in [4] even against a restricted class of adversaries called *specious*. Extra assumptions are therefore needed to get unconditional security. Our way out is to look at a model where the two parties have access to an ideal functionality which allows them to securely do any *classical* computation on any *classical* data held jointly by the two parties. In this model we give an unconditionally secure protocol with active security. This is the first such protocol.

Formally, we use the notationally simpler model of [4], but it is easy to see that security in this model implies security in the stand-alone model of [11], as long as the simulator is poly-time. The stand-alone model of [11] allows, inside a secure quantum protocol, to replace a classical ideal functionality by a classical protocol which securely implements that ideal functionality against poly-time quantum adversaries. The result is a protocol secure against poly-time quantum adversaries. This, in particular, implies sequential security, i.e., if a protocol is secure, it remains secure when run in sequence with other secure protocols. Since the secure evaluation of any classical function with security against poly-time quantum adversaries can be done under the assumption that learning with errors is hard [13,11] and under the more general assumption that mixed commitment schemes exist [12], our poly-time simulator provides an active-secure two-party plain-model protocol for computing on quantum data with security against any poly-time quantum adversaries. This is the first such protocol.

1.1 Overview of Our Construction

We reuse many ideas from the protocol provided in [4], which gives a two-party protocol for computing on quantum data securely against so-called specious adversaries. The protocol therein is unconditionally secure given an ideal functionality for classical computation. Specious adversaries are a quantum version of the classical notion of passive adversaries. Technically, a specious adversary is an adversary which is allowed to deviate from the protocol, except that at any step of the protocol it should be able to reconstruct the honest state of the protocol from its current state. This basically allows it to purify itself and not much else.

In the protocol in [4], all wires are encrypted using a Pauli encryption: a qubit $|v\rangle$ is represented as $|V\rangle = X^x Z^z |v\rangle$, where the two uniform key bits x and z are secret-shared between Alice and Bob. For example, to secret-share x , Alice will hold a uniformly random bit x_A and Bob will hold a uniformly random bit x_B such that $x = x_A \oplus x_B$. All wires are independently encrypted like this, which ensures that intermediate states are perfectly hidden from both parties. Computation is then done “through the encryption”. The CPTP map \mathcal{F} is described by a quantum circuit made out of the universal set of gates $\mathcal{U}\mathcal{G}$ consisting of gates X , Y , Z , CNOT , $H = \frac{1}{\sqrt{2}}(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix})$, $P = (\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix})$, and $R = (\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{smallmatrix})$, together with a set of ancilla wires initialized in state $|0\rangle$, and a set of output wires that are discarded (or, in our case, that remain encrypted forever). The protocol evaluates each gate of \mathcal{F} while preserving privacy. Handling the Pauli gates X , Y and Z is easy, as they commute or anti-commute with the encryption operators.

As an example, assume that the parties want to apply an X gate to a qubit $|v\rangle$, i.e., compute an encryption of $|v'\rangle = X|v\rangle$. Since up to an overall phase factor $XZ = ZX$, it follows that $X|V\rangle = XX^x Z^z |v\rangle = X^x Z^z X|v\rangle = X^x Z^z |v'\rangle$. So, the evaluation is simply performed on the encrypted qubit $|V\rangle$ and the key bits x and z are maintained. For the remaining Clifford gates CNOT, H and P other “commutation” rules with X and Z are used. For H, it is used that $HX = ZH$ and $HZ = XH$, so H is simply applied to the encrypted qubit, and then the keys x and z are swapped: Alice sets $x'_A = z_A$ and $z'_A = x_A$ and similarly for Bob. This leaves the non-Clifford gate R, where the relation $RX^x Z^z = P^x X^x Z^z R$ almost does the job, except that it leaves an extra P^x . Getting rid of this requires quantum computation and a classical secure two-party computation which computes how the parties should update their key bits. After such a gate-by-gate computation of U on the encrypted qubits, the shares of the keys needed for learning the states on ones own output wires are swapped with the other party in one atomic step, using the ideal functionality for classical secure computation.

The protocol in [4] is actually secure against an active adversary up to the final swap of key bits, as the encryptions of the wires are guaranteed to be perfect, independent of the behavior of the other party, as the parties pick their own shares of the sharings of x and z . This means that no party can get any information on any intermediary states, no matter how it deviates. It can, however, easily force the computation to be incorrect, by applying gates to the encrypted states, so the full protocol is not active secure. We note, however, that [4] is secure against what we could call *ultimately specious adversaries*: Adversaries promising to attack in such a way that both parties always be able to reconstruct the correct state at the *end* of the protocol, but that can otherwise behave as they want. This follows from the active security in the middle and a theorem in [4] which says that any attack which always allows both parties to obtain the correct output can be simulated given just the output of the corrupted party—basically, there is no way to learn extra information without sometimes irrevocably destroying the state of the other party.

In this paper, we use this observation in a protocol proceeding along the same lines as [4] but where we force the adversary to be ultimately specious. This is done by not only encrypting the wires, but by unconditionally authenticating them. In addition, we commit the parties to their key bits, to allow the recipient to verify the key bits swapped at the end. Since an unconditional quantum authentication code is also an unconditionally secure encryption, we get a protocol with at least the security of [4], but with the added property that an adversary who deviates from the protocol will be caught. More technically, if all checks of the authentication code succeed, then the authenticated values collapse to the correct values. This forces the adversary to either be detected or be ultimately specious. Since we do all the checks of the authentication codes *before* any key bits are revealed, the case where the adversary is detected can be simulated by simply asking the ideal world to abort too. The case where the adversary is ultimately specious is simulated similarly to [4].

The main technical challenge is then to devise an authentication code with these two properties:

1. It allows to perform computation “through the authentication”.
2. It allows to check the authentication code without revealing what is authenticated.

The first property is important for hiding intermediate values during the computation. The second property is important when we force the adversary to either be detected or ultimately specious: when he is detected, he should learn nothing on the incorrect outputs.

We devise an authentication code with these properties based on the Clifford-based quantum authentication code proposed in [1]. It authenticates a quantum message using a random unitary implementable using gates X, Y, Z, CNOT, H and P. The authentication works as follows. In order to satisfy the second property, take a qubit $|v\rangle$ on some wire. Alice prepends n new wires, in the all-zero state $|0^n\rangle$. Then she applies a uniformly random $(n + 1)$ -bit Clifford operator A to the $n + 1$ wires. She then sends the state to Bob, who appends n more wires in the all-zero state and applies a uniformly random $(2n + 1)$ -bit Clifford operator B to the $2n + 1$ wires. We can write this as $|V\rangle = B(|0^n\rangle \otimes A(|v\rangle|0^n\rangle))$. The key is (A, B) . This authentication can be checked in two different ways. Either, apply B^\dagger to the authenticated state and check that the first n wires are all zero. Or, apply B^\dagger and then apply A^\dagger to the last $n + 1$ wires and check that the last n wires are all zero. One way is used for Alice to check that Bob did not change the authenticated value. The other is used by Bob to check Alice. In order to perform these without leakage, we use a 2-party classical ideal functionality as described below.

In our scheme, Alice will hold A as her share of the key and Bob will hold B as his key share. They are committed to their share by being committed to a poly-size classical description of the operator applied. We sketch why the scheme has the two necessary properties.

1. Since the authentication is performed using only Clifford gates, an approach as in [4] will allow to fairly easily apply Clifford gates “through the authentication”. Since the Clifford unitaries form a group, the operation consisting of decrypting a qubit, applying a Clifford gate to it and reauthenticating it using a different key is also Clifford unitary. Hence, we can apply a Clifford gate to a swaddled qubit simply by changing the authentication keys in the appropriate manner. More precisely, to execute the Clifford gate $G \in \mathfrak{C}_1$, Alice updates her key to $A(G^\dagger \otimes \mathbb{I}_n)$, and Alice and Bob use a TPC to update Alice’s commitment to her key. Executing CNOT gates is similar, but involves two swaddlings. The R-gate requires new techniques reminiscent of the fault tolerant implementation of it [14,6,7]. The details appear within, but we basically reduce it to securely producing a state $|0\rangle$, a magic state, a measurement in the computational basis, and applying secure Clifford gates together with a carefully chosen secure classical computation which tells the parties how to update their keys.
2. If Bob is to check an authenticated qubit, we simply give him $|V\rangle$ and he applies B^\dagger and measures the first n wires, rejecting if they are not all zero. After that he re-applies B to recover the authentication $|V\rangle$. If Alice is to perform the check, we perform a secure classical computation where the inputs are the committed descriptions of A and B which effectively swaps the inner and outer authentications (see below for details). Alice then holds the outermost authentication and can easily test its integrity.

The final state of the computation is obtained after each party reveals the authentication keys needed by the other party to open its output wires.

2 Preliminaries

The set of linear operators from and to Hilbert space \mathcal{A} is denoted by $L(\mathcal{A})$. The set of trace 1 positive semi-definite operators is denoted by $D(\mathcal{A})$; it is the set of quantum states for register \mathcal{A} . For $\rho, \rho' \in D(\mathcal{A})$, we denote by $\Delta(\rho, \rho') := \frac{1}{2}\|\rho - \rho'\|_1$ the trace norm distance between ρ and ρ' . Finally, we denote by \mathfrak{C}_n the set of Clifford operators on n qubits. More information can be found in [5].

2.1 Secure Two-Party Classical Computation against Quantum Adversaries

Our protocol will use various *classical* two-party computations throughout its execution, each modeled as an ideal functionality. Recent work [12,11] show that composable classical two-party computation protocols can be devised with security against quantum adversaries provided some classical computational assumptions hold against this class of adversaries. One example of such an assumption is *learning with errors is hard* [13,11]. The framework in [11] allows us to replace the ideal functionalities with such secure protocols. Here we therefore focus on proving security given the ideal functionalities.

In the following, the ideal functionality for string commitment will be denoted by id_{sc} . It is defined as follows:

$$\begin{aligned} \text{id}_{\text{sc}}((id, s), \perp) &= (\perp, (\text{id}, \text{committed})) \text{ if } id \text{ is new,} \\ \text{id}_{\text{sc}}(\perp, (id, s)) &= ((id, \text{committed}), \perp) \text{ if } id \text{ is new,} \\ \text{id}_{\text{sc}}(s, s') &= (\perp, \perp) \text{ otherwise .} \end{aligned}$$

In order for Alice to commit on $s \in \{0, 1\}^*$, Alice and Bob call $\text{id}_{\text{sc}}((id, s), \perp)$, where id is an unused identifier chosen by Alice. In order for Bob to commit on s , Alice and Bob call $\text{id}_{\text{sc}}(\perp, (id, s))$, where id is chosen by Bob. The opening of a commitment is performed by calling the ideal functionality id_{open} defined as:

$$\begin{aligned} \text{id}_{\text{open}}(id, id) &= (\perp, s) \text{ if } \text{id}_{\text{sc}}((id, s), \perp) \text{ was performed ,} \\ \text{id}_{\text{open}}(id, id) &= (s, \perp) \text{ if } \text{id}_{\text{sc}}(\perp, (id, s)) \text{ was performed ,} \\ \text{id}_{\text{sc}}(\cdot, \cdot) &= (\perp, \perp) \text{ otherwise .} \end{aligned}$$

Note that state has to be passed from $\text{id}_{\text{sc}}(,)$ to $\text{id}_{\text{open}}(,)$ for the above descriptions to make sense. Our framework exactly allows that an ideal functionality from an earlier round passes its state to an ideal functionality in a later round. In the framework of [11] they would be considered one ideal functionality, with a state. We prefer the above notation for brevity of later protocol descriptions.

Ideal functionalities computing functions applied to committed values can now be easily defined. Suppose that Alice and Bob want to let Alice learn a function $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ upon two committed values, s and s' , where s is committed upon by Alice under id and s' is committed upon by Bob under id' . It suffices to define the ideal functionality id_f^* as

$$\begin{aligned} \text{id}_f^*((id, id'), (id, id')) &= (f(s, s'), \perp) \text{ if } \text{id}_{\text{open}}(id, id) = (\perp, s) \wedge \text{id}_{\text{open}}(id', id') = (s', \perp), \\ \text{id}_f^*(\cdot, \cdot) &= (\perp, \perp) \text{ otherwise .} \end{aligned}$$

The same if it is Bob who is to learn the output, but with $\text{id}_f^*((id, id'), (id, id')) = (\perp, f(s, s'))$. The same construction can be used to implement any efficiently computable function f evaluated upon *any* number of committed values. The ideal functionality can also easily be extended to produce commitments to the outputs of f . It is this extended ideal functionality we use most often. Again, $\text{id}_f^*(\cdot, \cdot)$ will have to share state with $\text{id}_{\text{sc}}(\cdot, \cdot)$ and $\text{id}_{\text{open}}(\cdot, \cdot)$, which is allowed in our framework.

Note that all the above ideal functionalities are defined such that at most one party has a non-trivial output (i.e., an output which is not known before the inputs are provided). We avoid using functions where both parties have a non-trivial output as an easy way to deal with the problem that fairness in classical secure two-party computation is provably impossible for most functionalities. It is not hard to see that it follows from known completeness results of quantum-secure classical two-party computation[12,11] that the classical ideal functionalities specified above can be implemented with security against poly-time quantum adversaries. We skip the details of this, as our focus here is on using the classical ideal functionalities for constructing secure two-party protocols for computing on quantum data.

2.2 Clifford-Based Quantum Authentication

In order to detect misbehaviors of an active adversary, we will be evaluating a circuit upon authenticated quantum bits. We will be using a quantum authentication scheme (QAS) [2] based on Clifford operators introduced in [1] as our main building block.

Definition 2.1 (Quantum authentication scheme). A quantum authentication scheme is a set of encryption and decryption superoperators $\{(\mathcal{E}_k^{S \rightarrow C}, \mathcal{D}_k^{C \rightarrow SF}) : k \in \mathcal{K}\}$, where \mathcal{K} is the set of possible keys, S is the input system, C is the ciphertext system, and F is a “flag” system that contains either $|\text{acc}\rangle$ or $|\text{rej}\rangle$. A QAS is such that for all $k \in \mathcal{K}$, $(\mathcal{D}_k \circ \mathcal{E}_k)(\rho_S) = \rho_S \otimes |\text{acc}\rangle\langle\text{acc}|_F$.

A QAS is secure if it satisfies the following:

Definition 2.2 (Security of a QAS). Let $\mathcal{E}_k^{S \rightarrow C}$ and $\mathcal{D}_k^{C \rightarrow SF}$ be the encoder and decoder corresponding to key k . Then, we say that the QAS $(\mathcal{E}, \mathcal{D})$ is ε -secure if, for all attacks U_{CR} , there exists two CP maps $\mathcal{U}_{R \rightarrow R}^{\text{acc}}$ and $\mathcal{U}_{R \rightarrow R}^{\text{rej}}$ with $\mathcal{U}^{\text{acc}} + \mathcal{U}^{\text{rej}} = \mathbb{1}$ such that for all inputs ψ_{SR} , we have that for some fixed state Ω_S :

$$\begin{aligned} & \left\| \frac{1}{\#\mathcal{K}} \sum_{k \in \mathcal{K}} \mathcal{D}_k \left(U_{CR} \mathcal{E}_k (\psi_{SR}) U_{CR}^\dagger \right) \right. \\ & \quad \left. - (\mathcal{U}^{\text{acc}}(\psi_{SR}) \otimes |\text{acc}\rangle\langle\text{acc}|_F + \mathcal{U}^{\text{rej}}(\psi_R) \otimes \Omega_S \otimes |\text{rej}\rangle\langle\text{rej}|_F) \right\|_1 \leq \varepsilon . \end{aligned} \tag{1}$$

This definition can be shown to be equivalent to the existence of a simulator that interacts only with an ideal functionality in which Eve’s only choice is whether or not to destroy the state and cause a rejection.

Definition 2.3 (Clifford-based QAS[1]). Let S be an s -qubit system, A be an n -qubit system, and let $C = S \otimes A$. Let \mathcal{K} index all Clifford unitaries C_k on $s + n$ qubits. Then, the Clifford-based QAS is defined by the following encryption and decryption maps where $P_{\text{acc}}^{SA} = \mathbb{1}_S \otimes |0^n\rangle\langle 0^n|_A$ and $P_{\text{rej}}^{SA} = \mathbb{1}_{SA} - P_{\text{acc}}^{SA}$:

$$\begin{aligned}\mathcal{E}_k(\rho_S) &= C_k (\rho_S \otimes |0^n\rangle\langle 0^n|_A) C_k^\dagger , \\ \mathcal{D}_k(\sigma_{SA}) &= \text{tr}_A \left(P_{\text{acc}} C_k^\dagger \sigma_{SA} C_k P_{\text{acc}} \otimes |\text{acc}\rangle\langle \text{acc}|_F \right. \\ &\quad \left. + \text{tr}(P_{\text{rej}} C_k^\dagger \sigma_{SA} C_k) \pi_{SA} \otimes |\text{rej}\rangle\langle \text{rej}|_F \right) ,\end{aligned}$$

where π_{SA} is an arbitrary fixed state that the decoder outputs when it rejects the authentication.

The following establishes the security of the QAS based on random Clifford operators. The proof of security is more or less the same as in [1] :

Theorem 2.4 (Security of Clifford-based QAS). The QAS defined above is $\varepsilon(n)$ -secure for $\varepsilon(n) = 6 \times 2^{-n}$.

It should be mentioned that picking a random Clifford operation acting upon ℓ qubits requires to pick a uniformly random $\text{poly}(\ell)$ -bit classical key k and the mapping between k and the corresponding Clifford operation can be performed efficiently [6,1]. In other words, the key size of the Clifford-based QAS is polynomial in the number of qubits $\ell = s + n$ used to authenticate an s -qubit quantum state.

2.3 Two-Party Quantum Protocols

We define two-party strategies in a similar way as in [4,10], with some adaptations made for the fact that we are computing a CPTP map and not just a unitary and that we allow ideal functionalities of different rounds to share states (equivalent to considering one, stateful functionality). Two-party protocols for the evaluation of some CPTP map are particular cases of two-party strategies. Two-party strategies have access to some oracle in each round. An oracle is just a CPTP map acting on registers at both Alice and Bob. Oracles implement some functionalities like a communication channel or some more complex two-party functionalities.

An m -turn two-party strategy with oracle calls is defined by tuples of quantum operations $\mathcal{A} := (\mathcal{A}_1, \dots, \mathcal{A}_{m+1})$, $\mathcal{B} := (\mathcal{B}_1, \dots, \mathcal{B}_{m+1})$, and $\mathcal{O} := (\mathcal{O}_1, \dots, \mathcal{O}_m)$. For $i \in [1..m+1]$, operations $\mathcal{A}_i \in L(\mathcal{A}_{i-1} \otimes \mathcal{A}_{i-1}^{\mathcal{O},\text{out}}) \mapsto L(\mathcal{A}_i \otimes \mathcal{A}_i^{\mathcal{O},\text{in}})$ and $\mathcal{B}_i \in L(\mathcal{B}_{i-1} \otimes \mathcal{B}_{i-1}^{\mathcal{O},\text{out}}) \mapsto L(\mathcal{B}_i \otimes \mathcal{B}_i^{\mathcal{O},\text{in}})$ are the actions performed at turn i by Alice and Bob respectively. The operation $\mathcal{O}_i : L(\mathcal{A}_i^{\mathcal{O},\text{in}} \otimes \mathcal{O}_{i-1} \otimes \mathcal{B}_i^{\mathcal{O},\text{in}}) \mapsto L(\mathcal{A}_i^{\mathcal{O},\text{out}} \otimes \mathcal{O}_i \otimes \mathcal{B}_i^{\mathcal{O},\text{out}})$ models the oracle provided to Alice and Bob at turn i , and \mathcal{O}_i a register used for passing state from the oracle of one round to the oracle at the next round. The oracle at turn $i \in [1..m]$ takes place right after \mathcal{A}_i and \mathcal{B}_i have been applied. In particular, these operations set the input registers $\mathcal{A}_{i-1}^{\mathcal{O},\text{in}}$ and $\mathcal{B}_{i-1}^{\mathcal{O},\text{in}}$ for the call to \mathcal{O}_i . The outputs are available to Alice and Bob in the next turn, in the output registers $\mathcal{A}_i^{\mathcal{O},\text{out}}$ and $\mathcal{B}_i^{\mathcal{O},\text{out}}$. We make

one exception to this general form, the last turn (that is turn $m + 1$) of a strategy does not invoke any oracle, and is there simply to allow Alice and Bob to post-process the output of the last oracle.

Let $\Pi = (\mathcal{A}, \mathcal{B}, \mathcal{O}, m)$ be an m -turn two-party strategy with oracle calls. The final state of the interaction between \mathcal{A} and \mathcal{B} upon joint input state $\rho_{\text{in}} \in D(\mathcal{A}_0 \otimes \mathcal{B}_0 \otimes \mathcal{R})$, where \mathcal{R} is a reference system with $\dim \mathcal{R} = \dim \mathcal{A}_0 \dim \mathcal{B}_0$, is denoted by

$$\begin{aligned} [\mathcal{A} * \mathcal{B}]^{\mathcal{O}}(\rho_{\text{in}}) := & (\mathcal{A}_{m+1} \otimes \mathcal{B}_{m+1} \otimes \mathbb{1}_{L(\mathcal{R} \otimes \mathcal{O}_{m+1})}) \\ & (\mathbb{1}_{L(\mathcal{A}_m \otimes \mathcal{B}_m \otimes \mathcal{R})} \otimes \mathcal{O}_m)(\mathcal{A}_m \otimes \mathcal{B}_m \otimes \mathbb{1}_{L(\mathcal{R} \otimes \mathcal{O}_m)}) \\ & \dots (\mathbb{1}_{L(\mathcal{A}_1 \otimes \mathcal{B}_1 \otimes \mathcal{R})} \otimes \mathcal{O}_1)(\mathcal{A}_1 \otimes \mathcal{B}_1 \otimes \mathbb{1}_{L(\mathcal{R})})(\rho_{\text{in}}) . \end{aligned}$$

A *communication oracle* from Alice to Bob is modeled by having $\mathcal{A}_i^{\mathcal{O}} \approx \mathcal{B}_i^{\mathcal{O}}$ and letting \mathcal{O}_i move the state in $\mathcal{A}_i^{\mathcal{O}}$ to $\mathcal{B}_i^{\mathcal{O}}$. A classical ideal functionality, as those described in Sect. 2.1, can easily be made available as oracle calls. We assume that in each round i at most one classical ideal functionality is applied. The parties place their inputs in the appropriate registers $\mathcal{A}_{i-1}^{\mathcal{O}}$ and $\mathcal{B}_{i-1}^{\mathcal{O}}$. The operation of the oracle \mathcal{O}_i is as follows: it measures the input registers $\mathcal{A}_{i-1}^{\mathcal{O}}$ and $\mathcal{B}_{i-1}^{\mathcal{O}}$ to force classical inputs. Then, it applies the appropriate classical ideal functionality on those classical inputs plus its classical internal state found in \mathcal{O}_{i-1} . This produces outputs for the parties and a new internal state. The outputs are placed in $\mathcal{A}_i^{\mathcal{O}}$ and $\mathcal{B}_i^{\mathcal{O}}$. The new state is placed in \mathcal{O}_i .

A two-party hybrid protocol for $\mathcal{F} : L(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \rightarrow L(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$ between parties \mathcal{A} and \mathcal{B} upon joint input state $\rho_{\text{in}} \in D(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}} \otimes \mathcal{R})$ is defined as:

Definition 2.5. An m -turn two-party hybrid protocol $\Pi_{\mathcal{F}}^{\mathcal{O}} = (\mathcal{A}, \mathcal{B}, \mathcal{O}, m)$ for $\mathcal{F} : L(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \rightarrow L(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$ is a m -turn two-party strategy with oracle calls, where $\mathcal{A}_0 := \mathcal{A}_{\text{in}}$, $\mathcal{B}_0 := \mathcal{B}_{\text{in}}$, $\mathcal{A}_{m+1} := \mathcal{A}_{\text{out}}$, $\mathcal{B}_{m+1} := \mathcal{B}_{\text{out}}$, and where for all $\rho_{\text{in}} \in D(\mathcal{A}_0 \otimes \mathcal{B}_0 \otimes \mathcal{R})$, $\Delta([\mathcal{A} * \mathcal{B}]^{\mathcal{O}}(\rho_{\text{in}}), (\mathcal{F} \otimes \mathbb{1}_{\mathcal{R}})(\rho_{\text{in}})) = 0$. In the following, we often write simply two-party protocol to refer to a two-party hybrid protocol.

For $i \in [0..m]$, the joint state after turn $i + 1$ in $\Pi_{\mathcal{F}}^{\mathcal{O}}$ is denoted by $[\mathcal{A} * \mathcal{B}]_{i+1}^{\mathcal{O}}(\rho_{\text{in}}) := (\mathbb{1}_{L(\mathcal{B}_{i+1} \otimes \mathcal{A}_{i+1} \otimes \mathcal{R})} \otimes \mathcal{O}_{i+1})(\mathcal{A}_{i+1} \otimes \mathcal{B}_{i+1} \otimes \mathbb{1}_{L(\mathcal{R} \otimes \mathcal{O}_{i+1})})[\mathcal{A} * \mathcal{B}]_i^{\mathcal{O}}(\rho_{\text{in}})$, where $[\mathcal{A} * \mathcal{B}]_0^{\mathcal{O}}(\rho_{\text{in}}) := \rho_{\text{in}}$, and $[\mathcal{A} * \mathcal{B}]_{m+1}^{\mathcal{O}}(\rho_{\text{in}}) := [\mathcal{A} * \mathcal{B}]^{\mathcal{O}}(\rho_{\text{in}})$.

3 Modeling Active Security

We start by extending the framework in [4] to handle active security. Our model is very standard, defining security via simulation, but for completeness we describe and motivate the changes made in [5].

Let $\Pi_{\mathcal{F}}^{\mathcal{O}} = (\mathcal{A}, \mathcal{B}, \mathcal{O}, m)$ be a m -turn two-party hybrid protocol. Let $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ be adversaries in $\Pi_{\mathcal{F}}^{\mathcal{O}}$. We denote by $[\tilde{\mathcal{A}} * \mathcal{B}]^{\mathcal{O}}$ and $[\mathcal{A} * \tilde{\mathcal{B}}]^{\mathcal{O}}$ the resulting m -turn two-party strategies. We will as usual define the security of such actively attacked protocols by comparing to a simulation. The simulation is basically an ideally secure evaluation of \mathcal{F} .

The ideally secure protocol for evaluating \mathcal{F} would be in a world where \mathcal{F} actually existed as an oracle—the parties would simply call this oracle. We can, however, not expect any protocol to be as secure as this, as for most protocols we cannot ensure that

either both parties get the output or no party gets the output, known as fairness, which is ensured in the above ideal setting. We will therefore consider a setting where one of the parties learns its output first, and where the party learning it first, if corrupted, can prevent the other party from learning its output. We will only give the definition for the case where Alice learns first. Deriving the definition for the symmetric case where Bob learns first is trivial.

To avoid confusion between the parties in the real protocol and in the ideal protocol, we formulate the ideal protocol with parties Charleen, \mathcal{C} , and Dan, \mathcal{D} , taking the seats of Alice respectively Bob. So, we look at an ideal functionality $\mathcal{F} : \text{L}(\mathcal{C}_{\text{in}} \otimes \mathcal{D}_{\text{in}}) \rightarrow \text{L}(\mathcal{C}_{\text{out}} \otimes \mathcal{D}_{\text{out}})$, but where we keep a mental note reminding that $\mathcal{C}_{\text{in}} := \mathcal{A}_{\text{in}}$, $\mathcal{C}_{\text{out}} := \mathcal{A}_{\text{out}}$, $\mathcal{D}_{\text{in}} := \mathcal{B}_{\text{in}}$ and $\mathcal{D}_{\text{out}} := \mathcal{B}_{\text{out}}$.

The ideal protocol for \mathcal{F} is then a 2-turn two-party hybrid protocol $\Gamma_{\mathcal{F}} = (\mathcal{C}, \mathcal{D}, \mathcal{F}, 2)$, which lets the parties query \mathcal{F} in turn 1, but only letting \mathcal{C} see her output in turn 1. In turn 2, Charleen then inputs a bit f , with $f = 1$ indicating that Dan should receive his output and $f = 0$ indicating that Dan should not receive his output. Dan will receive f , and if $f = 1$ he will additionally be given his output. This is handled by the second oracle. Then the parties output whatever they received from the oracles. In the honest protocol, Charleen always inputs $f = 1$. We call this *the ideal protocol for evaluating \mathcal{F} without fairness for Dan*.

The ideal protocol $\Gamma_{\mathcal{F}} = (\mathcal{C}, \mathcal{D}, \mathcal{F}, 2)$ for \mathcal{F} without fairness for Dan:

1. By convention we have $\mathcal{C}_0 = \mathcal{C}_{\text{in}}$ and $\mathcal{D}_0 = \mathcal{D}_{\text{in}}$ and that \mathcal{O}_0 is empty. In the ideal protocol we let $\mathcal{C}_{1,\text{in}}^{\mathcal{O},\text{in}} = \mathcal{C}_0$ and $\mathcal{D}_{1,\text{in}}^{\mathcal{F},\text{in}} = \mathcal{D}_0$, we let \mathcal{C}_1 and \mathcal{D}_1 be empty, and $\mathcal{C}_1 = \mathbb{1}_{\text{L}(\mathcal{C}_0)}$ and $\mathcal{D}_1 = \mathbb{1}_{\text{L}(\mathcal{D}_0)}$. I.e., in turn 1 the parties simply send their inputs to the first oracle \mathcal{F}_1 . For the first oracle $\mathcal{F}_1 : \text{L}(\mathcal{C}_1^{\mathcal{F},\text{in}} \otimes \mathcal{O}_0 \otimes \mathcal{D}_1^{\mathcal{F},\text{in}}) \mapsto \text{L}(\mathcal{C}_1^{\mathcal{F},\text{out}} \otimes \mathcal{O}_1 \otimes \mathcal{D}_1^{\mathcal{F},\text{out}})$, we set $\mathcal{C}_1^{\mathcal{F},\text{out}} \approx \mathcal{C}_{\text{out}}$, $\mathcal{O}_1 \approx \mathcal{D}_{\text{out}}$ and we let $\mathcal{D}_1^{\mathcal{F},\text{out}}$ be empty. We let $\mathcal{F}_1 = \mathcal{F}$. I.e., \mathcal{F}_1 simply applies \mathcal{F} to the inputs supplied by the parties, sends Charleen's output to Charleen in $\mathcal{C}_1^{\mathcal{F},\text{out}}$ and saves Dan's output in the internal state \mathcal{F}_1 of the oracle, giving Dan no output so far.
2. We let $\mathcal{C}_2 \approx \mathcal{C}_1^{\mathcal{F},\text{out}} (\approx \mathcal{C}_{\text{out}})$ and we let $\mathcal{C}_2^{\mathcal{F},\text{in}}$ be a one qubit register, holding a qubit we name $|f\rangle$. We let $\mathcal{C}_2 = \mathbb{1}_{\text{L}(\mathcal{C}_1^{\mathcal{F},\text{out}}, \mathcal{C}_2)} \otimes |1\rangle$, i.e., it moves the output from the oracle from $\mathcal{C}_1^{\mathcal{F},\text{out}}$ to \mathcal{C}_2 and it sets $|f\rangle = |1\rangle$. We let $\mathcal{D}_2^{\mathcal{F},\text{in}}$ and \mathcal{D}_2 be empty, so there is no need to specify \mathcal{D}_2 .

For the second oracle $\mathcal{O}_2 : \text{L}(\mathcal{C}_2^{\mathcal{F},\text{in}} \otimes \mathcal{O}_1 \otimes \mathcal{D}_2^{\mathcal{F},\text{in}}) \mapsto \text{L}(\mathcal{C}_2^{\mathcal{F},\text{out}} \otimes \mathcal{O}_2 \otimes \mathcal{D}_2^{\mathcal{F},\text{out}})$, we let $\mathcal{C}_2^{\mathcal{F},\text{out}}$ and \mathcal{O}_2 be empty, and we let $\mathcal{D}_2^{\mathcal{F},\text{out}}$ be of the same dimension as \mathcal{D}_{out} , plus room for one qubit $|a\rangle$. It starts by measuring $|f\rangle$ in the computational basis. If $|f\rangle = |1\rangle$, it then sets $\mathcal{D}_2^{\mathcal{F},\text{out}}$ to hold $|a\rangle = |1\rangle$ along with the state in \mathcal{O}_1 . If $|f\rangle = |0\rangle$, it sets $\mathcal{D}_2^{\mathcal{F},\text{out}}$ to hold $|a\rangle = |0\rangle$ along with some fixed dummy state $|\perp\rangle$ of the right dimension to fill $\mathcal{C}_2^{\mathcal{F},\text{out}}$. I.e., if Charleen inputs $f = 1$, then Dan will get his output. If $f = 0$, Dan gets no output, except a bit $a = 0$ telling him that Charleen cheated him of his output (we say that Charleen *aborted* the computation).

3. We let $\mathcal{C}_3 \approx \mathcal{C}_2$ and $\mathcal{C}_3 = \mathbb{1}_{\text{L}(\mathcal{C}_2, \mathcal{C}_3)}$. We let $\mathcal{D}_3 \approx \mathcal{D}_2^{\mathcal{F},\text{out}}$ and $\mathcal{D}_3 = \mathbb{1}_{\text{L}(\mathcal{D}_2^{\mathcal{F},\text{out}}, \mathcal{D}_3)}$. I.e., in the last round the parties just output whatever they received from \mathcal{F} , with Dan possibly outputting a dummy state, in case of abort.

Consider the powers of a corrupted Charleen, $\tilde{\mathcal{C}}$. She might in the first round provide an alternative input for \mathcal{F} , possibly saving her original input, or a part thereof, in some

ancilla. This is *input substitution*. Her choice of alternative input can only depend on her own original input, not that of Dan. This is *input independence*. Then she learns only the output of the oracle. This is *privacy*. After learning her own output she might then specify that Dan is not to learn his output. This is the necessary *lack of fairness*. A corrupted Dan has similar powers, except that he cannot abort after seeing his output. We then say that a protocol is secure if it only allows attacks which could be mounted in the ideal protocol, i.e., it only allows the inevitable attack.

Definition 3.1. Let $\Pi_{\mathcal{F}}^{\mathcal{O}} = (\mathcal{A}, \mathcal{B}, \mathcal{O}, m)$ be a two-party hybrid protocol for \mathcal{F} : $L(\mathcal{A}_{in} \otimes \mathcal{B}_{in}) \rightarrow L(\mathcal{A}_{out} \otimes \mathcal{B}_{out})$. Let $\Gamma_{\mathcal{F}}$ be the ideal protocol for \mathcal{F} without fairness for Dan. Let $\delta \in [0..1]$. We say that $\Pi_{\mathcal{F}}^{\mathcal{O}}$ is δ -active secure without fairness for Bob, if for all adversaries $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ in $\Pi_{\mathcal{F}}^{\mathcal{O}}$, there exist adversaries $\tilde{\mathcal{C}}$ and $\tilde{\mathcal{D}}$ in $\Gamma_{\mathcal{F}}$ with sizes polynomial in the sizes of $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ such that for all input states $\rho_{in} \in D(\mathcal{A}_{in} \otimes \mathcal{B}_{in} \otimes \mathcal{R})$,

$$\Delta \left([\tilde{\mathcal{A}} \circledast \tilde{\mathcal{B}}]^{\mathcal{O}}(\rho_{in}), [\tilde{\mathcal{C}} \circledast \tilde{\mathcal{D}}]^{\mathcal{F}}(\rho_{in}) \right) \leq \delta \text{ & } \Delta \left([\mathcal{A} \circledast \tilde{\mathcal{B}}]^{\mathcal{O}}(\rho_{in}), [\mathcal{C} \circledast \tilde{\mathcal{D}}]^{\mathcal{F}}(\rho_{in}) \right) \leq \delta.$$

If a protocol is δ -active secure for $\delta = 0$, then it is called perfectly active secure. If it is δ -active secure for δ negligible in some security parameter n , then it is called statistically active secure (in n).

4 Securely Swaddling Wires to Ensure Ultimate Speciousness

The Clifford based QAS, described in Sect. 2.2, can be used to share the authentication of a quantum message in a way that allows any of the players, when helped by a TPC, to verify that a state has not been tampered with, and this without being able to get information on the encoded state. This primitive will be used extensively in our protocol to ensure that all players are *ultimately specious*. It relies on a string commitment scheme and secure TPCs provided as oracles.

The basic idea consists in swaddling each input wire w into a set of $2n$ dummy wires where n belongs to Alice and n belongs to Bob. No party will be able to extract information about the state of the original wire from the swaddling. Moreover, any attempt to modify the state of the original wire will be detected except with negligible probability in n . The subprotocol $\text{Swaddle}(w)$, described below, uses two application of the Clifford-based QAS (one on top of the other) in order to achieve this. Alice authenticates the state of her wire w and commits to her authentication key $K_{w,a}$ using identifier $id_A(w)$. She then sends the resulting system to Bob who authenticates it using key $K_{w,b}$ that he also commits upon using identifier $id_B(w)$. Bob sends back the resulting system to Alice allowing her to test the validity of the swaddling. The swaddling therefore uses a total of $2n + 1$ wires, where Alice holds the *innermost* and Bob the *outermost* key of the resulting swaddling $s(w)$ of wire w .

Consider now the test performed by Alice at Step 7. Let A and B be the Clifford operators corresponding to keys $K_{w,a}$ and $K_{w,b}$ respectively. Notice that Alice can easily test the authenticity of a swaddling when she holds the outermost authentication. She simply applies A^\dagger upon the $2n + 1$ wires, measures her n dummy wires in the computational basis to verify that they are in state $|0^n\rangle$. She then re-applies A to the $2n + 1$

wires of the swaddling. If Alice holds the innermost authentication of the swaddling, the testing procedure relies on the ideal functionality id_{TEST} defined as:

$$\text{id}_{\text{TEST}}((i_a, i_b), (i'_a, i'_b)) = \begin{cases} ((t, K'_{w,a}), K'_{w,b}) & \text{if } i_a = i'_a = \text{id}_A(w) \wedge i_b = i'_b = \text{id}_B(w), \\ (\perp, \perp) & \text{otherwise,} \end{cases}$$

where $K'_{w,a}$, and $K'_{w,b}$ correspond to random Clifford operations $A' \in \mathfrak{C}_{2n+1}$ and $B' \in \mathfrak{C}_{n+1}$ respectively, and t corresponds to Clifford $T \in \mathfrak{C}_{2n+1}$ subject to:

1. $\text{id}_{\text{OPEN}}(\text{id}_A(w), \text{id}_A(w)) = (\perp, K'_{w,a})$ and $\text{id}_{\text{OPEN}}(\text{id}_B(w), \text{id}_B(w)) = (K'_{w,b}, \perp)$, and
2. If Alice holds the innermost key of w then $T = (\mathbb{1}_n \otimes B')(A^\dagger \otimes \mathbb{1}_n)B^\dagger$.

Swaddle(\mathcal{W}), with $\mathcal{W} \subseteq \mathcal{A}$:

1. Alice initializes $n \cdot \#\mathcal{W}$ dummy wires in state $|0\rangle$.
2. **For** each $w \in \mathcal{W}$, Alice randomly chooses Clifford $K_{w,a}$ on $n + 1$ qubits, commits to it in $\text{id}_{\text{SC}}((\text{id}_A(w), K_{w,a}), \perp)$, and applies it to w and her dummies.
3. Alice **sends** these $\#\mathcal{W} \cdot (n + 1)$ wires to Bob.
4. Bob initializes $\#\mathcal{W} \cdot n$ dummy wires in the state $|0\rangle$.
5. **For** each $w \in \mathcal{W}$, Bob randomly chooses a Clifford $K_{w,b}$ on $2n + 1$ qubits, commits to it in $\text{id}_{\text{SC}}(\perp, (\text{id}_B(w), K_{w,b}))$, and applies it to the $n + 1$ wires received from Alice as well as his own dummies.
6. Bob **sends** all $\#\mathcal{W} \cdot (2n + 1)$ wires back to Alice. Let $\mathbf{s}(w)$ denote the resulting swaddling of $w \in \mathcal{W}$.
7. **For** each $w \in \mathcal{W}$, Alice **calls** $\text{TestSwaddling}(\mathbf{s}(w))$.

Condition 1 ensures that the authentication keys with identifiers $\text{id}_A(w)$ and $\text{id}_B(w)$ have been updated to hold values $K'_{w,a}$ and $K'_{w,b}$ respectively. Conditions 2 makes sure that the state $|\sigma\rangle$ of a valid swaddling $\mathbf{s}(w)$ satisfies $T|\sigma\rangle = |0^n\rangle \otimes B'(|\varphi\rangle \otimes |0^n\rangle)$, where $|\varphi\rangle$ is the logical state of $\mathbf{s}(w)$. Notice that Alice gets no information about Bob's Clifford B' if she had no information about B to start with.

TestSwaddling($\mathbf{s}(w)$), with Alice doing the testing:

1. **If** Alice holds the outermost authentication key $K_{w,a}$ **then**:
 - Alice applies A^\dagger on $\mathbf{s}(w)$, where A is the Clifford corresponding to string $K_{w,a}$,
 - Alice tests that her dummies are together in state $|0^n\rangle$. **If not then she aborts.**
 - Alice re-applies A on the $2n + 1$ qubits of the swaddling.
2. **Else** they **call** $((t, K'_{w,a}), K'_{w,b}) = \text{id}_{\text{TEST}}((\text{id}_A(w), \text{id}_B(w)), (\text{id}_A(w), \text{id}_B(w)))$,
 - **If** one party gets \perp in id_{TEST} **then abort**.
 - Alice applies T on $\mathbf{s}(w)$ where T is the Clifford operator corresponding to string t .
 - Alice tests that her dummies are together in state $|0^n\rangle$. **If not then she aborts.**
 - Alice applies A' to the the $2n + 1$ qubits of the swaddling (note that A' and B' are committed upon with identifiers $\text{id}_A(w)$ and $\text{id}_B(w)$ respectively).

At the end of $\text{TestSwaddling}(\mathbf{s}(w))$, Alice holds the outermost authentication key $K'_{w,a}$ of the resulting swaddling. Bob can do the testing by the exact same procedure provided the roles of Alice and Bob are reversed in $\text{TestSwaddling}(\mathbf{s}(w))$. The security of these procedures will be discussed in Sect. 5.5. Intuitively, we expect that $\text{TestSwaddling}(\mathbf{s}(w))$ allows parties to test that a swaddling $\mathbf{s}(w)$ has not been tampered with. It will also be used to test that each party transforms a swaddling $\mathbf{s}(w)$ the

way they should during the execution of the protocol. Notice that no information about the logical state of wire w leaks to any party in $\text{Swaddle}(w)$ and $\text{TestSwaddling}(s(w))$ since statistically secure authentication must also encrypt w [2].

At the end of our protocol, each party will be asked to verify that the other party's registers upon which the circuit is evaluated are all in the states they should be. This subprotocol is called TestAllSwaddlings . It simply consists in the execution of $\text{TestSwaddling}(s(w))$ for all wires w held by each party and its description can be found in [5].

The following subprotocol implements the obvious way the openings of all committed Clifford operators, thereby allowing Alice and Bob to get the final state of the computation. Notice that since Alice learns Bob's secret keys before she unveils her own, our protocol will lack fairness for Bob.

OpenAllSwaddlings:

1. For each wire $w \in \mathcal{A}_{\text{out}}$, Bob reveals to Alice his committed secret key encrypting w , with identifier $id_B(w)$, by calling $\text{id}_{\text{OPEN}}(id_B(w), id_B(w))$.
2. For each wire $w \in \mathcal{B}_{\text{out}}$, Alice reveals to Bob her committed secret key encrypting w , with identifier $id_A(w)$, by calling $\text{id}_{\text{OPEN}}(id_A(w), id_A(w))$.

5 Description of the Protocol

We first start by defining the various spaces on which (the honest) Alice and Bob will be working. The circuit that they want to execute acts on some wires in Alice's possession and some in Bob's possession, and these will be swaddled as described above. We will denote by $\mathcal{A}_u, \mathcal{B}_u$ the spaces corresponding to Alice and Bob's unswaddled wires, reserving \mathcal{A} and \mathcal{B} for the actual swaddled wires.

We first initialize all the qubits by swaddling them, we then perform each gate from the circuit one after the other on the authenticated data. Hence, we need to give subprotocols for the initialization as well as for each of the gates in our universal set.

For all Clifford gates (i.e., all gates in $\mathcal{U}\mathcal{G}$ except for R), the subprotocols are fairly simple: we use classical two-party computation to reveal a Clifford operation that executes the gate while updating the encryption key; the revealed Clifford then looks uniformly distributed and independent of everything else.

Implementing the R-gate is more involved. We use ideas from fault-tolerant computation, where this gate is implemented by doing gate teleportation via a so-called *magic state*: one prepares a special state (namely $|M\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$) and then use a teleportation-like circuit, which itself requires only Clifford gates and measurements, to execute the gate. The problem is then reduced to that of producing this magic state, which can be done by a distillation process. The distillation process that we use is exactly the one considered in [3] (where $|M\rangle$ is an “H-type magic state” in their language); a description of it can also be found in [5].

5.1 Main Protocol

We now give the full description of our protocol, denoted $\hat{\Pi}_{\mathcal{F}}^{\mathcal{O}'} = (\mathcal{A}, \mathcal{B}, \mathcal{O}', m)$, allowing to evaluate the CPTP map $\mathcal{F} : L(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \mapsto L(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$ upon joint

input state $\rho_{\text{in}} \in D(\mathcal{R} \otimes \mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}})$. The oracle list needed to run the protocol in the hybrid model is provided implicitly in Sect. 5. The operations performed at each step by \mathcal{A} and \mathcal{B} are described informally as the instructions of Alice and Bob respectively. We will view \mathcal{F} as being implemented by a quantum circuit acting on $\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}$ together with ancillas \mathcal{A}_a and \mathcal{B}_a initialized in state $|0\rangle$ (we shall explain below how to test that ancillas are really in state $|0\rangle$). At the end of the circuit, some of these wires become part of the outputs \mathcal{A}_{out} and \mathcal{B}_{out} , and the rest are part of the environment that will remain encrypted (\mathcal{A}_e and \mathcal{B}_e). Let $G_1, G_2, \dots, G_{\ell(n)}$ be an enumeration of all gates of the circuit for \mathcal{F} where $\ell(n)$ is polynomial in n , and G_i is executed before G_{i+1} . This protocol calls a number of subprotocols partially described next and in details in [5].

Protocol $\hat{\Pi}_{\mathcal{F}}^{\mathcal{G}'}$ for the evaluation of \mathcal{F} upon joint input $\rho_{\text{in}} \in D(\mathcal{R} \otimes \mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}})$:

1. Alice and Bob **run Initialization**,
2. **For** $i = 1 \dots \ell(n)$:
 - **If** G_i is a one-bit Clifford gate applied to wire w **then** Alice and Bob **call** `OneQubitClifford($G_i, \mathfrak{s}(w)$)`,
 - **If** G_i is a CNOT-gate applied to control wire w_c and w_t **then** Alice and Bob **call** `CNOT($\mathfrak{s}(w_c), \mathfrak{s}(w_t)$)`,
 - **If** G_i is an R-gate applied to wire w **then** Alice and Bob **call** `RGate($\mathfrak{s}(w)$)`,
3. Alice and Bob **call** `TestAllSwaddlings`,
4. Alice and Bob **call** `OpenAllSwaddlings`,
5. Alice and Bob decrypt the swaddlings for all wires in \mathcal{A}_{out} and \mathcal{B}_{out} using the keys received by the other party together with their own.

5.2 Subprotocols

The **Initialization** subprotocol prepares all swaddlings required during the evaluation of the circuit. In addition to swaddling all wires holding the qubits upon which the circuit acts, some ancillary states also have to be swaddled: one wire in the magic state and one wire in state $|0\rangle$ per R-gate in the circuit.

Initialization:

1. **For** each R-gate being applied to a wire in \mathcal{A} , Alice adds one additional wire $m_{a,i}$ initialized in state $|M\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ to \mathcal{A} .
2. **For** each R-gate being applied to a wire in \mathcal{B} , Bob adds an additional wire $m_{b,i}$ initialized in state $|M\rangle$ to \mathcal{B} .
3. **For** each R-gate being applied to a wire in \mathcal{B} , Alice adds an additional wire $c_{a,i}$ initialized in state $|0\rangle$ to \mathcal{A} .
4. **For** each R-gate being applied to a wire in \mathcal{A} , Bob adds an additional wire $c_{b,i}$ initialized in state $|0\rangle$ to \mathcal{B} .
5. Alice **calls** `Swaddle(\mathcal{A})`.
6. Bob **calls** `Swaddle(\mathcal{B})`.
7. **For** every wire $a \in \mathcal{A}_a$ or added in step 3 above, Alice **calls** `VerifyAncilla($\mathfrak{s}(a)$)`.
8. **For** every wire $a \in \mathcal{B}_a$ or added in step 4 above, Bob **calls** `VerifyAncilla($\mathfrak{s}(a)$)`.
9. **For** each magic wire $m \in \mathcal{A}$, Alice **calls** `DistillMagic($\mathfrak{s}(m)$)`.
10. **For** each magic wire $m \in \mathcal{B}$, Bob **calls** `DistillMagic($\mathfrak{s}(m)$)`.

Subprotocols `VerifyAncilla($\mathfrak{s}(a)$)` and `DistillMagic($\mathfrak{s}(m)$)` are described in [5]. The following subprotocols evaluates any one-qubit Clifford, the CNOT-gate, respectively the R. The idea behind the one-qubit protocol is to use TPC to compute the gate

through the authentication. Of course, the TPC needed depends upon the gate. The idea in the CNOT-subprotocol is to take the two swaddled qubits involved and turn them into one big swaddling. Then, performing a CNOT on them is no different from performing any other Clifford gate on the whole block. We then separate the big swaddling back into its components.

OneQubitClifford($C, \mathfrak{s}(w)$) with $w \in \mathcal{A}$:

1. Alice and Bob perform a TPC whose outcome tells Alice a randomly chosen Clifford gate which performs the gate, and changes the QAS key.
2. Alice performs the gate on $\mathfrak{s}(w)$.

CNOT($\mathfrak{s}(w_c), \mathfrak{s}(w_t)$) with wire $w_c \in \mathcal{A}$ as control and w_t as target:

1. Bob **sends** $\mathfrak{s}(w_t)$ to Alice if Bob holds it, in which case Alice **calls** $\text{TestDummies}(\mathfrak{s}(w_t))$.
2. Alice performs a randomly selected Clifford C on $4n + 2$ qubits jointly on $\mathfrak{s}(w_c)$ and $\mathfrak{s}(w_t)$.
3. Alice **sends** $\mathfrak{s}(w_c)$ and $\mathfrak{s}(w_t)$ to Bob; Bob **calls** TestDummies jointly on them.
4. Alice and Bob perform a TPC whose outcome tells Bob to perform a Clifford unitary $C' = (K'_c \otimes K'_t)(\text{CNOT})K^\dagger$ where K is the key of the swaddling at this point, and K'_c and K'_t are randomly-chosen Cliffords that become the new key.
5. Bob **sends** $\mathfrak{s}(w_c)$ (and $\mathfrak{s}(w_t)$ if she held this one too) to Alice. Alice **calls** TestDummies on them.

RGate($\mathfrak{s}(w)$) with $w \in \mathcal{A}$:

1. Alice performs a swaddled CNOT with \mathfrak{m}_w as a control, and w as target.
2. Alice **sends** $\mathfrak{s}(w)$ to Bob. Bob **calls** $\text{TestSwaddling}(\mathfrak{s}(w))$ on it.
3. Alice **calls** $\text{Measure}(\mathfrak{s}(w))$.
4. Alice and Bob perform a TPC whose result is a Clifford which, if both measurement results were zero, updates the key, and if both measurement results were one, performs a swaddled $e^{i\pi/4}XP^\dagger$ and then updates the key. If the measurement results differ, then they abort.
5. Alice relabels $\mathfrak{m}_{a,i}$ to w .

The R-protocol performs the R-gate using gate teleportation[9,8] via the magic state is very similar to the fault-tolerant version of the R-gate introduced in [7,14]. To perform an R-gate on a wire w via gate teleportation, we would first perform a CNOT from the magic state to w , and then measure w in the computational basis. If the answer is 0, we do nothing, and if it is 1, we perform $e^{i\pi/4}XP^\dagger$ on the former magic state, that we then rename w . $\text{Measure}(\mathfrak{s}(w))$ is described in [5].

5.3 Security of the Subprotocols

We will now show that the protocol described in the previous section has the following property: any adversary which deviates significantly from the protocol will be caught cheating with high probability. The general strategy will be as follows. For the initialization, any adversary will be forced to input something into the protocol, and will end up with some state that is properly swaddled. Then, for every other protocol step, we will assume that at the beginning, the inputs are properly swaddled, and will aim to show

that after the protocol step is done, we are once again left with a correct swaddling of the data, to which the correct operation has been applied. Furthermore, at every step, any deviation from the protocol will be essentially equivalent to an attack on the authentication scheme, which means that an adversary's chances of succeeding in changing the state without getting caught will be negligible in the number of dummy wires per qubit.

5.4 Some Additional Definitions

Before we start, we will find it convenient to introduce some additional notation. From now on, ρ_0 will consist of ρ_{in} augmented with all additional qubits introduced in the Initialization phase: the additional ancillas, and the dummy wires Alice and Bob use for swaddling. Furthermore, we will denote by \mathcal{K}_a and \mathcal{K}_b systems which represent Alice's and Bob's current key, respectively. These should be thought of as being part of the inner state of the TPC ideal functionality, and therefore cannot be changed at will. \mathcal{A} represents all other systems at Alice, \mathcal{B} represents all systems in Bob's possession, and \mathcal{R} is a system that includes everything else and ensures that the total state is pure.

Furthermore, in the sequel, we will call a *step* an execution of any of the subprotocols listed above; each of these steps consists of multiple *turns* in the sense of Section 2.3: the state at turn i consists of the state before the i th use of an oracle in the protocol (either a communication oracle or a classical computation oracle).

We will denote by $C_{a,s}$ the operation that encodes Alice's qubits according to the keys stored in the TPC ideal functionalities at turn s in the protocol, and likewise for Bob's encoding operation $C_{b,s}$. See [5] for more precise definitions. Furthermore, we will denote by $[\tilde{\mathcal{A}} \circledast \mathcal{B} \wedge \overline{\text{E-ABORT}}]_s^{\mathcal{O}'} \left(\rho_{\text{in}}^{\mathcal{A}_{\text{in}} \mathcal{B}_{\text{in}} \mathcal{R}} \right)$ the global state of the protocol at turn s conditioned on the fact that the protocol doesn't abort before `TestAllSwaddlings` is completed. Note that this state is not normalized; its trace corresponds to the probability of not aborting before the end.

Definition 5.1 (Forcing). We will say that the protocol is $\mathcal{G}^{\mathcal{AB}}$ -forcing for $\tilde{\mathcal{A}}$ at turn s with initial operation $\tilde{\mathcal{E}}_0^{\mathcal{A}}$ if there exists a final operation $\tilde{\mathcal{E}}^{\mathcal{A}}$ such that:

$$\Delta \left([\tilde{\mathcal{A}} \circledast \mathcal{B} \wedge \overline{\text{E-ABORT}}]_s^{\mathcal{O}'} \left(\rho_{\text{in}}^{\mathcal{A}_{\text{in}} \mathcal{B}_{\text{in}} \mathcal{R}} \right), \tilde{\mathcal{E}} \circ C_{b,s} \circ \mathcal{G} \circ \tilde{\mathcal{E}}_0 \left(\rho_{\text{in}}^{\mathcal{A}_{\text{in}} \mathcal{B}_{\text{in}} \mathcal{R}} \right) \right) \leq \text{negl}(n) ,$$

where $\tilde{\mathcal{E}}_0$ is a completely positive, trace non-increasing map that acts only on qubits in Alice's possession at the beginning of the protocol: her own input qubits, the dummies she inputs in the swaddling, and the various ancillas that she adds.

In other words, if the protocol is \mathcal{G} -forcing for Alice at some turn s , then, if the protocol doesn't abort early, regardless of what she tries to do, it will be essentially equivalent to changing her input (using the initial operation $\tilde{\mathcal{E}}_0$), executing the operation \mathcal{G} (which will turn out to be the circuit that is supposed to be executed up to turn s), swaddling the result, and then doing an arbitrary operation on her share alone, represented by $\tilde{\mathcal{E}}$.

Definition 5.2. We say that a subprotocol is \mathcal{G} -forcing if the protocol is $\mathcal{G} \circ \mathcal{G}_0$ -forcing at the end of the subprotocol given that it was \mathcal{G}_0 -forcing at the beginning.

Definition 5.3 (Hiding). We will say that the protocol is hiding for $\tilde{\mathcal{A}}$ at turn s , if, for all input states $\rho_{\text{in}}^{\mathcal{A}_{\text{in}} \mathcal{B}_{\text{in}} \mathcal{R}_{\text{in}}}$, we have that

$$\Delta \left(\text{tr}_{\mathcal{B}} \left([\tilde{\mathcal{A}} * \mathcal{B}]_s^{\mathcal{O}'} (\rho_{\text{in}}^{\mathcal{A}_{\text{in}} \mathcal{B}_{\text{in}} \mathcal{R}_{\text{in}}}) \right), \text{tr}_{\mathcal{B}} \left([\tilde{\mathcal{A}} * \mathcal{B}]_s^{\mathcal{O}'} (\rho_{\text{in}}^{\mathcal{A}_{\text{in}} \mathcal{R}_{\text{in}}} \otimes |0\rangle\langle 0|^{\mathcal{B}_{\text{in}}}) \right) \right) \leq \text{negl}(n) .$$

Hence, the protocol is hiding if the state seen by a dishonest Alice is independent of Bob's input. Note that this in particular implies that $\tilde{\mathcal{A}}$'s action at turn s is necessarily independent of Bob's input.

5.5 Proving Hidingness and Forcingness

To construct the simulator needed to prove security, we will need to show two things. First, we will have to show that before the keys are revealed, the cheater's internal state can be produced by running the protocol internally with a dummy input from the honest party; this will follow from the fact that the Clifford QAS is a secure encryption scheme. Second, we will have to show that after the keys are revealed, the correct circuit has been applied. These two properties correspond to the “hiding” and “forcing” properties defined in the previous section, and proving these for our protocol will be the focus of this section. The fact that the protocol is hiding simply follows from the security of the Clifford QAS as an encryption scheme; we state this as a lemma below. To prove forcingness, the usual trick will be to assume that we pass every call to `TestDummies` in the protocol (since we only need to look at the no-early-abort case) and use the security definition of the Clifford QAS (Definition 2.2) to show that the dishonest party's attack can be represented by a completely positive, trace non-increasing map \mathcal{U}^{acc} that acts only on his/her other systems (i.e. the ones that were not involved in the `TestDummies`). If the adversary decides to try to break the QAS at this step, this \mathcal{U}^{acc} will simply decrease the trace to reflect the probability of abortion. We prove the forcingness of the various subprotocols in [5], and simply summarize the end result as Lemma 5.5 below.

Lemma 5.4. For every turn before `OpenAllSwaddlings`, the protocol is hiding for every adversary $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$.

Proof. During this phase of the protocol, all $\tilde{\mathcal{A}}$ ever gets from Bob (and $\tilde{\mathcal{B}}$ from Alice) is encrypted in a QAS, whose key is managed by the classical two-party computations. Hence, this follows directly from the security of the Clifford QAS (see Section 2.2, or the full version [5] for more details).

Lemma 5.5. All subprotocols are \mathcal{G} -forcing for any $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$, where \mathcal{G} is the operation performed by the subprotocol.

6 Proving Active Security

Here is the rough description of $\tilde{\mathcal{C}}$'s operations for the start of the simulation. These steps allow to simulate any execution aborting in the real world before the adversary $\tilde{\mathcal{A}}$ receives back all her swaddlings from \mathcal{B} in `TestAllSwaddlings`. Let s^* be the turn in $\tilde{\Pi}_{\mathcal{F}}^{\mathcal{O}'}$ at which this transmission is received by $\tilde{\mathcal{A}}$. The simulator $\tilde{\mathcal{C}}$ runs $\tilde{\mathcal{A}}$ as a subroutine using an internal copy of \mathcal{B} 's instructions run upon a (or any) dummy input state $|0\rangle^{\mathcal{B}_{\text{in}}}$. We denote by \mathcal{B}^* the simulated \mathcal{B} on a dummy input run internally in $\tilde{\mathcal{C}}$.

We define E-ABORT_s as the event consisting in an execution between two parties aborting at turn $s < s^*$. Let $[\tilde{\mathcal{C}} \circledast \mathcal{D} \wedge \text{E-ABORT}_s]^{\mathcal{F}}(\rho_{\text{in}})$ and $[\tilde{\mathcal{A}} \circledast \mathcal{B} \wedge \text{E-ABORT}_s]^{\mathcal{O}'}(\rho_{\text{in}})$ be denoting the joint state of an execution that aborts at turn s upon joint input state ρ_{in} between $\tilde{\mathcal{C}}$ and \mathcal{D} (in the ideal world) and between $\tilde{\mathcal{A}}$ and \mathcal{B} (in the real world) respectively. States $[\tilde{\mathcal{C}} \circledast \mathcal{D} \wedge \text{E-ABORT}_s]^{\mathcal{F}}(\rho_{\text{in}})$ and $[\tilde{\mathcal{A}} \circledast \mathcal{B} \wedge \text{E-ABORT}_s]^{\mathcal{O}'}(\rho_{\text{in}})$ are not normalized, $\text{tr}([\tilde{\mathcal{C}} \circledast \mathcal{D} \wedge \text{E-ABORT}_s]^{\mathcal{F}}(\rho_{\text{in}}))$ and $\text{tr}([\tilde{\mathcal{A}} \circledast \mathcal{B} \wedge \text{E-ABORT}_s]^{\mathcal{O}'}(\rho_{\text{in}}))$ are the probabilities that $[\tilde{\mathcal{C}} \circledast \mathcal{D}]^{\mathcal{F}}(\rho_{\text{in}})$ and $[\tilde{\mathcal{A}} \circledast \mathcal{B}]^{\mathcal{O}'}(\rho_{\text{in}})$ aborts at step s respectively. The proof of next lemma can be found in [5] and follows easily from the hiding property of the protocol expressed in Lemma 5.4.

Lemma 6.1 (Early abort). *Let $\tilde{\mathcal{A}}$ be an adversary in hybrid protocol $\hat{\Pi}_{\mathcal{F}}^{\mathcal{O}'} = (\mathcal{A}, \mathcal{B}, \mathcal{O}', m)$ for $\mathcal{F} : \text{L}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \rightarrow \text{L}(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$. Let s^* be the turn in $\hat{\Pi}_{\mathcal{F}}^{\mathcal{O}'}$ at which \mathcal{B} returns all of $\tilde{\mathcal{A}}$'s swaddlings in TestAllSwaddlings and let $0 \leq s < s^*$. Then, there exists an adversary $\tilde{\mathcal{C}}$ in $\Gamma_{\mathcal{F}}$ (polysize in the size of $\tilde{\mathcal{A}}$) such that for any $\rho_{\text{in}} \in \text{D}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}} \otimes \mathcal{R})$,*

$$\Delta \left([\tilde{\mathcal{C}} \circledast \mathcal{D} \wedge \text{E-ABORT}_s]^{\mathcal{F}}(\rho_{\text{in}}), [\tilde{\mathcal{A}} \circledast \mathcal{B} \wedge \text{E-ABORT}_s]^{\mathcal{O}'}(\rho_{\text{in}}) \right) \leq \text{negl}(n) .$$

By symmetry, the same is also true with respect to adversaries $\tilde{\mathcal{B}}$ in $\hat{\Pi}_{\mathcal{F}}^{\mathcal{O}'}$ and $\tilde{\mathcal{D}}$ in $\Gamma_{\mathcal{F}}$.

It remains to simulate the execution from turn s^* until the end. In the simulated world, if $\tilde{\mathcal{C}}$ reaches turn s^* when $\tilde{\mathcal{A}}$ and \mathcal{B}^* are interacting then the output state $\mathcal{F}(\rho_{\text{in}}^{\mathcal{A}_{\text{in}} \mathcal{R}} \otimes |0\rangle\langle 0|)$ can be recovered. The reason being that at turn s^* , all swaddlings have been tested by \mathcal{B}^* in TestAllSwaddlings . $\tilde{\mathcal{C}}$ can get the output state since it knows all keys allowing to decrypt all logical wires, and all these wires have not been tampered with. The simulation then works along the same lines than in [4]. At turn s^* , $\tilde{\mathcal{C}}$ is simulating \mathcal{B}^* 's quantum transmission of all swaddlings belonging to $\tilde{\mathcal{A}}$ back to $\tilde{\mathcal{A}}$. These swaddlings have been successfully tested by \mathcal{B}^* in TestAllSwaddlings . $\tilde{\mathcal{C}}$ intercepts all these swaddlings and decrypts them together with all swaddlings held by \mathcal{B}^* . $\tilde{\mathcal{C}}$ then recovers the output state. $\tilde{\mathcal{C}}$ undoes the quantum operation \mathcal{F} in order to recover $\tilde{\mathcal{A}}$'s effective input state before querying \mathcal{F}_1 with the effective input state. $\tilde{\mathcal{C}}$ swaddles back the answer to the query using the same keys. The swaddlings are finally sent to $\tilde{\mathcal{A}}$ before resuming the interaction between $\tilde{\mathcal{A}}$ and \mathcal{B}^* . Two things can happen in OpenAllSwaddlings : 1) $\tilde{\mathcal{A}}$'s behavior makes the execution **abort**, and 2) $\tilde{\mathcal{A}}$ and \mathcal{B} reach the end of the execution in normal conditions. In the first case, $\tilde{\mathcal{A}}$ may even get the output state of the computation while preventing \mathcal{B} from recovering his own. When the output state is available to $\tilde{\mathcal{A}}$, $\tilde{\mathcal{C}}$ will need to call \mathcal{F}_1 (with $\tilde{\mathcal{A}}$'s effective input) and \mathcal{F}_2 where the later call is without fairness (i.e., $f = 0$) when $\tilde{\mathcal{A}}$ prevents \mathcal{B} from decrypting his output logical wires.

Let $\overline{\text{E-ABORT}}$ be the event of not having E-ABORT_s at any turn $s < s^*$. Next lemma establishes the active security when no early aborting occurs. The proof can be found in [5]. It follows from the forcingness of the protocol established in Lemma 5.5.

Lemma 6.2 (No early abort). *For any quantum adversary $\tilde{\mathcal{A}}$ in hybrid protocol $\hat{\Pi}_{\mathcal{F}}^{\mathcal{O}'} = (\mathcal{A}, \mathcal{B}, \mathcal{O}', m)$ for $\mathcal{F} : \text{L}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \rightarrow \text{L}(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$, there exists an adversary $\tilde{\mathcal{C}}$ in $\Gamma_{\mathcal{F}}$ (polysize in the size of $\tilde{\mathcal{A}}$ and \mathcal{B}) such that for any $\rho_{\text{in}} \in \text{D}(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}} \otimes \mathcal{R})$,*

$$\Delta \left([\tilde{\mathcal{C}} \otimes \mathcal{D} \wedge \overline{\text{E-ABORT}}]^{\mathcal{F}}(\rho_{\text{in}}), [\tilde{\mathcal{A}} \otimes \mathcal{B} \wedge \overline{\text{E-ABORT}}]^{\mathcal{O}'}(\rho_{\text{in}}) \right) \leq \text{negl}(n) .$$

By symmetry, the same is also true with respect to adversaries $\tilde{\mathcal{B}}$ in $\hat{\Pi}_{\mathcal{F}}^{\mathcal{O}'}$ and $\tilde{\mathcal{D}}$ in $\Gamma_{\mathcal{F}}$.

Since Lemmas 6.1 and 6.2 together show that a simulator succeeds in reproducing any real execution, we conclude the active security of the protocol:

Theorem 6.3 (Active security). *For any polynomial-time quantum operation $\mathcal{F} : L(\mathcal{A}_{\text{in}} \otimes \mathcal{B}_{\text{in}}) \rightarrow L(\mathcal{A}_{\text{out}} \otimes \mathcal{B}_{\text{out}})$, the two-party hybrid protocol $\hat{\Pi}_{\mathcal{F}}^{\mathcal{O}'}$ is statistically active secure without fairness for Bob.*

References

1. Aharonov, D., Ben-Or, M., Eban, E.: Interactive proofs for quantum computations. In: Proceedings of Innovations in Computer Science (2008), <http://arxiv.org/abs/0810.5375>
2. Barnum, H., Crépeau, C., Gottesman, D., Smith, A., Tapp, A.: Authentication of quantum messages. In: 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 449–458 (2002)
3. Bravyi, S., Kitaev, A.: Universal quantum computation with ideal clifford gates and noisy ancillas. Physical Review A 71, 022316 (2005), quant-ph/0403025
4. Dupuis, F., Nielsen, J.B., Salvail, L.: Secure Two-Party Quantum Evaluation of Unitaries against Specious Adversaries. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 685–706. Springer, Heidelberg (2010)
5. Dupuis, F., Nielsen, J.B., Salvail, L.: Actively secure two-party evaluation of any quantum operation. Cryptology ePrint Archive, record 2012/304 (2012), <http://eprint.iacr.org/>
6. Gottesman, D.: Stabilizer codes and quantum error correction. PhD thesis, California Institute of Technology (1997)
7. Gottesman, D.: An introduction to quantum error correction and fault-tolerant quantum computation. In: Lomonaco Jr., S.J. (ed.) Quantum Information Science and Its Contributions to Mathematics. Proceedings of Symposia in Applied Mathematics, vol. 68, pp. 13–60 (April 2010), <http://arxiv.org/abs/0904.2557>
8. Gottesman, D., Chuang, I.L.: Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. Nature 402, 390–393 (1999)
9. Gottesman, D., Chuang, I.L.: Quantum teleportation is a universal computational primitive (August 1999), <http://arxiv.org/abs/quant-ph/9908010>
10. Gutoski, G., Watrous, J.: Toward a general theory of quantum games. In: 39th Annual ACM Symposium on Theory of Computing (STOC), pp. 565–574 (2007)
11. Hallgren, S., Smith, A., Song, F.: Classical Cryptographic Protocols in a Quantum World. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 411–428. Springer, Heidelberg (2011)
12. Lunemann, C., Nielsen, J.B.: Fully Simulatable Quantum-Secure Coin-Flipping and Applications. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 21–40. Springer, Heidelberg (2011)
13. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 84–93 (2005)
14. Shor, P.W.: Fault-tolerant quantum computation. In: 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 56–65 (1996)

On the Impossibility of Constructing Efficient Key Encapsulation and Programmable Hash Functions in Prime Order Groups

Goichiro Hanaoka, Takahiro Matsuda, and Jacob C.N. Schuldt

Research Institute for Secure Systems,
National Institute of Advanced Industrial Science and Technology
`{hanaoka-goichiro,t-matsuda,jacob.schuldt}@aist.go.jp`

Abstract. In this paper, we discuss the (im)possibility of constructing chosen ciphertext secure (CCA secure) key encapsulation mechanisms (KEMs) with low ciphertext overhead. More specifically, we rule out the existence of algebraic black-box reductions from the (bounded) CCA security of a natural class of KEMs to any non-interactive problem. The class of KEMs captures the structure of the currently most efficient KEMs defined in standard prime order groups, but restricts an encapsulation to consist of a single group element and a string. This result suggests that we cannot rely on existing techniques to construct a CCA secure KEM in standard prime order groups with a ciphertext overhead lower than two group elements. Furthermore, we show how the properties of an (algebraic) programmable hash function can be used to construct a simple, efficient and CCA secure KEM based on the hardness of the decisional Diffie-Hellman problem with a ciphertext overhead of just a single group element. Since this KEM construction is covered by the above mentioned impossibility result, this enables us to derive a lower bound on the hash key size of an algebraic programmable hash function, and rule out the existence of algebraic (poly, n) -programmable hash functions in prime order groups for any integer n . The latter result answers an open question posed by Hofheinz and Kiltz (CRYPTO'08) in the case of algebraic programmable hash functions in prime order groups.

1 Introduction

The development of efficient and secure public key encryption has long been a central research area in cryptography, and in particular, achieving security against chosen ciphertext attacks (CCA security) while maintaining practical efficiency has been the focus of many papers in the literature. One of the most commonly used performance measures for public key encryption schemes, and the measure that we are going to focus on in this paper, is ciphertext overhead which expresses the additional cost in terms of storage and bandwidth when operating with encrypted data as opposed to unencrypted data.

The currently most efficient encryption schemes are based on hybrid encryption [10]. In this approach, a public key component, referred to as a key encapsulation mechanism (KEM), is used to encrypt a random session key, and

the message is then encrypted using a symmetric cipher which is referred to as a data encapsulation mechanism (DEM). The ciphertext overhead of this type of construction is dominated by the KEM. Specifically, if the KEM achieves CCA security, a redundancy-free DEM can be used [27], since only one-time CCA security is required of the DEM to obtain a CCA secure hybrid encryption scheme. If the KEM achieves the slightly weaker notion of constrained CCA security [20], an authenticated DEM is required, which will introduce a small additional overhead corresponding to a message authentication code (MAC).¹

The currently most efficient (constrained) CCA secure KEMs, which are provably secure in the standard model, are defined in prime order groups, and have a ciphertext overhead of at least two group elements, e.g. [10,25,4,24,20,15]. An overview of these is given in Table 1. Note that when considering constrained CCA secure KEMs, the additional ciphertext overhead of a MAC must be taken into account.² The KEM by Cramer et al. [9], which achieves a ciphertext overhead of a single group element, can only be shown q -bounded CCA secure (for a predetermined number of decryption queries q), and hence will not lead to a fully CCA secure encryption scheme if combined with a DEM.

More Efficient Schemes? Given the existing KEMs, it is natural to ask: *Is it possible to construct a CCA secure KEM with a ciphertext overhead of less than two group elements?* Note that, besides being defined in prime order groups, the KEMs in Table 1 share some structural properties. More specifically, all of the KEMs include a random group element as part of the ciphertext which will be used to derive the session-key in the decapsulation. The remaining element(s) (except in KD [25], but see footnote³) are used to decide whether the ciphertext is accepted as “valid”, but does not otherwise contribute to the computation of the decapsulated key.

Given this, one might consider implementing a more space-efficient validity check, using MACs and hash functions, as a potential strategy for reducing the ciphertext overhead in the above mentioned schemes. To illustrate this approach, consider a KEM by Cramer and Shoup [10, Sect. 9.3]. In this KEM, a public key is of the form $pk = (g, X_1, X_2 = g^{x_1}X_1^{x_2}, X_3 = g^{y_1}X_1^{y_2}, X_4 = g^z)$, where g, X_1 are group generators, and the private key is $sk = (x_1, x_2, y_1, y_2, z)$. A ciphertext consists of $(c_1 = g^r, c_2 = X_1^r, c_3 = X_2^rX_3^{r\alpha})$ and the corresponding session-key is $K = X_4^r$, where r is picked at random from \mathbb{Z}_p , p is the order of

¹ Alternatively, the KEM can be generically converted to a CCA secure KEM [2], but this will likewise introduce an additional overhead of a MAC.

² For KEMs defined in prime order groups, each group element in the ciphertext overhead will contribute with at least 2λ bits for a security level of λ bits, since the order p of the group will have to satisfy $p > 2^{2\lambda}$ to prevent generic attacks against the underlying assumptions of the security of the KEMs. On the other hand, a MAC contributes with only λ bits.

³ While the KEM in [25] makes use of “implicit rejection” and does not explicitly check the validity of a ciphertext, it is relatively straightforward to make the scheme use explicit rejection through a validity check. This KEM will be IND-CCCA secure under the DDH assumption, and will fit the description mentioned here.

Table 1. The currently most efficient KEMs in terms of ciphertext overhead. The scheme Kiltz [24][†] is identical to Kiltz [24], except that no hash function is used to derive the session-key (see Sect. 4.2 of [24]). In the table, CCCA denotes constrained CCA [20] and q -CCA denotes q -bounded CCA [9]. Furthermore, DDH denotes the decisional Diffie-Hellman assumption, CDH the computational Diffie-Hellman assumption, GDH the gap Diffie-Hellman assumption, GHHD the gap hashed Diffie-Hellman assumption, and DBDH the decisional bilinear Diffie-Hellman assumption.

Scheme	Security	Hardness assumption	Ciphertext overhead
CS [10]	IND-CCA	DDH	$3 \mathbb{G} $
HaKu [15, Sect. 5]	IND-CCA	CDH	$3 \mathbb{G} $
HaKu [15, Sect. 4.1]	OW-CCA	CDH	$3 \mathbb{G} $
KD [25]	IND-CCCA	DDH	$2 \mathbb{G} $
HoKi [20]	IND-CCCA	DDH	$2 \mathbb{G} $
HaKu [15, Sect. 6]	IND-CCCA	DDH	$2 \mathbb{G} $
Kiltz [24]	IND-CCA	GHHD	$2 \mathbb{G} $
Kiltz [24] [†]	OW-CCA	GDH	$2 \mathbb{G} $
BMW [4]	IND-CCA	DBDH	$2 \mathbb{G} $
CHH+ [9]	IND- q -CCA	DDH	$1 \mathbb{G} $

the group, $\alpha = H(c_1, c_2)$, and H is a target collision resistant hash function. The decapsulation checks if $c_1^{x_1+y_1\alpha} c_2^{x_2+y_2\alpha} = c_3$, and outputs the session-key $K = c_1^z$ if this is the case. Otherwise, the ciphertext is rejected.

To reduce the ciphertext overhead, we might consider a slightly modified scheme in which the validity check is performed on the hash of the group elements instead of on the group elements themselves i.e. a ciphertext is of the form $(c_1 = g^r, c_2 = X_1^r, c_3' = H(X_2^r X_3^{r\alpha}))$, and the validity check is implemented as $H(c_1^{x_1+y_1\alpha} c_2^{x_2+y_2\alpha}) = c_3'$. Somewhat surprisingly, this leads to a CCA secure scheme as noted in [11]. This reduces the ciphertext overhead to match that of the other KEMs defined in standard prime order groups and based on non-interactive assumptions, when taking into account the additional overhead of a MAC required by these schemes e.g. Kurosawa-Desmedt [25], Hofheinz-Kiltz [20] and Hanaoka-Kurosawa [15].

A similar approach can be used to reduce the ciphertext overhead of the schemes Hofheinz-Kiltz [20], Hanaoka-Kurosawa [15], and Kurosawa-Desmedt [25] with explicit rejection. This yields KEMs with a ciphertext overhead of just a single group element and a hash value, which is lower than the currently most efficient schemes. However, unlike the modified version of the Cramer-Shoup KEM, the security proofs do not immediately extend to the modified KEMs. Hence, it is not obvious what level of security these schemes provide.

1.1 Our Contribution

In this paper, we discuss the impossibility of proving CCA security of KEMs in standard prime order groups with low ciphertext overhead. More specifically, as

our main result, we show that there is no algebraic black-box reduction from the q -bounded one-way non-adaptive CCA security ($\text{OW-}n\text{-CCA1}$) of a class of KEMs in which the ciphertext consists only of a single (random) group element and a string, to the hardness of *any non-interactive problem* defined in the group, where n is the number of group elements in the public key of the KEM. Since the majority of standard model security reductions are algebraic black-box reductions, this result sheds light on the question regarding the minimal ciphertext overhead achievable while maintaining CCA security by ruling out a natural class of KEMs with similar structure to the currently most efficient KEMs defined in standard prime order groups. Furthermore, the result holds even for security against adversaries who are restricted to make a single parallel decryption query. Hence, we can additionally rule out the existence of algebraic black-box reductions from the non-malleability of the captured KEMs due to the implications shown in [18,26]. These results imply that the approach of minimizing ciphertext overhead in the above mentioned schemes [20,15,25], by compressing group elements using a target collision resistant hash function (or a similar primitive), will not yield CCA secure or non-malleable KEMs based on non-interactive assumptions. Additionally, since the DDH-based KEM by Cramer et al. [9] is contained in the KEM class, our results imply that this scheme cannot be shown fully CCA secure or non-malleable based on any non-interactive assumption.

Secondly, we show a simple construction of a CCA-secure KEM using programmable hash functions introduced by Hofheinz and Kiltz [21]. These hash functions capture the “programmability” achieved by a random oracle, and have been shown useful, for example, in the construction of short signatures in the standard model [21,19]. We show that an algebraic $(q, 1)$ -programmable hash function allows the construction of a q -bounded CCA ($\text{IND-}q\text{-CCA2}$) secure KEM based on the DDH assumption, with a ciphertext overhead of just a single group element (see Section 5.1 for the definition of a (α, β) -programmable hash function). Since this construction is covered by the above impossibility result, we can derive a lower bound on the level of programmability provided by a hash function with a given hash key size. Specifically, we show that a hash function with n group elements in the hash key cannot be $(\alpha, 1)$ -programmable for any $\alpha > n$. Furthermore, we rule out the existence of algebraic (poly, β) -programmable hash functions in prime order groups for any $\beta > 0$. This result answers an open question posed by Hofheinz and Kiltz [21] in the case of algebraic programmable hash functions in prime order groups. We note that all known constructions of programmable hash functions are algebraic [21,19], and that the properties of these hash functions suggest that this may be inherent [21, Sect. 1.5].

1.2 Used Techniques and Related Work

The type of impossibility results we show is commonly known as a *black-box separation* [23]. More specifically, a black-box reduction from the security of a cryptographic scheme to the hardness of a problem is an algorithm which, given (black-box) access to any successful adversary against the scheme, successfully

breaks any instance of the problem. A black-box separation result shows that such a black-box reduction cannot exist.

Two main lines of techniques have been used for showing black-box separations: oracle separations [23,28] and meta-reductions [3,8]. The former technique is typically used to show separations between primitives, e.g. [23,29,12,14,13], and is based on showing the existence of an oracle under which the primitive acting as a building block exists, but any instantiation of the “target” primitive is broken. The latter technique is somewhat more direct, and aims at showing that if there exists a reduction which, for example, reduces the security of a primitive to a computational assumption, then there exists a meta-reduction which uses the reduction as a black-box to break a (possibly different) computational assumption. Meta-reductions have successfully been used in e.g. [3,8,1]. For an overview of these definitions, techniques, and results, see [28,32].

While we are not considering a primitive-to-primitive reduction, we make use of a variant of the oracle separation technique. In particular, we show the existence of a *distribution* of oracles under which, on average over the choice of the oracle, the non-interactive problem remains hard, while the CCA security of any of the considered KEMs can be broken. We show that such a distribution of an oracle is sufficient to rule out a fully black-box reduction [28] from the security of a KEM in the considered class of KEMs, to the hardness of any non-interactive problem. Since almost all security reductions for cryptographic primitives are of this type, ruling out the existence of fully black-box reduction gives strong evidence that the currently used techniques are not sufficient to prove the security of the KEMs in question. Our proof techniques, especially our formal treatment of the distribution of oracles, might be of independent interest.

The type of (fully) black-box reductions we are going to consider are *algebraic* reductions [3,8,1]. Essentially, the algebraic property requires that the reduction only creates group elements by means of the group operation, and does not map arbitrary bit strings to group elements e.g. by applying a hash function to some string to obtain a group element. More specifically, for an algebraic algorithm, it is required that it is possible to compute the representation of a group element output by the algorithm in terms of the group elements which are given as input. For example, if an algebraic algorithm takes as input the group elements g_1, g_2 and outputs the element h , it should be possible to compute x_1, x_2 such that $g_1^{x_1} \cdot g_2^{x_2} = h$, given access to the randomness used by the algorithm. As argued in previous papers [3,1], considering algebraic reductions is not overly restrictive, and almost all known security reductions for CCA secure KEMs in the standard model are algebraic. In particular, the security reduction for the KEMs shown in Table 1 are all algebraic.

2 Preliminaries

In this paper, we use the following basic notations and terminology. \mathbb{N} denotes the set of all natural numbers, and if $n \in \mathbb{N}$ then $[n] = \{1, \dots, n\}$. “PPTA” denotes a *probabilistic polynomial time algorithm*. If S is a set, “ $x \leftarrow S$ ” denotes picking

x uniformly from S , and if \mathbb{D} is a probability distribution, then " $x \leftarrow \mathbb{D}$ " denotes choosing x according to \mathbb{D} . If \mathcal{A} is a probabilistic algorithm then $y \leftarrow \mathcal{A}(x; r)$ denotes that \mathcal{A} computes y as output by taking x as input and using r as randomness, and $\mathcal{A}^{\mathcal{O}}$ denotes that \mathcal{A} has access to the oracle \mathcal{O} . Unless otherwise stated, λ denotes the security parameter. A function $f(\lambda) : \mathbb{N} \rightarrow [0, 1]$ is said to be *negligible* (resp. *noticeable*) if for all positive polynomials $p(\lambda)$ and all sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$ (resp. $f(\lambda) \geq 1/p(\lambda)$). Let \mathbf{X} be a vector, then we denote by $\mathbf{X}[i]$ the i -th component of \mathbf{X} .

Group Description. In this paper, we consider non-interactive problems with respect to a family of prime-order groups $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ indexed by the security parameter λ . For convenience, we consider the following PTA, which we call a *group scheme*, with which we will define such problems. (When λ is easily inferred from the context, we will usually leave out the subscript λ and just write \mathbb{G} .)

Definition 1. A group scheme GS is a deterministic PTA which takes a security parameter 1^λ as input, and outputs a group instance Λ consisting of a description of a group \mathbb{G} , a prime p that corresponds to the order of the group \mathbb{G} , and a generator $g \in \mathbb{G}$. This process is denoted by $\Lambda = (g, p, \mathbb{G}) \leftarrow \text{GS}(1^\lambda)$.

Note that since GS is assumed to be deterministic, there is a one-to-one correspondence between λ and the group description Λ generated by $\text{GS}(1^\lambda)$.

Algebraic Algorithms. Intuitively, an algorithm \mathcal{R} is called *algebraic* if there exists a corresponding algorithm, called the *extractor*, such that for any group element output by \mathcal{R} , the extractor can compute the representation of the group element with respect to the group elements given to \mathcal{R} as input.

Formally, we adopt a similar approach to [1] and define the notion of algebraic algorithms as follows:

Definition 2. Let \mathcal{R} be a PPTA that takes $\Lambda = (g, p, \mathbb{G})$ (output by GS), a string $\text{aux} \in \{0, 1\}^*$, and group elements $\mathbf{X} \in \mathbb{G}^n$ for some $n \in \mathbb{N}$ as input, and outputs a group element $Y \in \mathbb{G}$ and a string $\text{ext} \in \{0, 1\}^*$. \mathcal{R} is called *algebraic* with respect to GS if there exists a PPTA \mathcal{E} receiving the same input as \mathcal{R} (including the same random coins r) such that for any $\Lambda \leftarrow \text{GS}(1^\lambda)$, all polynomial size \mathbf{X} , and any string $\text{aux} \in \{0, 1\}^*$, the following probability is negligible in λ :

$$\Pr[(Y, \text{ext}) \leftarrow \mathcal{R}(\Lambda, \mathbf{X}, \text{aux}; r); (\mathbf{y}, \text{ext}) \leftarrow \mathcal{E}(\Lambda, \mathbf{X}, \text{aux}, r) : Y \neq \mathbf{X}^{\mathbf{y}}]$$

where the probability is over the choice of r and the randomness used by \mathcal{E} , and $\mathbf{X}^{\mathbf{y}}$ is defined by $\prod_{i \in [| \mathbf{X} |]} \mathbf{X}[i]^{\mathbf{y}[i]}$.

Note that the definition of an algebraic algorithm does not exclude the possibility that the auxiliary information aux contains group elements of \mathbb{G} , but the representation of the output element Y computed by the extractor must be with respect to \mathbf{X} .

The above definition of algebraic algorithms can naturally be extended to algebraic “oracle” algorithms, which play an important role in our result. Specifically, let \mathcal{R} be an oracle PPTA that takes $\Lambda = (g, p, \mathbb{G})$ (output by $\text{GS}(1^\lambda)$),

a string $aux \in \{0, 1\}^*$, and group elements $\mathbf{X} \in \mathbb{G}^n$ for some $n \in \mathbb{N}$ as input, and outputs a group element $Y \in \mathbb{G}$ and a string $ext \in \{0, 1\}^*$. Furthermore, let $q = q(\lambda)$ be the number of queries made by \mathcal{R} to the oracle. We say that \mathcal{R} is an *algebraic oracle algorithm* if there exists an algebraic algorithm $\widehat{\mathcal{R}}$ (which we denote the *decomposition* of \mathcal{R}) such that executing $\mathcal{R}^O(\Lambda, \mathbf{X}, aux)$ is equivalent to performing the following sequence of computations: First set $\mathbf{X}_0 \leftarrow \mathbf{X}$ and $aux_0 \leftarrow aux$. Run $(\mathbf{Y}_1, (ext||st_1)) \leftarrow \widehat{\mathcal{R}}(\Lambda, \mathbf{X}_0, aux_0)$ and repeat

$$(\mathbf{X}'_i, aux'_i) \leftarrow \mathcal{O}(\Lambda, \mathbf{Y}_i, ext_i); \quad \mathbf{X}_{i+1} \leftarrow (\mathbf{X}_i || \mathbf{X}'_i); \quad aux_{i+1} \leftarrow (st_i || aux'_i); \\ (\mathbf{Y}_{i+1}, (ext_{i+1} || st_{i+1})) \leftarrow \widehat{\mathcal{R}}(\Lambda, \mathbf{X}_{i+1}, aux_{i+1})$$

for $i = 1, \dots, q$. The last vector \mathbf{Y}_{q+1} output by $\widehat{\mathcal{R}}$ is assumed to contain the single element Y .

Note that since the decomposition $\widehat{\mathcal{R}}$ of an algebraic oracle algorithm \mathcal{R} is defined as an algebraic algorithm, the representation of any group element output by \mathcal{R} or included in the oracle queries made by \mathcal{R} , can be calculated by appropriately using the extractor for $\widehat{\mathcal{R}}$. In this case, the representation is with respect to all group elements that are given as input to \mathcal{R} or returned in response to \mathcal{R} 's oracle queries.

The above definition can easily be extended to algorithms outputting multiple group elements. Note that we will regard any algorithm whose output does not contain any group elements, as an algorithm which outputs the “identity element” $1_{\mathbb{G}}$. Hence, this type of algorithm will also be considered to be algebraic.

Non-interactive Problems with Respect to a Prime Order Group. A non-interactive problem (NIP) P with respect to a group scheme GS consists of a tuple of algebraic PPTAs, (I, V, U) , that are defined as follows:

Instance Generator I: This algorithm takes a group description $\Lambda = (g, p, \mathbb{G})$ (output from $GS(1^\lambda)$) as input, and outputs a pair consisting of a problem instance and a witness (y, w) .

Verification Algorithm V: This algorithm takes a problem instance y , a string x , and a witness w as input (where (y, w) are output by $I(\Lambda)$), and outputs \top or \perp . We say that x is a valid solution to the problem instance y if $V(y, x, w) = \top$, and otherwise we say that x is an invalid solution.

Threshold Algorithm U: This algorithm takes a problem instance y as input, and outputs a string x .

For a NIP $P = (I, V, U)$ with respect to GS and an algorithm A , define the experiment $\text{Expt}_{GS, A}^P(\lambda)$ as:

$$(g, p, \mathbb{G}) \leftarrow GS(1^\lambda); \quad (y, w) \leftarrow I(\Lambda); \quad x \leftarrow A(1^\lambda, y); \quad \text{Return } (V(y, x, w) \stackrel{?}{=} \top)$$

Furthermore, define the threshold $\delta(\lambda) = \Pr[\text{Expt}_{GS, U}^P(\lambda) = 1]$. Then, for an algorithm A , we define the advantage of A in solving P by:

$$\text{Adv}_{GS, A}^P(\lambda) = \Pr[\text{Expt}_{GS, A}^P(\lambda) = 1] - \delta(\lambda).$$

Definition 3. Let P be a NIP with respect to a group scheme GS . We say that P is hard if, for any PPTA \mathcal{A} , there exists a negligible function $\mu(\cdot)$ such that $\text{Adv}_{\mathsf{GS}, \mathcal{A}}^{\mathsf{P}}(\lambda) \leq \mu(\lambda)$.

Intuitively, the algorithm U represents a trivial solution strategy for the problem P , and any successful algorithm is required to be better than this U . Typically, U always returns an invalid answer for “search” problems i.e. $\delta(\lambda) = 0$, and returns a random bit for “decision” problems i.e. $\delta(\lambda) = 1/2$.

The above definition of a non-interactive problem essentially captures all the non-interactive problems defined for prime order groups, which are used to prove the security of existing cryptographic primitives. Specifically, the definition includes the standard computational and decisional Diffie-Hellman problems as well as their q -type variants (q -SDH, q -ABDHE, etc.), and will even capture the CPA security of a KEM.

Key Encapsulation. Since in this paper we will only treat KEMs based on a group with prime order, for convenience we define a KEM with respect to a group scheme GS , and change the security experiments accordingly.

A KEM Γ with respect to a group scheme GS consists of the following three PPTAs (KG , Enc , Dec). KG is a key generation algorithm which takes a group description $\Lambda = (g, p, \mathbb{G})$ (output from $\mathsf{GS}(1^\lambda)$) as input, and outputs a public/secret key pair (pk, sk) ; Enc is an encapsulation algorithm which takes pk as input, and outputs a ciphertext c and a session-key $K \in \mathcal{K}$ (where \mathcal{K} is a session-key space specified by pk); and Dec is a decapsulation algorithm which takes sk and c as input, and outputs a session-key K or an error symbol \perp indicating that c is “invalid”. We require $\mathsf{Dec}(sk, c) = K$ for all (pk, sk) output by $\mathsf{KG}(\Lambda)$ and all (c, K) output by $\mathsf{Enc}(pk)$.

Typically, security notions for KEMs are expressed by the combination of a security goal (**GOAL**) and an adversary’s attack type (**ATK**). In this paper, we will consider *indistinguishability* (**IND**) and *one-wayness* (**OW**) as security goals **GOAL**, and *chosen plaintext attacks* (**CPA**), *non-adaptive chosen ciphertext attacks* (**CCA1**), *adaptive chosen ciphertext attacks* (**CCA2**), and their q -bounded analogues [9] (q -**CCA1** and q -**CCA2**) as an adversary’s attack types **ATK**. Due to space limitations, we will only define OW - q -**CCA1** security. See the full version for the remaining definitions.

We say that a KEM $\Gamma = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ with respect to GS is OW - q -**CCA1** secure if, for any PPTA \mathcal{A} , the following advantage function is negligible in λ :

$$\begin{aligned} \text{Adv}_{\mathsf{GS}, \mathcal{A}}^{\mathsf{OW}-q\text{-}\mathsf{CCA1}}(\lambda) &= \Pr[\Lambda \leftarrow \mathsf{GS}(1^\lambda); (pk, sk) \leftarrow \mathsf{KG}(\Lambda); \mathsf{st} \leftarrow \mathcal{A}_1^{\mathsf{Dec}(sk, \cdot)}(pk); \\ &\quad (c^*, K^*) \leftarrow \mathsf{Enc}(pk); K' \leftarrow \mathcal{A}_2(\mathsf{st}, c^*): K^* = K'] \end{aligned}$$

where the decapsulation oracle $\mathsf{Dec}(sk, \cdot)$ can only be queried q times.

Algebraic Black-Box Reductions. We will consider the following type of reductions from the security of a KEM to the hardness of a non-interactive problem.

Definition 4. Let GS be a group scheme, and let Γ be a KEM and P be a NIP with respect to GS . Furthermore, let GOAL-ATK be a security notion for a KEM. We say that there is an algebraic black-box reduction from the GOAL-ATK security of Γ to P if there exists an algebraic oracle PPTA \mathcal{R} with the following property: For any (possibly computationally unbounded) algorithm \mathcal{A} , if $\text{Adv}_{\Gamma, \mathcal{A}}^{\text{GOAL-ATK}}(\lambda)$ is non-negligible, then so is $\text{Adv}_{\text{GS}, \mathcal{R}, \mathcal{A}}^{\mathsf{P}}(\lambda)$.

We note that this type of reduction is categorized as a “fully” black-box reduction in the taxonomy by Reingold et al. [28]. In particular, the reduction algorithm is required to work universally for all successful (possibly inefficient) algorithms.

3 A Class of Simple and Space Efficient KEMs

The class of KEMs we consider essentially captures the structure of the existing KEMs defined in standard prime order groups like Cramer-Shoup [10], Kurosawa-Desmedt [25] (with explicit rejection), Hofheinz-Kiltz [20], Hanaoka-Kurosawa [15], and Cramer et al. [9], but requires the ciphertexts to consist of a single random group element and a string i.e. a ciphertext is required to be of the form $(g^r, f(pk, r))$ where $r \leftarrow \mathbb{Z}_p$, p is the order of the group, pk is the public key of the scheme, $\tilde{f} : \mathcal{PK} \times \mathbb{Z}_p \rightarrow \{0, 1\}^*$ is a scheme-dependent function, and \mathcal{PK} is the public key space. This captures the approach highlighted in the introduction of replacing the group elements used for validity checking in the above KEMs with the output of a hash function or similar primitive. However, we note that $\tilde{f}(pk, r)$ is not limited to be used in a potential validity check, but might also be used when deriving the session-key. The session-keys encapsulated by the ciphertexts are assumed to be group elements, but can be derived from all the information available in the encapsulation in any “algebraic” way. Note that this captures the key derivation in the above KEMs, and does not rule out the use of target collision resistant hash function, or the use of a Waters hash function [30] (and similar constructions).

We consider a class of KEMs $\mathcal{K}_{\text{GS}, n}$ defined with respect to a group scheme GS and a parameter $n \in \mathbb{N}$.

Definition 5. A KEM $\Gamma = (\text{KG}, \text{Enc}, \text{Dec})$ with respect to a group scheme GS belongs to $\mathcal{K}_{\text{GS}, n}$, where $n \in \mathbb{N}$, if it has the following properties:

1. The public key space \mathcal{PK} is $\{\Lambda\} \times \{0, 1\}^{\mu(\lambda)} \times \mathbb{G}^n$, where μ is a scheme-dependent function and $\Lambda = (g, p, \mathbb{G}) \leftarrow \text{GS}(1^\lambda)$ i.e. a public key pk returned by $\text{KG}(\Lambda)$ is of the form $pk = (\Lambda, \text{aux}, X_1, \dots, X_n)$.
2. A ciphertext $C \in \mathbb{G} \times \{0, 1\}^*$ and the corresponding session-key $K \in \mathbb{G}$ are of the form

$$C = (c, d) = (g^r, \tilde{f}(pk, r)) \quad \text{and} \quad K = g^{f_0(pk, r)} \prod_{i \in [n]} X_i^{f_i(pk, r)}$$

where $r \leftarrow \mathbb{Z}_p$, and $\tilde{f} : \mathcal{PK} \times \mathbb{Z}_p \rightarrow \{0, 1\}^*$ and $f_0, \dots, f_n : \mathcal{PK} \times \mathbb{Z}_p \rightarrow \mathbb{Z}_p$ are efficiently computable scheme-dependent functions.

3. For any $pk = (\Lambda, aux, X_1, \dots, X_n) \in \mathcal{PK}$, the session-key obtained by decapsulating a ciphertext $C = (c, d)$ generated by $\text{Enc}(pk)$ is of the form

$$K = g^{\psi_0(pk, C, y_1, \dots, y_n)} c^{\psi_1(pk, C, y_1, \dots, y_n)} \quad (1)$$

where $\psi_i(pk, C, y_1, \dots, y_n) = \psi_{i,0}(pk, C) + \sum_{j \in [n]} \psi_{i,j}(pk, C) \cdot y_j$ for $i \in \{0, 1\}$, $y_k = \log_g X_k$ for $k \in [n]$, and $\{\psi_{i,j} : \mathcal{PK} \times \mathbb{G} \times \{0, 1\}^* \rightarrow \mathbb{Z}_p\}_{i \in \{0, 1\}, j \in [n]}$ are efficiently computable scheme-dependent functions.

4. For any $pk = (\Lambda, aux, X_1, \dots, X_n) \in \mathcal{PK}$, the second component $d \in \{0, 1\}^*$ of a ciphertext $C = (c, d)$ generated by $\text{Enc}(pk)$ can be re-computed as follows: $d = \tilde{\psi}(pk, c, y_1, \dots, y_n)$, where $y_i = \log_g X_i$ for $i \in [n]$ and $\tilde{\psi} : \mathcal{PK} \times \mathbb{G} \times \mathbb{Z}_p^n \rightarrow \{0, 1\}^*$ is a scheme-dependent function.

We would like to note the following:

- The values y_1, \dots, y_n defined above might not correspond to the private key of the scheme, and the decapsulation might not be done as shown in (1), but it is required that any valid session-key K output by Dec satisfy equation (1).
- If a KEM in $\mathcal{K}_{GS,n}$ satisfies correctness, it must hold that

$$f_0(pk, r) + \sum_{i \in [n]} f_i(pk, r) \cdot y_i = \psi_0(pk, C, y_1, \dots, y_n) + r \cdot \psi_1(pk, C, y_1, \dots, y_n)$$

for any $r \in \mathbb{Z}_p$. Hence, the requirement that the functions ψ_0 and ψ_1 are linear functions in y_1, \dots, y_n is arguably a very mild restriction.

- That the component d of a ciphertext can be recomputed using $\tilde{\psi}$, is natural if d is used as a part of a validity check. Note, however, that no requirements are made regarding how a KEM in $\mathcal{K}_{GS,n}$ implements validity checking.
- That aux and $d = \tilde{f}(pk, r)$ are strings imply that the representation of a group element output by any algebraic algorithm taking a public key or a ciphertext as input, cannot depend on group elements derived from aux or d .
- There are no restrictions on the scheme-dependent functions $\tilde{f}, \{f_i\}_{i \in \{0, \dots, n\}}$, $\tilde{\psi}, \{\psi_{i,j}\}_{i \in \{0, 1\}, j \in [n]}$ (other than that they are efficiently computable), and these might use non-linear functions like cryptographic hash functions and MACs, which is not allowed in “structure-preserving” encryption [5].

4 Main Impossibility Result

The following theorem captures our main result.

Theorem 6. For any group scheme GS , for any KEM $\Gamma \in \mathcal{K}_{GS,n}$ where $n \in \mathbb{N}$, and for any NIP P with respect to GS , if P is hard, then there is no algebraic black-box reduction from the OW-n-CCA1 security of Γ to P.

We will show this theorem by using a variant of the oracle separation technique [23,28]. Specifically, the theorem follows from the following lemma. (In the lemma, $\text{Adv}_{GS, \mathcal{A}}^{\text{DL}}(\lambda)$ denotes the advantage of an algorithm \mathcal{A} in solving the discrete logarithm problem wrt. GS . See the full version for a formal definition.)

Lemma 7. Let \mathbf{GS} be a group scheme, $\Gamma \in \mathcal{K}_{\mathbf{GS},n}$ be a KEM where $n \in \mathbb{N}$, and \mathbf{P} be a NIP with respect to \mathbf{GS} . Furthermore, let $\mathbf{GOAL} \in \{\mathbf{OW}, \mathbf{IND}\}$ and $\mathbf{ATK} \in \{\mathbf{CPA}, q\text{-CCA1}, \mathbf{CCA1}, q\text{-CCA2}, \mathbf{CCA2}\}$ (with $q \in \mathbb{N}$). Assume there exists a distribution \mathbb{D} of an oracle \mathcal{O} satisfying the following two conditions:

1. There exists an algebraic oracle PPTA \mathcal{A} such that $\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}_{\Gamma, \mathcal{A}^{\mathcal{O}}}^{\mathbf{GOAL-ATK}}(\lambda)]$ is non-negligible.
2. For any algebraic oracle PPTA \mathcal{A} , there exist PPTAs \mathcal{B}_1 and \mathcal{B}_2 , a polynomial $Q(\lambda)$, and a negligible function $\mu(\lambda)$ such that

$$\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}_{\mathbf{GS}, \mathcal{A}^{\mathcal{O}}}^{\mathbf{P}}(\lambda)] \leq Q(\lambda) \cdot \mathsf{Adv}_{\mathbf{GS}, \mathcal{B}_1}^{\mathbf{DL}}(\lambda) + \mathsf{Adv}_{\mathbf{GS}, \mathcal{B}_2}^{\mathbf{P}}(\lambda) + \mu(\lambda).$$

Then, if \mathbf{P} is hard, there is no algebraic black-box reduction from the $\mathbf{GOAL-ATK}$ security of Γ to \mathbf{P} .

Proof Sketch. (A formal proof can be found in the full version.) The lemma is proved by contradiction. We assume simultaneously that the NIP $\mathbf{P} = (\mathbf{I}, \mathbf{V}, \mathbf{U})$ is hard and that there is an algebraic black-box reduction from the $\mathbf{GOAL-ATK}$ security of the KEM Γ to \mathbf{P} . The latter guarantees that there exists an algebraic oracle PPTA \mathcal{R} that satisfies Definition 4. We consider two separate cases: the discrete logarithm (DL) problem with respect to \mathbf{GS} is *hard*, and the DL problem with respect to \mathbf{GS} is *not* hard. For each case we will show a contradiction.

The second case is fairly easy to show and does not require the use of the oracle \mathcal{O} . Specifically, that the DL problem is not hard implies the existence of an adversary \mathcal{A}' that successfully breaks the KEM Γ in the sense of the $\mathbf{GOAL-ATK}$ security considered in the lemma, by simply recovering the randomness r from the challenge ciphertext. Then, the definition of \mathcal{R} implies that $\mathcal{R}^{\mathcal{A}'}$ can solve \mathbf{P} with non-negligible advantage. Here, note that $\mathcal{R}^{\mathcal{A}'}$ can be implemented by a single PPTA \mathcal{R}' which internally runs \mathcal{A}' since both \mathcal{R} and \mathcal{A}' are PPTAs. However, the existence of such \mathcal{R}' contradicts that \mathbf{P} is hard.

The first case, in which the DL problem is hard, has some similarities with the above case, but is more interesting and more involved. We make use of the oracle \mathcal{O} chosen according to \mathbb{D} . The first condition of the lemma guarantees the existence of an algebraic oracle PPTA \mathcal{A} which, given access to \mathcal{O} , has non-negligible “expected” advantage in breaking the $\mathbf{GOAL-ATK}$ security of the KEM Γ , where the expectation is over the choice of \mathcal{O} according to \mathbb{D} .

Now, in order to reach a contradiction, we would like to construct an algebraic oracle PPTA $\widehat{\mathcal{R}}$ which, given access to \mathcal{O} (chosen according to \mathbb{D}), has non-negligible “expected” advantage in solving \mathbf{P} , by using \mathcal{A} and the reduction algorithm \mathcal{R} as building blocks. One might think that just running $\mathcal{R}^{\mathcal{A}^{\mathcal{O}}}$ on input the problem instance y given to $\widehat{\mathcal{R}}$ is enough for that purpose, but this is not the case. Recall that \mathcal{R} is only guaranteed to work when $\mathcal{A}^{\mathcal{O}}$ successfully breaks the $\mathbf{GOAL-ATK}$ security of Γ . Furthermore, only the “expected” advantage of \mathcal{A} is guaranteed to be non-negligible. Hence, there is a possibility that a particular choice of \mathcal{O} is “bad,” and that \mathcal{A} ’s advantage under this \mathcal{O} is not non-negligible. If the oracle \mathcal{O} is bad, then nothing is guaranteed about the success probability

of $\mathcal{R}^{\mathcal{A}^\mathcal{O}}$ in solving P . In particular, the advantage of \mathcal{R} might even be negative, and the overall expected advantage in solving P might not be non-negligible.

To deal with this issue, $\widehat{\mathcal{R}}$ first tests whether the given oracle \mathcal{O} is “good” by running $\mathcal{R}^{\mathcal{A}^{(\cdot)}}$ many times with independently generated problem instances. If the success probability of $\mathcal{R}^{\mathcal{A}^\mathcal{O}}$ (measured by $\widehat{\mathcal{R}}$) sufficiently exceeds the threshold $\delta(\lambda)$, then $\widehat{\mathcal{R}}$ labels the oracle “good” and runs $\mathcal{R}^{\mathcal{A}^\mathcal{O}}$ with the given instance y . Otherwise, $\widehat{\mathcal{R}}$ runs the threshold algorithm U on input y to avoid a heavy negative contribution to the expected advantage. Constructed as above, $\widehat{\mathcal{R}}$ ’s expected advantage (over the choice of \mathcal{O} according to \mathbb{D}) can be shown to be non-negligible. Then, the existence of such an algebraic oracle algorithm, together with the second condition given in the lemma and the assumption that the DL problem is hard, implies the existence of a PPTA \mathcal{B}_2 that solves P with non-negligible advantage, which again contradicts that P is hard. Hence, we can conclude that either P is not hard or there exists no algebraic black-box reduction from the **GOAL-ATK** security of Γ to the hardness of P . \square

To make use of the above Lemma 7, we will first define a distribution of oracles, and then proceed to show that the conditions 1 and 2 are satisfied for this distribution. This will complete the proof of Theorem 6.

4.1 The Oracle and the Distribution

The oracle we consider is associated to a KEM which belongs to the class $\mathcal{K}_{\text{GS},n}$. More specifically, for any KEM $\Gamma \in \mathcal{K}_{\text{GS},n}$ and corresponding public key space \mathcal{PK} , consider an oracle $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2\}$ defined by a function $F : \mathcal{PK} \rightarrow \mathbb{Z}_p^n \times \{0, 1\}^\lambda$ where n indicates the number of group elements in a public key of Γ (i.e. $pk = (\Lambda, aux, X_1, \dots, X_n)$ and $\Lambda = (g, p, \mathbb{G}) \leftarrow \text{GS}(1^\lambda)$):

- \mathcal{O}_1 takes as input a public key pk , and returns \perp if $pk \notin \mathcal{PK}$. Otherwise, \mathcal{O}_1 computes $(r_1, \dots, r_n, \sigma) \leftarrow F(pk)$ and the ciphertexts $\{(C_i, K_i) \leftarrow \text{Enc}(pk; r_i)\}_{i \in [n]}$. Lastly, \mathcal{O}_1 returns $(C_1, \dots, C_n, \sigma)$.
- \mathcal{O}_2 takes as input a public key pk , session-keys $(K_1, \dots, K_n) \in \mathbb{G}^n$, and a tag $\sigma \in \{0, 1\}^\lambda$. If $pk \notin \mathcal{PK}$, then \mathcal{O}_2 returns \perp . Otherwise, \mathcal{O}_2 computes $(r_1, \dots, r_n, \sigma') \leftarrow F(pk)$ and $(C_i, K'_i) \leftarrow \text{Enc}(pk; r_i)$ for each $i = [n]$, and then checks if $K_i = K'_i$ for all $i \in [n]$ and $\sigma = \sigma'$. If the check fails, then \mathcal{O}_2 returns \perp . Otherwise, \mathcal{O}_2 computes $\{u_i = \psi_1(pk, C_i, y_1, \dots, y_n)\}_{i \in [n]}$ and returns these values, where ψ_1 is the scheme-dependent function of Γ .

By picking the function $F : \mathcal{PK} \rightarrow \mathbb{Z}_p^n \times \{0, 1\}^\lambda$ uniformly at random from all possible functions with the proper domain and range, we obtain a distribution of the defined oracle \mathcal{O} . In the following, this distribution will be denoted \mathbb{D} .

4.2 Breaking a KEM Using the Oracle

We will now show that an oracle \mathcal{O} chosen according to the distribution \mathbb{D} defined in Section 4.1 can be used to break the **OW- n -CCA1** security of a KEM in $\mathcal{K}_{\text{GS},n}$.

Lemma 8. Let GS be a group scheme, $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ be a KEM where $n \in \mathbb{N}$, and let \mathbb{D} be the distribution of the oracles $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ described above. Then there exists an algebraic oracle PPTA \mathcal{A} such that $\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}_{\Gamma, \mathcal{A}, \mathcal{O}}^{\mathsf{OW}-n\text{-CCA1}}(\lambda)] \geq \frac{3}{4n^2}$.

Proof Sketch. (A formal proof can be found in the full version.) In the following, we consider the $\mathsf{OW}-n\text{-CCA1}$ experiment in which we take into account the choice of oracle \mathcal{O} according to \mathbb{D} . Note that the success probability of an adversary \mathcal{A} in this experiment is given by $\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}_{\Gamma, \mathcal{A}, \mathcal{O}}^{\mathsf{OW}-n\text{-CCA1}}(\lambda)]$.

Recall that the session-key K of a ciphertext $C = (c, d)$ of the KEM Γ is of the form $K = g^{\psi_0(pk, C, y_1, \dots, y_n)} c^{\psi_1(pk, C, y_1, \dots, y_n)}$ where $\psi_1(pk, C, y_1, \dots, y_n) = \psi_{1,0}(pk, C) + \sum_{i \in [n]} \psi_{1,i}(pk, C) \cdot y_i$. Therefore, from a $\mathsf{OW}-n\text{-CCA1}$ adversary's viewpoint, the difficulty of recovering a session-key K from a ciphertext C must lie in the calculation of the component $c^{\sum_{i \in [n]} \psi_{1,i}(pk, C) \cdot y_i}$. However, we construct an algebraic oracle PPTA \mathcal{A} that makes use of the oracle \mathcal{O} and the decapsulation oracle \mathcal{O}_{dec} to calculate this component for the challenge ciphertext C^* , and thereby break the $\mathsf{OW}-n\text{-CCA1}$ security of Γ . \mathcal{A} is constructed as follows:

Given a public key $pk = (A, aux, X_1, \dots, X_n)$ for Γ , \mathcal{A} simply submits this to \mathcal{O}_1 to obtain n randomly generated ciphertexts (C_1, \dots, C_n) under pk . Then \mathcal{A} submits (C_1, \dots, C_n) to \mathcal{O}_{dec} to obtain the corresponding decapsulations (K_1, \dots, K_n) . Lastly, \mathcal{A} will submit (K_1, \dots, K_n) to \mathcal{O}_2 to obtain the values $\{u_j = \psi_{1,0}(pk, C_j) + \sum_{i \in [n]} \psi_{1,i}(pk, C_j) \cdot y_i\}_{j \in [n]}$, where $y_i = \log_g X_i$ for $i \in [n]$.

For a ciphertext C , let $\psi(pk, C) = (\psi_{1,1}(pk, C), \dots, \psi_{1,n}(pk, C)) \in \mathbb{Z}_p^n$ be a row vector. Furthermore, let $\mathbf{y}^T = (y_1, \dots, y_n) \in \mathbb{Z}_p^n$ and let $u'_j = u_j - \psi_{1,0}(pk, C_j)$ for all $j \in [n]$. Using the values returned by \mathcal{O}_2 , \mathcal{A} can construct a system of equations $\{\psi(pk, C_j) \cdot \mathbf{y} = u'_j\}_{j \in [n]}$. If the vectors $\{\psi(pk, C_j)\}_{j \in [n]}$ are linearly independent, this system will have the unique solution $\mathbf{y}^T = (y_1, \dots, y_n)$, and \mathcal{A} will be able to recover this by solving the equation system. Obtaining \mathbf{y} allows \mathcal{A} to trivially calculate K^* .

The tricky part is the case in which the vectors $\{\psi(pk, C_j)\}_{j \in [n]}$ are linearly dependent. Recall that the ciphertexts (C_1, \dots, C_n) are generated randomly by the oracle \mathcal{O}_1 , and that the challenge ciphertext C^* is likewise randomly generated by the $\mathsf{OW}-n\text{-CCA1}$ experiment. The key observation, which we will show in the full proof, is that if the vectors $\{\psi(pk, C_j)\}_{j \in [n]}$ are linearly dependent, then there is a high probability that the vector $\psi(pk, C^*)$ is linearly dependent on the $n-1$ vectors $\{\psi(pk, C_j)\}_{j \in [n-1]}$. Hence, $\psi(pk, C^*) \cdot \mathbf{y} = \sum_{i \in [n]} \psi_{1,i}(pk, C^*) \cdot y_i$ can be represented as a linear combination of the $n-1$ values $\{u'_j = \psi(pk, C_j) \cdot \mathbf{y}\}_{j \in [n-1]}$, where the latter are known to \mathcal{A} . Therefore, in this case, \mathcal{A} can calculate $\psi_1(pk, C^*) = \psi_{1,0}(pk, C^*) + \sum_{i \in [n]} \psi_{1,i}(pk, C^*) \cdot y_i$, from which \mathcal{A} can recover K^* .

In the full proof, we will show that regardless of the probability that the n vectors $\{\psi(pk, C_j)\}_{j \in [n]}$ are linearly dependent, \mathcal{A} will have an expected success probability with the claimed lower bound. \square

Breaking Non-malleability. Inspecting the proof of the above lemma reveals that the n decapsulation queries made by \mathcal{A} can be done in a single parallel decapsulation query containing n ciphertexts. Since indistinguishability against a parallel

chosen ciphertext attack is implied by the notion of non-malleability [18,26], this implies that the constructed adversary can be used to successfully attack the non-malleability of the KEM Γ as well. Hence, our impossibility result can easily be extended to rule out the existence of an algebraic black-box reduction from the non-malleability of a KEM $\Gamma \in \mathcal{K}_{\text{GS},n}$ to a NIP P with respect to GS.

4.3 Simulating the Oracle While Solving a NIP

In this subsection, we will show that the oracle defined in Section 4.1 is essentially useless for an algebraic algorithm trying to solve a NIP.

Lemma 9. *Let GS be a group scheme, $\Gamma \in \mathcal{K}_{\text{GS},n}$ be a KEM where $n \in \mathbb{N}$, and P be a NIP with respect to GS. Furthermore, let \mathbb{D} be the distribution of the oracles $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ (corresponding to Γ) defined as above. Then, for any algebraic oracle PPTA \mathcal{A} , there exist PPTAs \mathcal{B}_1 and \mathcal{B}_2 , a polynomial $Q(\lambda)$, and a negligible function $\mu(\lambda)$ such that*

$$\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}_{\text{GS},\mathcal{A}^{\mathcal{O}}}^{\mathsf{P}}(\lambda)] \leq Q(\lambda) \cdot \mathsf{Adv}_{\text{GS},\mathcal{B}_1}^{\mathsf{DL}}(\lambda) + \mathsf{Adv}_{\text{GS},\mathcal{B}_2}^{\mathsf{P}}(\lambda) + \mu(\lambda).$$

Proof Sketch. (A formal proof can be found in the full version.) In the following, we consider the NIP hardness experiment $\mathsf{Expt}_{\text{GS},\mathcal{A}}^{\mathsf{P}}(\lambda)$ in which we take into account the choice of the oracle \mathcal{O} according to \mathbb{D} . Note that the advantage of an adversary \mathcal{A} in this experiment is given by $\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}_{\text{GS},\mathcal{A}^{\mathcal{O}}}^{\mathsf{P}}(\lambda)]$.

To prove the lemma, we show that for any algebraic oracle PPTA \mathcal{A} with access to \mathcal{O} and which attempts to solve the NIP P , it is possible to construct another PPTA \mathcal{B}_2 which has almost the same advantage as \mathcal{A} in solving the same P without access to \mathcal{O} .

More specifically, \mathcal{B}_2 will make use of \mathcal{A} as a building block and simulate the oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ chosen according to \mathbb{D} for \mathcal{A} . \mathcal{B}_2 takes a problem instance y (of P) as input, and generates an empty list L which is used to simulate \mathcal{O} . Then \mathcal{B}_2 picks randomness $r_{\mathcal{A}}$ and runs \mathcal{A} with input y and randomness $r_{\mathcal{A}}$.

The main difficulty of simulating the oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ is that \mathcal{A} may use \mathcal{O}_1 and \mathcal{O}_2 multiple times in any order. Recall, however, that when chosen according to \mathbb{D} , the function F used in \mathcal{O}_1 and \mathcal{O}_2 is a random function, and the tag $\sigma \in \{0, 1\}^\lambda$, which is contained in the output of F , works like an information-theoretically secure MAC. Therefore, when \mathcal{A} asks an \mathcal{O}_2 -query with a fresh pk (that has not appeared in any of \mathcal{A} 's previous queries), \mathcal{B}_2 can immediately return \perp , which will be an almost perfect simulation of \mathcal{O}_2 for this type of query. \mathcal{B}_2 simulates \mathcal{O}_1 by “lazy-sampling” of the random function F and generating ciphertexts $\{C_i = (c_i, d_i)\}_{i \in [n]}$ using Enc . All values returned to \mathcal{A} in an \mathcal{O}_1 -query, as well as the encapsulated session-keys, are stored by \mathcal{B}_2 in the list L . Furthermore, when \mathcal{A} makes a valid \mathcal{O}_2 -query $(pk, K_1, \dots, K_n, \sigma)$ (for which \mathcal{O}_2 will not return \perp), \mathcal{B}_2 can run the extractor corresponding to (the decomposition of) \mathcal{A} to obtain values $\{u_i\}_{i \in [n]}$ such that $K_i = c_i^{u_i} g^{z_i}$ for some value $z_i \in \mathbb{Z}_p$ unknown to \mathcal{B}_2 . Lastly, \mathcal{B}_2 returns (u_1, \dots, u_n) . Whether the values $\{K_i\}_{i \in [n]}$ are correct decapsulation results can be checked using the list L . Note that since the randomness $r_{\mathcal{A}}$ is chosen by \mathcal{B}_2 , \mathcal{B}_2 is able to run the extractor for \mathcal{A} .

Here, however, we have to be careful because the above simulation could fail if either of the following events occurs: (1) the extractor fails, or (2) there is an index $i \in [n]$ such that the extracted value u_i is different from $\psi_1(pk, C_i, y_1, \dots, y_n)$. Fortunately, the probability of (1) occurring is negligible by the definition of an algebraic oracle algorithm. Moreover, the probability of (2) occurring for an index $i \in [n]$ in one of \mathcal{A} 's \mathcal{O}_2 -queries can be bounded by the advantage $\text{Adv}_{\text{GS}, \mathcal{B}_1}^{\text{DL}}(\lambda)$ of another PPTA \mathcal{B}_1 which solves the DL problem. Put differently, \mathcal{B}_2 succeeds in simulating the oracle \mathcal{O} for \mathcal{A} almost perfectly. Furthermore, since \mathcal{B}_2 succeeds in solving P whenever \mathcal{A} does, the lemma follows. \square

5 Lower Bounds for Programmable Hash Functions

In this section, we show lower bounds on the “programmability” of an algebraic programmable hash functions defined in a prime order group. We will do this indirectly, by first showing how to construct a CCA secure KEM based on the DDH assumption, with a ciphertext consisting of just a single group element, from an algebraic programmable hash function. Since this KEM construction is captured by the class considered in the previous sections, we can derive the lower bounds by combining this with the previous impossibility result.

5.1 Programmable Hash Functions

Definition 10. Let $\alpha, \beta \in \mathbb{N}$. A (α, β) -programmable hash function H with respects to a group scheme GS and with input length $\ell(\lambda)$, consists of the following four algorithms (HGen , Eval , HTrapGen , HTrapEval)

- HGen takes a group description $\Lambda = (g, p, \mathbb{G})$ (output from $\text{GS}(1^\lambda)$) and returns a hash key κ .
- Eval takes κ and a string $s \in \{0, 1\}^{\ell(\lambda)}$ as input, and returns a group element of \mathbb{G} .
- HTrapGen takes Λ and group elements h_1, h_2 as input, and returns a hash key κ and a trapdoor τ .
- For all group elements $h_1, h_2 \in \mathbb{G}$, the statistical difference between the keys $\kappa \leftarrow \text{HGen}(\Lambda)$ and the first component κ of the output from $\text{HTrapGen}(\Lambda, h_1, h_2)$ is negligible.
- On input a string $s \in \{0, 1\}^{\ell(\lambda)}$ and trapdoor τ , HTrapEval returns $a_s, b_s \in \mathbb{Z}_p$ such that $\text{Eval}(\kappa, s) = h_1^{a_s} h_2^{b_s}$.
- For all group elements $h_1, h_2 \in \mathbb{G}$ and $(\kappa, \tau) \leftarrow \text{HTrapGen}(\Lambda)$, and for all strings $s_1, \dots, s_\alpha \in \{0, 1\}^{\ell(\lambda)}$ and $s'_1, \dots, s'_\beta \in \{0, 1\}^{\ell(\lambda)}$ such that $s_i \neq s'_j$ for all i, j , the probability $\Pr[a_{s_1} = \dots = a_{s_\alpha} = 0 \wedge a_{s'_1}, \dots, a_{s'_\beta} \neq 0]$ is noticeable in λ , where $(a_{s_i}, b_{s_i}) \leftarrow \text{HTrapEval}(\tau, s_i)$, $(a_{s'_j}, b_{s'_j}) \leftarrow \text{HTrapEval}(\tau, s'_j)$, and the probability is taken over the randomness used by HTrapGen .

If H is $(q(\lambda), \beta)$ -programmable for every polynomial $q(\lambda)$, we say that H is (poly, β) -programmable. Furthermore, we say that a programmable hash function is *algebraic* if all algorithms of H are algebraic algorithms.

In the following, we will make explicit use of the extractors for HGen and Eval . More specifically, let $\kappa \leftarrow \text{HGen}(\Lambda)$ be given by $\kappa = (\text{aux}, X_1, \dots, X_n)$ where $X_i \in \mathbb{G}$ for all $i \in [n]$ and it is assumed that aux does not contain any elements of \mathbb{G} . Let $h \in \mathbb{G}$ be an element returned by Eval on input a string s . Then, if HGen and Eval are algebraic algorithms, there exist extractors $\mathcal{E}_{\text{HGen}}$ and $\mathcal{E}_{\text{Eval}}$ with the following properties:

- On input $\Lambda = (g, p, \mathbb{G})$ and randomness r_{HGen} used to run HGen , $\mathcal{E}_{\text{HGen}}$ returns values (y_1, \dots, y_n) such that $X_i = g^{y_i}$ for all $i \in [n]$.
- On input $\kappa = (\text{aux}, X_1, \dots, X_n)$ and a string $s \in \{0, 1\}^{\ell(\lambda)}$, $\mathcal{E}_{\text{Eval}}$ returns values (a_1, \dots, a_n) such that $h = \prod_{i \in [n]} X_i^{a_i} = \text{Eval}(\kappa, s)$.

We note that all known constructions of programmable hash functions [21, 19] are algebraic.

5.2 A Simple KEM Based on a Programmable Hash Function

We now show how to construct an IND- q -CCA2 secure KEM with ciphertexts consisting of a single group element, from an algebraic $(q, 1)$ -programmable hash function. Let $H = (\text{HGen}, \text{Eval}, \text{HTrapGen}, \text{HTrapEval})$ be an algebraic programmable hash function with respect to GS. Let $\ell(\lambda)$ be the input length of H . We also assume that any group element of \mathbb{G} where $\Lambda = (g, p, \mathbb{G}) \leftarrow \text{GS}(1^\lambda)$ can be described with $\ell(\lambda)$ bits. Then we construct a KEM Γ as follows:

KG : On input $\Lambda = (g, p, \mathbb{G})$, pick randomness r_{HGen} for HGen and run $\kappa = (\text{aux}, X_1, \dots, X_n) \leftarrow \text{HGen}(\Lambda; r_{\text{HGen}})$. Furthermore, run $\mathcal{E}_{\text{HGen}}(\Lambda, r_{\text{HGen}})$ to obtain (y_1, \dots, y_n) such that $X_i = g^{y_i}$ for all $i \in [n]$, and set $pk \leftarrow (\Lambda, \kappa)$ and $sk \leftarrow (\kappa, y_1, \dots, y_n)$.

Enc : On input $pk = (\Lambda, \kappa)$, pick randomness $r \in \mathbb{Z}_p$, and compute the ciphertext $c = g^r$ and the session-key $K = \text{Eval}(\kappa, c)^r$. (Here, c is treated as an $\ell(\lambda)$ -bit string.)

Dec : On input $sk = (\kappa, y_1, \dots, y_n)$ and $c = g^r$, compute $h \leftarrow \text{Eval}(\kappa, c)$, run the extractor $\mathcal{E}_{\text{Eval}}$ to obtain (a_1, \dots, a_n) satisfying $h = \prod_{i \in [n]} X_i^{a_i}$, and compute the session-key $K = c^{\sum_{i \in [n]} a_i y_i}$.

The correctness of the KEM follows from the properties of the extractors $\mathcal{E}_{\text{HGen}}$ and $\mathcal{E}_{\text{Eval}}$:

$$K = \text{Eval}(\kappa, c)^r = h^r = \left(\prod_{i \in [n]} X_i^{a_i} \right)^r = \prod_{i \in [n]} (g^r)^{a_i y_i} = c^{\sum_{i \in [n]} a_i y_i}$$

Note that the DDH-based KEM by Cramer et al. [9] can be seen as a concrete instantiation of the above KEM in which we use the concrete programmable hash function proposed in [19, Sect. 3.3].

Theorem 11. *Let GS be a group scheme. Assume that H is an algebraic $(q, 1)$ -programmable hash function with respect to GS. Then there exists an algebraic black-box reduction from the IND- q -CCA2 security of the above KEM Γ to the hardness of the DDH problem with respect to GS.*

The proof of the above theorem is given in the full version.

Note that due to the assumed algebraic property of the programmable hash function, the above KEM Γ falls into the class $\mathcal{K}_{GS,n}$ described in Section 3, where n is the number of group elements in the hash key of the programmable hash. Furthermore, the DDH problem is captured by the definition of a non-interactive problem described in Section 2. Hence Theorem 6 implies that there exists no algebraic black-box reduction from the $OW-n\text{-CCA1}$ security of the KEM Γ to the hardness of any non-interactive problem⁴. On the other hand, the above Theorem 11 shows that such a reduction (from the stronger security notion $IND-n\text{-CCA2}$) is possible assuming the existence of an algebraic $(n, 1)$ -programmable hash function. Since any (n, β) -programmable hash function is (n, β') -programmable if $\beta \geq \beta'$, this immediately gives us the following theorem.

Theorem 12. *For any group scheme GS and any integer $\beta \in \mathbb{N}$, there exists no algebraic (n, β) -programmable hash function with respect to GS whose hash key contains less than n group elements of \mathbb{G} , where \mathbb{G} is the group described by Λ which is output by GS .*

Considering the case in which the parameter n for the programmable hash functions is considered to be any polynomial in λ , we obtain the following theorem, which answers the open question posed by Hofheinz and Kiltz [21] in the case of algebraic programmable hash function defined in prime order groups.

Theorem 13. *For any group scheme GS and any integer $\beta \in \mathbb{N}$, there exists no algebraic (poly, β) -programmable hash functions with respect to GS .*

6 Discussion

We have shown that there exists no algebraic black-box reduction from the $OW-n\text{-CCA1}$ security of a KEM in the class $\mathcal{K}_{GS,n}$, to the hardness of any non-interactive problem with respect to GS . The class $\mathcal{K}_{GS,n}$ essentially captures the structure of the efficient KEMs [10,20], [15, Sect. 4.1], and [25] (with explicit rejection), but requires the ciphertext to consist of just a single random group element and a string.

Our results leave several open problems. Specifically, it remains an open problem to prove the (non-)existence of a CCA secure KEM based on a non-interactive assumption, defined in a standard prime order group, and with a ciphertext overhead of just two group elements. Another interesting question is whether our results can be extended to rule out constrained CCA [20] secure KEMs based on non-interactive assumptions.

Furthermore, we have focused on simple KEMs in which the session-key lies within the group. More precisely, the KEM class $\mathcal{K}_{GS,n}$ does not capture the

⁴ Note that this does not contradict the results by Cramer et al. [9]. More specifically, while the KEM defined in [9] was shown to be $IND-n\text{-CCA}$ under the DDH assumption via an algebraic black-box reduction, the scheme requires a public key containing $\mathcal{O}(n^2\lambda)$ group elements.

structure of schemes which make use of a pairing to derive the session-key like [4], or apply a type of key-derivation function, such as the hardcore bit-based schemes like [15, Sect. 5], [17, Sect. 3], and [31, Sect. 3], the HDH-based versions of [15, Sect. 4.2] and [6, Sect. 6.2], or a combination of these like [17, Sect. 5.2 and 5.3] and [31, Sect. 5]. Note, however, that these schemes apply the key-derivation function (and/or the pairing) to one or more “seed” group elements to obtain a session-key. Here, it might be interesting to investigate the security provided by the “core part” of the schemes, in which the seed group element(s) is considered to be the session-key. Note that for all of the above mentioned schemes, these “core parts” can be shown to be OW-CCA2 secure under an appropriate non-interactive assumption. Furthermore, the structure of these “core parts” is captured by $\mathcal{K}_{GS,n}$ if the ciphertext is reduced to consist of a single random group element and a string, for example, by applying the approach of compressing the group elements used for validity checking. In this case, our results imply that these “core parts” cannot be shown $\text{OW-}n\text{-CCA1}$ secure based on a non-interactive assumption via an algebraic black-box reduction. This observation might provide some insight into the (im)possibility of constructing more efficient KEMs that make use of key-derivation functions, but drawing any formal conclusions regarding this, remains an open problem.

Since our results are restricted to KEMs defined in prime order groups, it is natural to ask whether similar results will hold in composite order groups. Note, however, that in composite order groups, it is possible to achieve a KEM with a ciphertext overhead of just a single group element [22, Sect. 5]. While this KEM only achieves constrained CCA security (based on a non-interactive assumption), it can be converted to a fully CCA secure KEM using the techniques from [2] which will result in an additional ciphertext overhead of a MAC.

Lastly, we have shown lower bounds on the programmability of algebraic programmable hash functions in prime order groups. Furthermore, the definition of a programmable hash function requires the hash function to have some “algebraic properties” (see the discussion in [21, Sect. 1.5]), which seems to suggest that constructions of programmable hash functions are inherently algebraic.

Acknowledgement. We would like to thank Shota Yamada for suggesting looking at KEMs based on programmable hash functions, and the anonymous reviewers for helpful comments. The second and third authors are supported by JSPS Research Fellowships for Young Scientists.

References

1. Abe, M., Groth, J., Ohkubo, M.: Separating Short Structure-Preserving Signatures from Non-interactive Assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 628–646. Springer, Heidelberg (2011)
2. Baek, J., Galindo, D., Susilo, W., Zhou, J.: Constructing Strong KEM from Weak KEM (or How to Revive the KEM/DEM Framework). In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 358–374. Springer, Heidelberg (2008)

3. Boneh, D., Venkatesan, R.: Breaking RSA May Not Be Equivalent to Factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (1998)
4. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security from Identity-Based Techniques. Cryptology ePrint Archive, Report 2005/288 (2005)
5. Camenisch, J., Haralambiev, K., Kohlweiss, M., Lapon, J., Naessens, V.: Structure Preserving CCA Secure Encryption and Applications. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 89–106. Springer, Heidelberg (2011)
6. Cash, D., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman Problem and Applications. Cryptology ePrint Archive, Report 2008/067 (2008), <http://eprint.iacr.org/>, This is the full version of [7]
7. Cash, D., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman Problem and Applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
8. Coron, J.-S.: Optimal Security Proofs for PSS and Other Signature Schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
9. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-Secure Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)
10. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. Computing 33(1), 167–226 (2003)
11. Emura, K., Hanaoka, G., Matsuda, T., Otake, G., Yamada, S.: Chosen Ciphertext Secure Keyed-Homomorphic Public-Key Encryption (2012) (manuscript)
12. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The Relationship between Public Key Encryption and Oblivious Transfer. In: FOCS, pp. 325–335 (2000)
13. Gertner, Y., Malkin, T., Myers, S.: Towards a Separation of Semantic and CCA Security for Public Key Encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 434–455. Springer, Heidelberg (2007)
14. Gertner, Y., Malkin, T., Reingold, O.: On the Impossibility of Basing Trapdoor Functions on Trapdoor Predicates. In: FOCS, pp. 126–135 (2001)
15. Hanaoka, G., Kurosawa, K.: Efficient Chosen Ciphertext Secure Public Key Encryption under the Computational Diffie-Hellman Assumption. Cryptology ePrint Archive, Report 2008/211 (2008), <http://eprint.iacr.org/>, This is the full version of [16]
16. Hanaoka, G., Kurosawa, K.: Efficient Chosen Ciphertext Secure Public Key Encryption under the Computational Diffie-Hellman Assumption. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 308–325. Springer, Heidelberg (2008)
17. Haralambiev, K., Jager, T., Kiltz, E., Shoup, V.: Simple and Efficient Public-Key Encryption from Computational Diffie-Hellman in the Standard Model. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 1–18. Springer, Heidelberg (2010)
18. Herranz, J., Hofheinz, D., Kiltz, E.: Some (in)sufficient conditions for secure hybrid encryption. Inf. Comput. 208(11), 1243–1257 (2010)
19. Hofheinz, D., Jager, T., Kiltz, E.: Short Signatures From Weaker Assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 647–666. Springer, Heidelberg (2011)

20. Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
21. Hofheinz, D., Kiltz, E.: Programmable Hash Functions and Their Applications. *J. of Cryptology* 25(3), 484–527 (2012)
22. Hofheinz, D., Kiltz, E.: The Group of Signed Quadratic Residues and Applications. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 637–653. Springer, Heidelberg (2009)
23. Impagliazzo, R., Rudich, S.: Limits on the Provable Consequences of One-Way Permutations. In: STOC, pp. 44–61 (1989)
24. Kiltz, E.: Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
25. Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
26. Matsuda, T., Matsuura, K.: Parallel Decryption Queries in Bounded Chosen Ciphertext Attacks. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 246–264. Springer, Heidelberg (2011)
27. Phan, D.H., Pointcheval, D.: About the Security of Ciphers (Semantic Security and Pseudo-Random Permutations). In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 182–197. Springer, Heidelberg (2004)
28. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of Reducibility between Cryptographic Primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
29. Simon, D.R.: Findings Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
30. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
31. Yamada, S., Kawai, Y., Hanaoka, G., Kunihiro, N.: Public Key Encryption Schemes from the (B)CDH Assumption with Better Efficiency. IEICE Transactions 93-A(11), 1984–1993 (2010)
32. Yerukhimovich, A.: A Study of Separation in Cryptography: New Results and New Models. PhD thesis, the University of Maryland (2011), <http://www.cs.umd.edu/~arkady/thesis/thesis.pdf>

Hardness of Computing Individual Bits for One-Way Functions on Elliptic Curves

Alexandre Duc¹ and Dimitar Jetchev²

¹ LASEC,

² LACAL,

EPFL, 1015 Lausanne, Switzerland

Abstract. We prove that if one can predict any of the bits of the input to an elliptic curve based one-way function over a finite field, then we can invert the function. In particular, our result implies that if one can predict any of the bits of the input to a classical pairing-based one-way function with non-negligible advantage over a random guess then one can efficiently invert this function and thus, solve the Fixed Argument Pairing Inversion problem (FAPI-1/FAPI-2). The latter has implications on the security of various pairing-based schemes such as the identity-based encryption scheme of Boneh–Franklin, Hess’ identity-based signature scheme, as well as Joux’s three-party one-round key agreement protocol. Moreover, if one can solve FAPI-1 and FAPI-2 in polynomial time then one can solve the Computational Diffie–Hellman problem (CDH) in polynomial time.

Our result implies that all the bits of the functions defined above are hard-to-compute assuming these functions are one-way. The argument is based on a list-decoding technique via discrete Fourier transforms due to Akavia–Goldwasser–Safra as well as an idea due to Boneh–Shparlinski.

Keywords: One-way function, hard-to-compute bits, bilinear pairings, elliptic curves, fixed argument pairing inversion problem, Fourier transform, list decoding.

1 Introduction

One-way functions (OWF) are functions that are easy to compute but hard to invert. Yet, the definition of a one-way function does not say much about the security of a particular predicate over the input of this function. What if, for instance, the least significant bit is easy to compute? In this case, one might be able to leak partial information if one hides the secret key using this one-way function. Hence, proving that partial information is hard to predict is of primary interest.

In this paper, we study hardness of computing individual bits for one-way functions whose domains are subgroups of points on elliptic curves. Before we ask any question about extracting bits from the input of such functions, we need to specify a binary representation of the points of the subgroup. This amounts to specifying a particular short Weierstrass equation (model) representing the

isomorphism class of the elliptic curve and then specifying a binary representation for the finite field to obtain a representation of the coordinates of the points on this Weierstrass model. Since two distinct short Weierstrass equations could represent the same isomorphism class, it is important to distinguish between elliptic curves (taken up to isomorphism) and short Weierstrass equations.

From the point of view of the hardness of classical computational problems in cryptography (e.g., the elliptic curve discrete logarithm problem or the Diffie–Hellman problem), it makes no difference which short Weierstrass representation one chooses for a given isomorphism class since a solution of the problem on one Weierstrass representation yields (via the isomorphism transformation) a solution on any other Weierstrass representation for the same isomorphism class of elliptic curves. Yet, knowing that one can predict the least significant bit of the input for a particular Weierstrass equation does not necessarily imply that one can predict with the same advantage the least-significant bit on another short Weierstrass equation representing the same curve.

Suppose that we are interested in defining a function whose domain is a particular subgroup of points $\mathbb{G} \subseteq E(\mathbb{F}_p)$ of \mathbb{F}_p -points on the curve. By an efficiently computable elliptic curve based function from \mathbb{G} to a set \mathcal{Y} we mean a function $f: \mathbb{G} \rightarrow \mathcal{Y}$ that can be efficiently computed on any of the short Weierstrass representations of \mathbb{G} and that is independent of the short Weierstrass representation for E . In other words, it is a function on the isomorphism class of E . Our main example for elliptic curve based functions will be functions arising from classical elliptic curve cryptographic pairings: if $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a cryptographic pairing and if $Q \in \mathbb{G}$, then $f_Q: \mathbb{G} \rightarrow \mathbb{G}_T$ defined by $f_Q(P) := e(P, Q)$ is an elliptic curve based function. This function is conjectured to be one-way and is essential in Boneh and Franklin’s identity-based encryption scheme (IBE) [5]. The one-wayness of f_Q (a problem known as the Fixed Argument Pairing Inversion 2 (FAPI-2)) is also linked to the security of Joux’s three-party one-round key agreement protocol [23] and to the identity-based signature scheme by Hess [19]. This problem and potential approaches to solve it is studied in [11]. More generally, the hardness of f_Q is linked to the hardness of the bilinear Diffie–Hellman problem (BDH) [5].

In our main result (Theorem 1), we show that given an elliptic curve based function, if one can predict (with non-negligible advantage over the point in \mathbb{G} as well as the short Weierstrass equation for E) the k th bit of the input to f then one can efficiently invert f . More precisely, we consider an elliptic curve based function $f: \mathbb{G} \rightarrow \mathcal{Y}$ on a cyclic subgroup $\mathbb{G} \subseteq E(\mathbb{F}_p)$ of points and we show that if there is an algorithm that takes as input $f(R)$ for some hidden element $R \in \mathbb{G}$ and a short Weierstrass model for E and predicts the k th bit of the x -coordinate of R on that model, then we can invert f (here, the prediction algorithm is imperfect and the only requirement is that it works with non-negligible advantage measured over a random point $R \in \mathbb{G}$ as well as a random short Weierstrass equation representing E). The major consequence of our result is that *all* the bits of the input to the pairing-based one-way function are hard-to-compute assuming that FAPI-2 is hard. More generally, this result

applies to any elliptic curve based one-way function. Our proof uses methods developed by Akavia et al. [2] based on list-decoding via discrete Fourier transforms. We introduce a new code, the *elliptic curve multiplication code*, similar to the multiplication code presented in [2] but whose predicates are evaluated over different short Weierstrass equations. We show that, given access to the k th bit of the x -coordinate of R , we have access to a noisy codeword that can be list-decoded to recover R resulting in an inversion of the one-way function. We believe this code might be of independent interest.

1.1 Previous Work

The first hard-to-compute predicate for a one-way function was found by Blum and Micali [4] for the discrete logarithm (DL) one-way function over a finite field \mathbb{F}_p . Subsequently, the question of constructing predicates that are hard-to-compute from one-way functions was studied extensively. For instance, Håstad and Näslund showed that every bit in the RSA [33] one-way function is hard-to-compute [17] using a result of Alexi, Chor, Goldreich and Schnorr [3]. Similarly, for the DL one-way function, Håstad, Schrift and Shamir showed that all the bits are hard-to-compute if the DL is taken modulo a Blum integer [18] following the work of Schrift and Shamir [36]. By changing the way the bits are represented, Schnorr showed that almost all of the bits in the DL function are hard-to-compute [35]. A similar hardness result (independent of the bit representation) was proven in [17]. The last two results also hold for the elliptic curve discrete logarithm (ECDL). For the elliptic curve Diffie–Hellman problem, the hardness of the LSB of the x - and y -coordinates has been studied as well [6, 22].

However, all these results apply to a specific one-way function and have to be significantly modified to be used on another OWF (or sometimes cannot be modified at all). Thus, finding generic hard-to-compute predicates that apply to more general collections of one-way functions is highly desirable [14, 13].

In 2003, Akavia, Goldwasser and Safra presented a new method to prove that some predicates are hard-to-compute for a one-way function [2]. Their work follows the work by Goldreich and Levin [14]. Using their methodology, security results can be proven for entire classes of functions. Furthermore, it is elegant to use and hides the cumbersome bit manipulations that appeared in the previous proofs. The method relies on the construction of a code that encodes the preimages of the one-way function we try to invert. This means that given a one way function $f: \mathcal{X} \rightarrow \mathcal{Y}$ and a predicate $P(x)$ for $x \in \mathcal{X}$, we construct a code \mathcal{C}^P that associates to $x \in \mathcal{X}$ a codeword $C_x^P \in \mathcal{C}^P$. If we have access to a corrupted codeword w (which we can get through the bit prediction oracle) and if there is a PPT algorithm that computes a list of all $x \in \mathcal{X}$ such that C_x^P is close to w (in the sense of Hamming distance), then the predicate P is hard-to-compute for f .

The method is used to prove the security of certain bits in RSA, in the Rabin cryptosystem [32], in the DL problem and in the ECDL problem. More precisely, one can prove the security of the $\mathcal{O}(\log \log n)$ least and most significant bits of these functions, where n is the size of the domain of the one-way function.

In 2009, Morillo and Ràfols [27] extended these results and were able to prove the security of *all* bits in RSA, Rabin and DL for prime orders or RSA moduli using a careful analysis of the Fourier coefficients of the function that maps an element of $\mathbb{Z}/n\mathbb{Z}$ to the value of the k th bit of its corresponding representative in $[0, n - 1]$. They also extended the result to the Paillier trapdoor permutation [31].

Previous results exist on bit-security for pairing-based protocols [12,34,39] as well as for other bit security results [15,16,28,29]. For instance, a version of the hidden number problem [6,7,8,9,20,26,30,37,38] has been studied by Galbraith et al. in [12] shedding some light on the security of protocols such as Joux's three-party key-agreement protocol [23]. Yet, this is not directly related to FAPI-2.

Overview. The paper is organized as follows: in Section 2, we introduce basic definitions and present our main theorem. In Section 3, we recall basic notions from discrete Fourier transforms on abelian groups used to prove our theorem as well as two important properties from list-decoding, namely Fourier concentration and recoverability. We also mention one of the key ingredients - an algorithm due to Akavia–Goldwasser–Safra that efficiently recovers the heavy Fourier coefficients of a noisy codeword. In Section 4, we prove our main theorem by introducing a new code that we call the elliptic curve multiplication code (ECMC) and by showing that it satisfies the two properties by using techniques due to Boneh–Shparlinski as well as Morillo and Ràfols. We explicitly present the full reduction algorithm by referring to the appendix for various technical details. In Section 5, we show how our result applies to pairing-based one-way functions and also discuss the implications.

2 Main Theorem

2.1 Preliminaries

Let p be a prime and let E be an elliptic curve over \mathbb{F}_p . Throughout, we always represent the elements of \mathbb{F}_p via the binary representations of the integers $\{0, 1, \dots, p - 1\}$.

In order to discuss the individual bits of a point on E , we fix a short Weierstrass equation $W: y^2 = x^3 + ax + b$, $a, b \in \mathbb{F}_p$, $4a^3 + 27b^2 \neq 0$ representing E . Let $\mathcal{W}(E)$ be the set of all such short Weierstrass equations. Two short Weierstrass equations $y^2 = x^3 + ax + b$ and $y^2 = x^3 + a'x + b'$ represent the same elliptic curve E over \mathbb{F}_p if and only if there exists an element $\lambda \in \mathbb{F}_p^\times$ such that $a' = \lambda^4 a$ and $b' = \lambda^6 b$. Hence, the set $\mathcal{W}(E)$ is in bijection with \mathbb{F}_p^\times . For a point $R \in E(\mathbb{F}_p)$ and $W \in \mathcal{W}(E)$, the x - and y -coordinates of R on the short Weierstrass model W are denoted by $(R_W)_x$ and $(R_W)_y$, respectively. Once a short Weierstrass equation $W: y^2 = x^3 + ax + b$ is fixed, we denote the short Weierstrass equation $y^2 = x^3 + \lambda^4 ax + \lambda^6 b$ by W_λ . Given any point Q on W , the point $Q_\lambda = (\lambda^2 x, \lambda^3 y)$ is on W_λ for any $\lambda \in \mathbb{F}_p^\times$.

Next, a function $\nu: \mathbb{N} \rightarrow \mathbb{R}$ is called *negligible* if for every constant $c \in \mathbb{R}_{>0}$, there exists $k_0 \in \mathbb{N}$ such that $|\nu(k)| < k^{-c}$ for all $k > k_0$. A function $\rho: \mathbb{N} \rightarrow \mathbb{R}$

is *non-negligible* if there exists a constant $c \in \mathbb{R}_{>0}$ and a $k_0 \in \mathbb{N}$ such that $|\rho(k)| > k^{-c}$ for all $k > k_0$.

Remark 1. Note that a function being *non-negligible* is a stronger requirement than a function not being negligible.¹ By the above definitions, every non-negligible function is not negligible. However, there are functions that are not negligible, but are not non-negligible. For example, take any non-negligible function $f: \mathbb{N} \rightarrow \mathbb{R}$ and define the function

$$g(n) = \begin{cases} 0 & \text{if } n \text{ is even,} \\ f(n) & \text{otherwise.} \end{cases}$$

Obviously, g is neither negligible nor non-negligible.

Let \mathcal{X} be a finite set and let \mathcal{D} be a probability distribution on \mathcal{X} . We write $x \in_{\mathcal{D}} \mathcal{X}$ if the element x is chosen according to the distribution \mathcal{D} from \mathcal{X} . We now recall some basic notions from cryptography:

Definition 1 (One-way function). A function $f: \mathcal{X} \rightarrow \mathcal{Y}$ is a one-way function (OWF) if the following conditions hold:

- Given $x \in \mathcal{X}$, one can compute $f(x)$ in polynomial time in $\log |\mathcal{X}|$,
- For every probabilistic polynomial time (PPT) (in $\log |\mathcal{X}|$) algorithm \mathcal{A} , there exists a negligible function $\nu_{\mathcal{A}}$, such that

$$\Pr[f(z) = y | y = f(x), z = \mathcal{A}(y)] < \nu_{\mathcal{A}}(\log |\mathcal{X}|) ,$$

where the probability is taken over $x \in \mathcal{X}$ chosen uniformly at random. In other words, for every PPT (in $\log |\mathcal{X}|$) algorithm \mathcal{A} , the advantage to invert f is negligible.

Remark 2. When we are dealing with complexities, we consider sets in their bit representation as a computer would do. For instance, in the above definition, we can see the function f as $f: \{0, 1\}^n \rightarrow \{0, 1\}^m$ for $m, n \in \mathbb{N}$. This means that, f is one-way if one can compute $f(x)$ in $\text{poly}(n)$ time and if there is no PPT algorithm that can find a preimage in $\text{poly}(n)$ time.

Definition 2 (Elliptic curve based OWF). A one-way function whose domain is a subgroup $\mathbb{G} \subseteq E(\mathbb{F}_p)$ is called elliptic curve based OWF if its output does not depend on the short Weierstrass equation representing the isomorphism class² of E .

Definition 3 (maj_g). Given a boolean function $g: \mathcal{X} \rightarrow \{a_1, a_2\}$, we define

$$\text{maj}_g := \max_{b \in \{a_1, a_2\}} \Pr_{x \in_U \mathcal{X}}[g(x) = b] .$$

In other words, maj_g is the probability of the most probable of the two outcomes. This notion is useful when we deal with biased predicates.

¹ Confusing the two notions seems to be a common mistake in the cryptographic literature.

² Here, by an isomorphism class of E we really mean the \mathbb{F}_p -isomorphism class.

Remark 3. For the rest of the paper, we will be using the majority values of the predicates B_k on \mathbb{F}_p that return the k th least significant bit. If $x \in \mathbb{F}_p$ is viewed as an element of $[0, p - 1]$ then we will be using $\delta_p(k) := \text{maj}_{B_k}$ for the probability of occurrence of the majority value.

Definition 4 (Efficiently computable predicate). A boolean predicate $P: \mathcal{X} \rightarrow \{0, 1\}$ is efficiently computable with respect to a one-way function $f: \mathcal{X} \rightarrow \mathcal{Y}$ if there exists a PPT (in $\log |\mathcal{X}|$) algorithm \mathcal{A} that can compute $P(x)$ from $f(x)$ with a non-negligible advantage over the majority value, i.e., such that

$$\Pr_{x \in_U \mathcal{X}} [\mathcal{A}(f(x)) = P(x)] \geq \text{maj}_P + \frac{1}{\text{poly}(\log |\mathcal{X}|)},$$

for some polynomial that is independent of $|\mathcal{X}|$.

Definition 5 (Hard-to-compute predicate). A boolean predicate P is hard-to-compute with respect to a one-way function f if it is not efficiently computable.

Remark 4. Note that often, the term *hard-core* predicate is misused for *hard-to-compute* predicate. In this paper, we never use the term hard-core predicate (which, to the best of our knowledge, means that every algorithm that predicts P has negligible advantage over a random guessing; the latter is a strong definition and is not suitable for computational purposes). The difference between hard-core and hard-to-compute comes from the difference between a non-negligible function and a function that is not negligible as showed before.

2.2 Main Result on Hard-to-Compute Bits

Throughout, we consider bits that take values in $\{\pm 1\}$ instead of $\{0, 1\}$ where we substitute -1 for 1 and 1 for 0 , i.e., the new value is $(-1)^b$ with $b \in \{0, 1\}$. For a prime field \mathbb{F}_p , let $B_k: \mathbb{F}_p \rightarrow \{\pm 1\}$ be the predicate returning the value of the k th bit of $x \in \mathbb{F}_p$ viewed as an integer in $\{0, 1, \dots, p - 1\}$. Suppose that E is an elliptic curve over \mathbb{F}_p and $\mathbb{G} \subseteq E(\mathbb{F}_p)$ is a subgroup of prime order n of cryptographically meaningful size (i.e., $n = \Theta(p)$).

Let $R \in \mathbb{G}$ be a hidden point, let \mathcal{Y} be any set and let $f: \mathbb{G} \rightarrow \mathcal{Y}$ be a one-way function. Suppose that we have an imperfect oracle \mathcal{U}_k that takes as input a short Weierstrass equation $W \in \mathcal{W}(E)$ and the value $f(R) \in \mathcal{Y}$ and predicts $B_k((R_W)_x)$ with non-negligible advantage over the majority value $\delta_p(k)$ (here, the advantage is taken over a random short Weierstrass equations $W \in \mathcal{W}(E)$ and over a random point $R \in \mathbb{G}$). Then, we show that we can efficiently invert f .

Before we state precisely the theorem, we rigorously define the advantage of the bit-prediction oracle \mathcal{U}_k :

Definition 6 (Advantage). We say that \mathcal{U}_k has advantage ϵ in predicting the predicate B_k of the x -coordinate of the input $R \in \mathbb{G}$ of the one way function f if

$$\text{Adv}_f^{x,k}(\mathcal{U}_k) := \left| \Pr_{\substack{W \in_U \mathcal{W}(E) \\ R \in_U \mathbb{G}, z}} [\mathcal{U}_k(W, f(R); z) = B_k((R_W)_x)] - \delta_p(k) \right| > \epsilon,$$

where z is a random variable corresponding to the random coins used by the oracle \mathcal{U}_k . Similarly, we define the advantage $\text{Adv}_f^{y,k}(\mathcal{U}_k)$ of predicting the k th bit of the y -coordinate of the input point R to f .

We are now ready to state the main theorem:

Theorem 1. Let $k \geq 0$ be an integer and let $\epsilon \in (0, 1)$. Let E and \mathbb{G} be as above (i.e., \mathbb{G} is a subgroup of prime order $n = \Theta(p)$). Let \mathcal{Y} be any set and let $f: \mathbb{G} \rightarrow \mathcal{Y}$ be an elliptic curve based one-way function on E . Let $\mathcal{U}_k(W, v; z)$ be an algorithm that takes as input $W \in \mathcal{W}(E)$, $v \in \mathcal{Y}$ and outputs an element of $\{\pm 1\}$ in time T . Assume that $\text{Adv}_f^{x,k}(\mathcal{U}_k) > \epsilon$. Then, there exists an algorithm \mathcal{A} that inverts $f: \mathbb{G} \rightarrow \mathcal{Y}$ in time $T \cdot \text{poly}(\log p, \frac{1}{\epsilon})$ for some polynomial that is independent of p , E , \mathbb{G} , and ϵ .

Remark 5. One can see from the above theorem that if $1/\epsilon = \text{poly}(\log p)$ and if $T = \text{poly}(\log p)$ then one can invert the function f efficiently (in time polynomial in $\log p$). This means that either the k th bit of the input to f is hard-to-compute or that the function is invertible.

Remark 6. Note that our result is on average as our argument exploits in an essential way the freedom to change the short Weierstrass equation. This is why we assume that algorithm \mathcal{U}_k works on a non-negligible fraction of all the short Weierstrass equations W . Ideally, one wishes to fix a short Weierstrass equation W and prove similar hardness result only on W . This last question appears to be very difficult and out of reach with the current techniques that one has so far for showing hardness of bits (see also [6] for a similar remark regarding hardness of computing the least significant bit for Diffie–Hellman secrets).

3 Hard-to-Compute Predicates via List-Decoding

3.1 Fourier Transforms

In order to describe the general method of Akavia–Goldwasser–Safra, we briefly recall some basic notions related to Fourier transforms.

Let G be a finite abelian group. If $f, g: G \rightarrow \mathbb{C}$ are functions then their *inner product* is defined as $\langle f, g \rangle := 1/|G| \sum_{x \in G} \overline{f(x)}g(x)$. The ℓ_2 -norm on the space $\mathbb{C}(G)$ of all complex valued functions $f: G \rightarrow \mathbb{C}$ is then $\|f\|_2 := \sqrt{\langle f, f \rangle}$. A *character* of G is a homomorphism $\chi: G \rightarrow \mathbb{C}^\times$, i.e., $\chi(x+y) = \chi(x)\chi(y)$ for all $x, y \in G$. The set of all characters of G forms a group \widehat{G} , the character group. The elements of \widehat{G} form an orthonormal basis for the space $\mathbb{C}(G)$ (the *Fourier basis*). One can then describe a function $f \in \mathbb{C}(G)$ via its *Fourier expansion* $\sum_{\chi \in \widehat{G}} \langle f, \chi \rangle \chi$. Equivalently, one can define the Fourier transform $\widehat{f}: \widehat{G} \rightarrow \mathbb{C}$ of f by $\widehat{f}(\chi) = \langle f, \chi \rangle$. The coefficients $\widehat{f}(\chi)$ in the Fourier basis $\{\chi\}_{\chi \in \widehat{G}}$ are the *Fourier coefficients* of f . When $G = \mathbb{Z}/n\mathbb{Z}$, the characters of G are defined by $\chi_\alpha(x) := \omega_n^{\alpha x}$, for $\alpha \in \mathbb{Z}/n\mathbb{Z}$ and $\omega_n := \exp(2\pi i/n)$. The *weight* of a Fourier coefficient $\widehat{f}(\chi)$ is $|\widehat{f}(\chi)|^2$. Using these definition, we can define *heavy characters* with respect to a function f :

Definition 7 (Heavy characters). Given a function $f: G \rightarrow \mathbb{C}$ and a threshold τ , we denote by $\text{Heavy}_\tau(f)$ the set of characters for which the weight of the corresponding Fourier coefficient of f is at least τ . In other words,

$$\text{Heavy}_\tau(f) := \{\chi \in \widehat{G} : |\widehat{f}(\chi)|^2 \geq \tau\}.$$

We will frequently approximate a function $f \in \mathbb{C}(G)$ using subsets $\Gamma \subset \widehat{G}$ of characters via its restriction: $f|_\Gamma := \sum_{\chi \in \Gamma} \widehat{f}(\chi)\chi$.

3.2 Codes, Fourier Concentration and Recoverability

When working on an abelian group G , we consider binary codewords of length $|G|$. Every codeword corresponding to an element $x \in G$ will be represented by a function $C_x: G \rightarrow \{\pm 1\}$. If $G = \mathbb{Z}/n\mathbb{Z}$ then C_x is represented by $(C_x(0), C_x(1), \dots, C_x(n-1))$. We define now two properties of codes we will use in our proof.

Definition 8 (Concentration). Let $\epsilon > 0$ be a real number. A function $f: G \rightarrow \{\pm 1\}$ is called Fourier ϵ -concentrated if there exists a set of characters $\Gamma \subseteq \widehat{G}$ of size $\text{poly}(\log |G|, 1/\epsilon)$ (for a polynomial that does not depend on $|G|$, ϵ or the function f) such that $\|f - f|_\Gamma\|_2 \leq \epsilon$.

A code $\mathcal{C} = \{C_x: G \rightarrow \{\pm 1\}\}$ is ϵ -concentrated if each of its codewords C_x is Fourier ϵ -concentrated. In other words, we can approximate with an error at most ϵ every codeword using a polynomial number (in $\log |G|$ and $1/\epsilon$) of characters $\chi \in \widehat{G}$. A function is called Fourier concentrated if it is ϵ -concentrated for every $\epsilon > 0$. A code is called Fourier concentrated if all of its codewords are Fourier concentrated.

Definition 9 (Recoverable code). A code $\mathcal{C} = \{C_x: G \rightarrow \{\pm 1\}\}$ is recoverable if there exists an algorithm that takes as input a character $\chi \in \widehat{G}$ and a threshold τ and outputs (in time polynomial in $\log |G|$ and $1/\tau$) the list $\{x \in G: \chi \in \text{Heavy}_\tau(C_x)\}$ of all codewords having χ as a τ -heavy coefficient.

Using the orthogonality of the characters $\chi \in \widehat{G}$, one shows [2, Lem.1] that if a code \mathcal{C} is concentrated, then a word $w_x: G \rightarrow \mathbb{C}$ and a close codeword C_x have at least one heavy Fourier coefficient in common. We show here a slight modification of this lemma.

Lemma 1 ([2, Lem.1]). Let $f: \mathbb{Z}/n\mathbb{Z} \rightarrow \{\pm 1\}$ be a Fourier concentrated function and let $g: \mathbb{Z}/n\mathbb{Z} \rightarrow \{\pm 1\}$ such that

$$\Pr_{x \in \mathbb{Z}/n\mathbb{Z}} [f(x) = g(x)] \geq \text{maj}_f + \epsilon, \quad (1)$$

for some $\epsilon > 0$. Then there exists a threshold τ such that $1/\tau$ is polynomial in $1/\epsilon$ and $\log n$, and $\exists \chi \neq 0, \chi \in \text{Heavy}_\tau(f) \cap \text{Heavy}_\tau(g)$.

We omit the proof since it is straightforward from the proof in [2].

In Section 4, we will apply this lemma in the following way: every preimage $x \in G$ of the one-way function we try to invert corresponds to a codeword C_x .

First, we recover a noisy version w_x of C_x by using the prediction oracle. If the code is concentrated, the words w_x and C_x share at least one heavy coefficient. Thus, if we can compute this heavy coefficient in polynomial time and if the code is recoverable, then we can recover x in polynomial time.

One recovers the heavy coefficient using the following theorem:

Theorem 2 ([2, Thm.6]). *There exists a randomized learning algorithm over $\mathbb{Z}/n\mathbb{Z}$ that, given a function $w: \mathbb{Z}/n\mathbb{Z} \rightarrow \{\pm 1\}$, $0 < \tau$ and $0 < \delta < 1$, returns a list of $\mathcal{O}(1/\tau)$ characters containing $\text{Heavy}_\tau(w)$ with probability at least $1 - \delta$ (here, the probability is taken over the random coins of the algorithm) and that has running time³*

$$\tilde{\mathcal{O}}\left(\log(n) \ln^2 \frac{(1/\delta)}{\tau^{5.5}}\right).$$

For completeness, we recall the algorithm in the full version of the paper [10].

Remark 7. In the language of Akavia et al. [2, §2.3], $\mathbb{Z}/n\mathbb{Z}$ is a *learnable domain*. It turns out that any finite abelian group G is a learnable domain [1].

4 Proof of Theorem 1

We will reduce the proof of Theorem 1 to a list-decoding problem that will be solved using Akavia et al.'s method. The first step is to properly define a code that reflects our input recovery problem. We explain in Section 4.1 that the straightforward definition of such a code does not quite work since the Fourier transforms of the codewords are difficult to analyze from the point of view of concentration and recoverability. In order to overcome this difficulty, we use an idea motivated by the work of Boneh and Shparlinski on the Hidden Number Problem that modifies the prediction oracle via extra randomization while still keeping the non-negligible advantage. This leads us to the definition of the Elliptic Curve Multiplication Code (ECMC) (Definition 10).

4.1 The Elliptic Curve Multiplication Code (ECMC)

Let $B_k: \mathbb{F}_p \rightarrow \{\pm 1\}$ be the binary predicate that returns 1 if the k th least significant bit of the argument is 0 and -1 otherwise. A natural way to associate a code to this predicate is to fix a (base) short Weierstrass equation $W \in \mathcal{W}(E)$ and a hidden point R and define the codewords

$$C_R^{B_k, W}: \mathbb{F}_p \rightarrow \{\pm 1\}, \quad C_R^{B_k, W}(\lambda) = B_k(\lambda^2 \cdot (R_W)_x) = B_k((R_{W_\lambda})_x).$$

The above definition is natural since the isomorphism class $\mathcal{W}(E)$ of short Weierstrass equations consists precisely of the equations W_λ where $\lambda \in \mathbb{F}_p^\times$. So, each codeword encodes the k th bit of all representations of the point $R \in \mathbb{G}$ on the equations from $\mathcal{W}(E)$. In order to study how concentrated these codes are, one

³ A function is $\tilde{\mathcal{O}}(f(n))$ if it is $\mathcal{O}(f(n) \cdot \log(f(n))^k)$ for some $k \in \mathbb{N}$.

needs precise estimates of the Fourier coefficients of these functions. Yet, the only tool we are aware of that gives such estimates are standard estimates from analytic number theory on Gauss sums (see full version of the paper).

Unfortunately, these are not sufficient to get any information about how concentrated the code is. If one is able to replace the square term λ^2 with a linear term in λ , one could obtain a much better control on the code. As mentioned above, we use an idea of Boneh and Shparlinski [6, §5] that modifies the prediction oracle via further randomization while keeping the advantage non-negligible.

The idea works as follows: suppose that \mathcal{U}_k is the prediction oracle from the statement of Theorem 1. Recall that given a hidden point $R \in \mathbb{G}$, the oracle returns an element of $\{\pm 1\}$ in such a way that $\text{Adv}_f^{x,k}(\mathcal{U}_k) > \epsilon$.

If $\mathbb{F}_p^2 \subset \mathbb{F}_p$ is the set of squares in \mathbb{F}_p , let $r: \mathbb{F}_p^2 \rightarrow \mathbb{F}_p$ be a function satisfying $r(\lambda)^2 = \lambda$ that is chosen uniformly at random among all such functions. The observation of Boneh and Shparlinski is that one can define an auxiliary prediction oracle \mathcal{U}'_k using \mathcal{U}_k as follows:

$$\mathcal{U}'_k(W_\lambda, f(R); z) = \begin{cases} \mathcal{U}_k(W_{r(\lambda)}, f(R); z) & \text{if } \lambda \in \mathbb{F}_p^\times \text{ is a square in } \mathbb{F}_p^\times \\ \arg \max_{b \in \{\pm 1\}} \Pr_{x \in U \mathbb{F}_p} [B_k(x) = b] & \text{otherwise} \end{cases}.$$

Hence, if λ is not a square, the oracle returns the most common value which is the best random strategy to guess B_k . We now associate a code to the modified oracle \mathcal{U}'_k rather than to the original oracle \mathcal{U}_k and thus, arrive at the following definition (we include the more general case of binary predicates that are not necessarily the predicates B_k):

Definition 10 (Elliptic curve multiplication code (ECMC)). Let E be an elliptic curve over \mathbb{F}_p and let $P: \mathbb{F}_p \rightarrow \{\pm 1\}$ be a binary predicate. Let $\mathbb{G} \subset E(\mathbb{F}_p)$ be a cyclic subgroup. Given a (base) short Weierstrass equation $W: y^2 = x^3 + ax + b$ representing E , the elliptic curve multiplication code is the code $\mathcal{C}^{P,W} = \{C_R^{P,W}: \mathbb{F}_p \rightarrow \{\pm 1\}\}_{R \in \mathbb{G}}$ defined by

$$C_R^{P,W}(\lambda) = P(\lambda \cdot (R_W)_x),$$

where R_W denotes the tuple (x, y) representing the point R on W .

Remark 8. Reducing the quadratic term λ^2 with λ is a big advantage since (as we will show in Section 4.2), the Fourier transform of $B_k(\lambda)$ is simpler to analyze for the purpose of studying heavy coefficients than the Fourier transform of $B_k(\lambda^2)$. This makes it easier to show that the code $\mathcal{C}^{B_k,W}$ is Fourier concentrated and recoverable and, thus, apply the techniques of Akavia et al. to obtain a list-decoding algorithm.

Lemma 2. Let $W \in \mathcal{W}(E)$ be a fixed (base) short Weierstrass equation and let \mathcal{U}_k be the prediction algorithm from the statement of Theorem 1. There exists a set S of points $R \in \mathbb{G}$ satisfying

$$|S| \geq \frac{\epsilon}{4(1 - \delta_p(k) - \frac{\epsilon}{4})} |\mathbb{G}| \tag{2}$$

such that for every $R \in S$, given $f_Q(R)$, we have access to a corrupted codeword $w_{R,W}$ such that

$$\Pr_{\lambda \in U \mathbb{F}_p} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] \geq \delta_p(k) + \frac{\epsilon}{4}, \quad \forall R \in S. \quad (3)$$

Proof. Recall that our prediction algorithm \mathcal{U}_k satisfies:

$$\Pr_{W_i, R; z} [\mathcal{U}_k(W_i, f(R); z) = B_k((R_{W_i})_x)] > \delta_p(k) + \epsilon. \quad (4)$$

The latter is equivalent to

$$\Pr_{\lambda, R; z} [\mathcal{U}_k(W_\lambda, f(R); z) = B_k(\lambda^2 \cdot (R_W)_x)] > \delta_p(k) + \epsilon. \quad (5)$$

Given a hidden point $R \in \mathbb{G}$, define $w_{R,W}$ as follows:

$$w_{R,W}(\lambda) = \begin{cases} \mathcal{U}_k(W_{r(\lambda)}, f(R); z) & \text{if } \lambda \text{ is a square} \\ \arg \max_{b \in \{\pm 1\}} \Pr_{x \in U \mathbb{F}_p} [B_k(x) = b] & \text{otherwise,} \end{cases}$$

where $r: \mathbb{F}_p^2 \rightarrow \mathbb{F}_p$ is chosen uniformly at random among all function $r: \mathbb{F}_p^2 \rightarrow \mathbb{F}_p$ satisfying $r(\lambda)^2 = \lambda$ and where z is the random coin used by \mathcal{U}_k . Using the randomness of r , we estimate

$$\begin{aligned} \Pr_{\lambda, R; z} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] &= \\ &= \frac{1}{2} \Pr_{\substack{\lambda \in U \mathbb{F}_p^2 \\ R; z}} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] + \frac{1}{2} \Pr_{\substack{\lambda \notin \mathbb{F}_p^2 \\ R; z}} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] \\ &= \frac{1}{2} \Pr_{\substack{\lambda' \in U \mathbb{F}_p \\ R; z}} [\mathcal{U}_k(W_{\lambda'}, f(R); z) = B_k(\lambda'^2 \cdot (R_W))] + \frac{1}{2} \delta_p(k) \\ &> \frac{1}{2} (\delta_p(k) + \epsilon) + \frac{1}{2} \delta_p(k) = \delta_p(k) + \frac{\epsilon}{2}. \end{aligned} \quad (6)$$

Next, let $S \subseteq \mathbb{G}$ be the subset of all points $R \in \mathbb{G}$ that satisfy

$$\Pr_{\lambda; z} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] > \delta_p(k) + \frac{\epsilon}{4}.$$

Points in this set satisfy (3). We now show that the set S satisfies (2). Using (6), we arrive at

$$\begin{aligned} \delta_p(k) + \frac{\epsilon}{2} &< \frac{1}{|\mathbb{G}|} \sum_{R \in \mathbb{G}} \Pr_{\lambda; z} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] \\ &= \frac{1}{|\mathbb{G}|} \left(\sum_{R \in S} \Pr_{\lambda; z} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] + \sum_{R \in \mathbb{G} \setminus S} \Pr_{\lambda; z} [w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] \right) \\ &< \frac{1}{|\mathbb{G}|} \left(|S| + |\mathbb{G} \setminus S| \left(\delta_p(k) + \frac{\epsilon}{4} \right) \right) = \frac{|S|}{|\mathbb{G}|} \left(1 - \delta_p(k) - \frac{\epsilon}{4} \right) + \left(\delta_p(k) + \frac{\epsilon}{4} \right). \end{aligned}$$

Since $\delta_p(k) \neq 1$, we obtain (2). \square

Remark 9. If $1/\epsilon = \text{poly}(\log p)$ then the above lemma tells us that the k th bit is predictable with non-negligible advantage over a random guess for a polynomial fraction of all the points $R \in \mathbb{G}$.

In the next section, we explain in more detail the two major properties of the ECMC associated to the k th bit predicates, namely, Fourier concentration and recoverability. This is done via the methods developed in [27].

4.2 Fourier Concentration of ECMC

In order to gain more control on the size of the Fourier coefficients $\widehat{B}_k(\alpha)$, and thus, be able to pick the heavy ones, we use another idea of Morillo and Ràfols: since p is odd, we can assume that $\alpha \in \left[-\frac{p-1}{2}, \frac{p-1}{2}\right]$. Consider the following two cases for α :

- When $\alpha \geq 0$, we consider $\delta_{\alpha,k} := 2^k\alpha - (p-1)/2 \bmod p$ and let $\lambda_{\alpha,k} \in [0, 2^{k-1} - 1]$ be the unique integer for which $2^k\alpha = (p-1)/2 + \delta_{\alpha,k} + p\lambda_{\alpha,k}$.
- When $\alpha < 0$, we consider $\delta_{\alpha,k} = 2^k\alpha + (p+1)/2 \bmod p$ and let $\lambda_{\alpha,k} \in [0, 2^{k-1} - 1]$ be the unique integer for which $2^k\alpha = -(p+1)/2 + \delta_{\alpha,k} + p\lambda_{\alpha,k}$.

For both of the cases, there are unique integers $\mu_{\alpha,k} \in [0, r]$ and $r_{\alpha,k} \in [0, 2^k - 1]$ such that $a_p(\alpha 2^k - (p-1)/2) = \mu_{\alpha,k} 2^k + r_{\alpha,k}$, where $a_p(x) = \min(x \bmod p, p - x \bmod p)$ for $y \bmod p$ being taken in $[0, p-1]$. From here, one characterizes (see full version of the paper) the asymptotic behavior of $|\widehat{B}_k(\alpha)|$ by $|\widehat{B}_k(\alpha)|^2 < \mathcal{O}\left(1/(\lambda_{\alpha,k}^2 \mu_{\alpha,k}^2)\right)$.

The idea of having the above representation $(\lambda_{\alpha,k}, \mu_{\alpha,k})$ is that it is very convenient for picking the heavy Fourier coefficients: one simply has to pick the coefficients α for which $(\lambda_{\alpha,k}, \mu_{\alpha,k})$ is in a box $[0, 1/\tau] \times [0, 1/\tau]$ for $\tau = \text{poly}(\log p)$.

4.3 Recoverability of ECMC and End of Proof

Fix a short Weierstrass equation $W \in \mathcal{W}(E)$. According to Lemma 2, there exists a subset $S \subset \mathbb{G}$ of size determined by (2) and the property that for any $R' \in S$, we have access to a corrupted codeword $w_{R',W}$ satisfying (3). The problem is that our hidden point $R \in \mathbb{G}$ need not be in S . In order to remedy this, we repeat the following procedure: we pick a random multiple $s \in [1, n-1]$ and set $R' = sR$. Note that when s is invertible modulo n , knowing R' is equivalent to knowing R . Thus, if s is chosen uniformly at random, we have $1/\text{poly}(\log p)$ -chance of obtaining R' in the set S .

Suppose for the moment that R' happens to be in S . One can then use Lemma 1 to deduce that there exists $0 < \tau < 1$ for which $1/\tau$ is polynomial in $\log p$ and $1/\epsilon$ such that the noisy codeword $w_{W,R'}$ and the actual codeword $C_{R'}^{B_k,W}$ share a τ -heavy Fourier coefficient. Then, we apply the learning algorithm of Akavia et al. (Theorem 2) to efficiently compute all τ -heavy Fourier

characters χ_β for the noisy codeword $w_{R',W}$. We then run the recovery algorithm (Algorithm 1) for each of these τ -heavy Fourier coefficients to decode the hidden R' and thus, obtain the possible R 's by computing $s^{-1}R'$. Assuming that $w_{R',W}$ is Fourier concentrated, we only have to run this algorithm $\text{poly}(\log p)$ times, so we get a polynomial time (in $\log p$ and $1/\epsilon$) recovery procedure for R' .

Notice that we have no way of knowing whether $R' \in S$ unless we try to recover the point via the above recovery procedure. Yet, by using a random choice of $s \in [1, n - 1]$ and repeating the procedure $\text{poly}(\log p)$ times, we obtain (with high probability) a point R' in the set S guaranteed by Lemma 2 and thus, prove Theorem 1.

The method used in our proof is close to the list-decoding method of Akavia et al. [2, Lem. 5] and was successfully used by Morillo and Ràfols [27, §6]. The reason it works is that the codeword $C_R^{B_k,W}$ is τ -concentrated in $\Gamma_{R,W} = \{\chi_\beta : \beta \equiv \alpha \cdot (R_W)_x \pmod{p}, \chi_\alpha \in \Gamma\}$ where Γ is the set of additive characters $\chi_\alpha : \mathbb{F}_p \rightarrow \mathbb{C}^\times$ where $(\lambda_{\alpha,k}, \mu_{\alpha,k})$ is in a small square of size $\mathcal{O}(1/\tau)$ and lower-right corner at $(0, 0)$, i.e., $\Gamma = \{\chi_\alpha : \lambda_{\alpha,k} = \mathcal{O}(1/\tau), \mu_{\alpha,k} = \mathcal{O}(1/\tau)\}$. Here, we will take τ such that $1/\tau = \text{poly}(\log p)$.

Algorithm 1. The recovery algorithm

Input: An additive character χ_β of \mathbb{F}_p , a threshold parameter τ with $1/\tau \in \text{poly}(\log p)$ and $z \in \mathcal{Y}$ such that $z = f(R)$ for a hidden point R .

Output: The hidden point $R \in \mathbb{G}$ such that $f(R) = z$.

- 1: Calculate $\Gamma \leftarrow \{\alpha \in \mathbb{F}_p : \lambda_{\alpha,k} = \mathcal{O}(1/\tau), \mu_{\alpha,k} = \mathcal{O}(1/\tau)\}$.
- 2: **for** $\alpha \in \Gamma \setminus \{0\}$ **do**
- 3: Compute $x \leftarrow \beta\alpha^{-1} \pmod{p}$
- 4: **if** $y \in \mathbb{F}_p$ exists so that $R = (x, y) \in W(\mathbb{F}_p)$ and $f(R) = z$ **then**
- 5: **return** R
- 6: **end if**
- 7: **end for**
- 8: **return** false.

The above algorithm works in time polynomial in $\log p$ because 1) the algorithm from Theorem 2 works in polynomial time (in $\log p$); 2) the set S from Lemma 2 is a polynomial fraction of all points in \mathbb{G} and hence, a randomly chosen multiple will be recoverable with probability $1/\text{poly}(\log p)$ (so, we need on average $\text{poly}(\log p)$ trials to exit the **repeat** loop). This completes the proof of Theorem 1.

5 Application to Pairing-Based One-Way Functions

5.1 Pairing-Based One-Way Functions

We define now a pairing-based one-way function. For an prime n , let $E[n]$ be the subgroup of points of E of order n (the points in $E[n]$ are defined over the algebraic closure $\overline{\mathbb{F}}_p$ of \mathbb{F}_p). Let k be the smallest integer for which $n \mid p^k - 1$ (also

Algorithm 2. Elliptic curve-based OWF inversion algorithm

Input: An elliptic curve E/\mathbb{F}_p , a subgroup $\mathbb{G} \subset E$ of prime order $n = \Theta(p)$, an element $z = f(R) \in \mathcal{Y}$ for a hidden point $R \in \mathbb{G}$ and access to a (noisy) prediction oracle \mathcal{U}_k for $B_k((R_W)_x)$ for $W \in \mathcal{W}(E)$.

Output: An input point $R \in \mathbb{G}$ such that $f(R) = z$.

- 1: Fix a (base) short Weierstrass equation $W \in \mathcal{W}(E)$.
- 2: Choose τ such that $1/\tau = \text{poly}(\log p)$
- 3: Choose a random function $r: \mathbb{F}_p^2 \rightarrow \mathbb{F}_p$.
- 4: **repeat**
- 5: Choose a random $s \in [1, n - 1]$ and set $R' \leftarrow sR$
- 6: Apply the algorithm of Theorem 2 to compute $\text{Heavy}_\tau(w_{R', W})$ for the function (noisy codeword) $w_{R', W}$ from Lemma 2 defined via r and \mathcal{U}_k .
- 7: **for** $\chi_\beta \in \text{Heavy}_\tau(w_{R', W})$ **do**
- 8: Run Algorithm 1 with $z' \leftarrow z^s$ to try to recover R'
- 9: **if** Algorithm 1 does not fail **then**
- 10: **break**
- 11: **end if**
- 12: **end for**
- 13: **until** R' is recovered
- 14: **return** $R \leftarrow s^{-1}R'$

known as the *embedding degree*) and let μ_n be the subgroup of order n of $\mathbb{F}_{p^k}^\times$. Let $e: E[n] \times E[n] \rightarrow \mu_n$ be a bilinear pairing, e.g., the Tate or the Weil pairing. Let $\mathbb{G} := \langle S \rangle$ for an $S \in E(\mathbb{F}_p)$. To avoid having $e(R, Q) = 1$ for all $R, Q \in \mathbb{G}$, we need to suitably twist e and define what we refer to as a *cryptographic pairing*:

Definition 11 (Cryptographic pairing). Let $\xi: E \rightarrow E$ be a non-trivial endomorphism defined over an extension field of \mathbb{F}_p (ξ is often referred to as a *distortion map*). We define the cryptographic pairing $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mu_n$ as

$$\hat{e}(R, Q) = e(R, \xi(Q)) , \quad R, Q \in \mathbb{G} .$$

Here, if \mathbb{G} is a cyclic subgroup and if $R, Q \in \mathbb{G}$ then $e(R, Q)$ will be trivial since e is bilinear and alternating. The role of the endomorphism ξ is to *distort* Q in such a way that $e(R, \xi(Q)) \neq 1$.

A typical example of a cryptographic pairing (see [5]) is a twisted version of the Weil pairing. More precisely, let $p \equiv 2 \pmod{3}$ and $q > 3$ be two primes such that q divides $p-1$ and let E be the elliptic curve over \mathbb{F}_p defined by $y^2 = x^3 + 1$. Let \mathbb{G} be the cyclic group generated by a random $S \in E(\mathbb{F}_p)$ of order q . The distortion map is defined as $\xi(Q_x, Q_y) = (\zeta Q_x, Q_y)$, for $\zeta \in \mathbb{F}_{p^2}$, $\zeta \notin \mathbb{F}_p$ such that $\zeta^2 + \zeta + 1 = 0$ (such a ζ exists as long as $X^2 + X + 1$ has a zero in $\mathbb{F}_p[X]$ which is equivalent to $p \equiv 2 \pmod{3}$). One could think of ζ as distorting one of the points so that it is mapped to a point that is outside of the group \mathbb{G} and that is defined over a non-trivial extension of \mathbb{F}_p .

Definition 12 (Pairing-based one-way function). Let E be an elliptic curve over \mathbb{F}_p with Weierstrass equation $y^2 = x^3 + ax^2 + b$ and let $\mathbb{G} \subseteq E[n]$ be a cyclic subgroup. Let $Q \in \mathbb{G}$ be a fixed generator and let $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mu_n$ be a

cryptographic bilinear pairing. We define a function $f_Q: \mathbb{G} \rightarrow \mu_n$ by $f_Q(R) := \hat{e}(R, Q)$, for $Q \in \mathbb{G}$. The preimage R will often be referred to as a hidden point.

The function $f_Q(R)$ is believed to be one-way [11,23,5].

Obviously, we can apply Theorem 1 to f_Q :

Corollary 1. *Let $k \geq 0$ be an integer and let $\epsilon \in (0, 1)$. Let E , \mathbb{G} and Q be as above (i.e., \mathbb{G} is cyclic of order $n = \Theta(p)$ and $Q \in \mathbb{G}$ is a generator). Let $\mathcal{U}_k = \mathcal{U}_k(W, v; z)$ be an algorithm that takes as input $W \in \mathcal{W}(E)$, $v \in \mu_n$ and outputs an element of $\{\pm 1\}$ in time T . Assume that $\text{Adv}_{f_Q}^{x,k}(\mathcal{U}_k) > \epsilon$. Then there exists an algorithm \mathcal{A} that inverts $f_Q: \mathbb{G} \rightarrow \mu_n$ in time $T \cdot \text{poly}(\log p, \frac{1}{\epsilon})$ for some polynomial that is independent of p , E , \mathbb{G} , ϵ and Q .*

5.2 Consequences of Our Result

Corollary 1 implies that either every bit of the input of f_Q is hard-to-compute or that f_Q can be inverted efficiently, i.e., FAPI-2 is easy. The hardness of FAPI-2 has been related to various problems [11,24,23].

Definition 13 (BDH). *Let $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear pairing, let S be a generator of \mathbb{G} and let n be the order of \mathbb{G} . The Bilinear Diffie–Hellman problem (BDH) is the following problem: given $\langle S, aS, bS, cS \rangle$, $a, b, c \in \mathbb{Z}/n\mathbb{Z}$, compute $\hat{e}(S, S)^{abc}$.*

The following relations hold. The hardness of BDH implies the hardness of the computational Diffie–Hellman problem (CDH) in both \mathbb{G} and \mathbb{G}_T , which imply the hardness of FAPI-2. Recall that CDH in \mathbb{G} consists in computing abS given $\langle aS, bS, S \rangle$. The hardness of FAPI-2 implies also the hardness of the discrete logarithm in \mathbb{G}_T . Hence, our result implies that if we assume that CDH is hard in both groups, every bit of the input of f_Q is hard-to-compute. Many cryptographic schemes relies on the hardness of BDH or FAPI-2. We show what implications an easy FAPI-2 would have.

Boneh–Franklin’s Identity-Based Encryption Scheme. The security of this well-known scheme [5] relies on the hardness of BDH. If FAPI-2 is easy, then an adversary can recover the secret key of any user of the system. Recall that in IBE, the secret key is computed as $d_{ID} := sQ_{ID}$, where Q_{ID} is a point dependent on the identity of the owner of the key and s is the master key. Two points are also public parameters of the scheme: P , which is a generator of \mathbb{G} and $P_{pub} := sP$. Hence, $\hat{e}(P_{pub}, Q_{ID}) = \hat{e}(sQ_{ID}, P) = \hat{e}(d_{ID}, P)$ and using an inversion algorithm to invert f_P , one can recover the secret key of the user associated with ID . Note that if the algorithm is imperfect, one can easily add some randomness by trying to invert $\hat{e}(P_{pub}, Q_{ID})^r$ for a random r instead.

Hess’ Identity-Based Signature Scheme. In a similar fashion, it was shown in [11] that one can forge signatures in Hess’ identity-based signature scheme [19] if FAPI-2 is easy. In this scheme, let s be the master key, $U, V := sU$ be parameters

and h, H hash functions. A signature of a message m consists in a pair (u, v) where $v := h(m, r)$, $r := \hat{e}(R, U)^k$ for a random k , a random R and where $u := vS_{ID} + kR$, with $S_{ID} = sH(ID)$. The signature is verified if $r = \hat{e}(u, U) \cdot \hat{e}(H(ID), -V)^v$. To forge a signature, an adversary selects a random r and selects $v = h(m, r)$. Then, using the algorithm for f_U he inverts $r\hat{e}(H(ID), V)^v = \hat{e}(u, U)$.

Joux's Tripartite Protocol. In this scheme [23], three parties, A , B and C , pick two elements $Q, R \in \mathbb{G}$ such that $\hat{e}(Q, R) \neq 1$ and broadcast respectively (aQ, aR) , (bQ, bR) and (cQ, cR) in one round after which every party can compute the shared secret key $\hat{e}(Q, R)^{abc}$ (here, a, b and c are random secrets selected by A , B and C , respectively). Using an algorithm for f_R on $\hat{e}(aQ, bR)$, one can recover abQ . The shared secret key is then $\hat{e}(abQ, cR)$.

6 Conclusions

In conclusion, we proved that all the bits of elliptic curve based one-way functions are hard-to-compute. In particular, we proved that all the bits of the pairing-based one-way function are hard-to-compute assuming that CDH is hard. We proved our result for the x -coordinate of the point but the result can trivially be extended to the y -coordinate. In [2], the hardness result is proven for every *segment predicate* and not only for some particular bits. Intuitively, a segment predicate over $\mathbb{Z}/n\mathbb{Z}$ is a predicate that splits $\mathbb{Z}/n\mathbb{Z}$ in $\text{poly}(\log n)$ segments, or a multiplicative shift of it. Our work can be easily extended to prove the hardness of these predicates using the same ECMC code. There is another important aspect of bit security for the specific pairing-based one-way function to be studied: instead of considering a prediction oracle that works on an isomorphism class, we consider an imperfect oracle on an ordinary isogeny class of elliptic curves (i.e., elliptic curves with the same number of points) as was done in [22] for the least significant bits of the Diffie–Hellman secrets for elliptic curves. Note that using techniques based on isogeny graphs and rapid mixing of random walks [21], one can obtain a very strong conclusion for almost every isogeny class: namely, assuming that the oracle works with non-negligible advantage on a non-negligible fraction of all short Weierstrass equations in this class then one can solve FAPI-2 for *every* curve in this class. Proving such a result is the subject of a forthcoming paper.

Acknowledgments. We are grateful to Dan Boneh, David Freeman, Rosario Gennaro, Florian Hess, Eike Kiltz, Arjen Lenstra, Amin Shokrollahi, Igor Shparlinski, Martijn Stam, Serge Vaudenay and Ramarathnam Venkatesan for helpful discussions.

References

1. Akavia, A.: Learning Noisy Characters, Multiplication Codes, and Cryptographic Hardcore Predicates. Ph.D. thesis, Massachusetts Institute of Technology (2008)
2. Akavia, A., Goldwasser, S., Safra, S.: Proving Hard-Core Predicates Using List Decoding. In: FOCS, pp. 146–157. IEEE Computer Society (2003)
3. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.: RSA and Rabin Functions: Certain Parts are as Hard as the Whole. SIAM J. Comput. 17(2), 194–209 (1988)
4. Blum, M., Micali, S.: How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. SIAM J. Comput. 13(4), 850–864 (1984)
5. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian (ed.) [25], pp. 213–229
6. Boneh, D., Shparlinski, I.: On the Unpredictability of Bits of the Elliptic Curve Diffie–Hellman Scheme. In: Kilian (ed.) [25], pp. 201–212
7. Boneh, D., Venkatesan, R.: Hardness of Computing the Most Significant Bits of Secret Keys in Diffie–Hellman and Related Schemes. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (1996)
8. Boneh, D., Halevi, S., Howgrave-Graham, N.: The Modular Inversion Hidden Number Problem. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 36–51. Springer, Heidelberg (2001)
9. Boneh, D., Venkatesan, R.: Rounding in Lattices and its Cryptographic Applications. In: Saks, M.E. (ed.) SODA, pp. 675–681. ACM/SIAM (1997)
10. Duc, A., Jetchev, D.: Hardness of Computing Individual Bits for One-way Functions on Elliptic Curves (full version). Cryptology ePrint Archive, Report 2011/329 (2011), <http://eprint.iacr.org/>
11. Galbraith, S., Hess, F., Vercauteren, F.: Aspects of Pairing Inversion. IEEE Transactions on Information Theory 54(12), 5719–5728 (2008)
12. Galbraith, S.D., Hopkins, H.J., Shparlinski, I.E.: Secure Bilinear Diffie–Hellman Bits. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 370–378. Springer, Heidelberg (2004)
13. Goldmann, M., Näslund, M., Russell, A.: Complexity Bounds on General Hard-Core Predicates
14. Goldreich, O., Levin, L.: A Hard-Core Predicate for all One-Way Functions. In: STOC, pp. 25–32. ACM (1989)
15. Gonzalez Vasco, M.I., Shparlinski, I.: On the Security of Diffie–Hellman Bits. Electronic Colloquium on Computational Complexity (ECCC) 7(45) (2000)
16. Gonzalez Vasco, M.I., Shparlinski, I.: Security of the most significant bits of the shamir message passing scheme. Math. Comput. 71(237), 333–342 (2002)
17. Håstad, J., Näslund, M.: The Security of Individual RSA Bits. In: FOCS, pp. 510–521 (1998)
18. Håstad, J., Schrift, A., Shamir, A.: The Discrete Logarithm Modulo a Composite Hides $O(n)$ Bits. J. Comput. Syst. Sci. 47(3), 376–404 (1993)
19. Hess, F.: Efficient Identity Based Signature Schemes Based on Pairings. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
20. Howgrave-Graham, N., Nguyen, P.Q., Shparlinski, I.: Hidden number problem with hidden multipliers, timed-release crypto, and noisy exponentiation. Math. Comput. 72(243), 1473–1485 (2003)
21. Jao, D., Miller, S.D., Venkatesan, R.: Do All Elliptic Curves of the Same Order Have the Same Difficulty of Discrete Log? In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 21–40. Springer, Heidelberg (2005)

22. Jetchev, D., Venkatesan, R.: Bits Security of the Elliptic Curve Diffie–Hellman Secret Keys. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 75–92. Springer, Heidelberg (2008)
23. Joux, A.: A One Round Protocol for Tripartite Diffie–Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
24. Joux, A.: The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In: Fieker, C., Kohel, D.R. (eds.) ANTS 2002. LNCS, vol. 2369, pp. 20–32. Springer, Heidelberg (2002)
25. Kilian, J. (ed.): CRYPTO 2001. LNCS, vol. 2139. Springer, Heidelberg (2001)
26. Li, W.-C.W., Näslund, M., Shparlinski, I.E.: Hidden Number Problem with the Trace and Bit Security of XTR and LUC. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 433–448. Springer, Heidelberg (2002)
27. Morillo, P., Ràfols, C.: The Security of All Bits Using List Decoding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 15–33. Springer, Heidelberg (2009)
28. Nguyen, P.Q., Shparlinski, I.: The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. *J. Cryptology* 15(3), 151–176 (2002)
29. Nguyen, P.Q., Shparlinski, I.: The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. *Des. Codes Cryptography* 30(2), 201–217 (2003)
30. Nguyen, P.Q.: The dark side of the hidden number problem: Lattice attacks on DSA. In: Proc. Workshop on Cryptography and Computational Number Theory, pp. 321–330 (2001)
31. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
32. Rabin, M.: Digitalized signatures and public-key functions as intractable as factorization (1979)
33. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21(2), 120–126 (1978)
34. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairings. In: Proceedings of SCIS 2000, Okinawa, Japan (2000)
35. Schnorr, C.: Security of Almost ALL Discrete Log Bits. *Electronic Colloquium on Computational Complexity (ECCC)* 5(33) (1998)
36. Schrift, A., Shamir, A.: The Discrete Log is Very Discreet. In: STOC, pp. 405–415. ACM (1990)
37. Shparlinski, I.E.: On the Generalised Hidden Number Problem and Bit Security of XTR. In: Bozta, S., Shparlinski, I. (eds.) AAECC 2001. LNCS, vol. 2227, pp. 268–277. Springer, Heidelberg (2001)
38. Shparlinski, I., Winterhof, A.: A hidden number problem in small subgroups. *Math. Comput.* 74(252), 2073–2080 (2005)
39. Smart, N.P.: Identity-based authenticated key agreement protocol based on weil pairing. *Electronics Letters* 38(13), 630–632 (2002)

Homomorphic Evaluation of the AES Circuit

Craig Gentry¹, Shai Halevi¹, and Nigel P. Smart²

¹ IBM Research

² University of Bristol

Abstract. We describe a working implementation of leveled homomorphic encryption (without bootstrapping) that can evaluate the AES-128 circuit in three different ways. One variant takes under over 36 hours to evaluate an entire AES encryption operation, using NTL (over GMP) as our underlying software platform, and running on a large-memory machine. Using SIMD techniques, we can process over 54 blocks in each evaluation, yielding an amortized rate of just under 40 minutes per block. Another implementation takes just over two and a half days to evaluate the AES operation, but can process 720 blocks in each evaluation, yielding an amortized rate of just over five minutes per block. We also detail a third implementation, which theoretically could yield even better amortized complexity, but in practice turns out to be less competitive.

For our implementations we develop both AES-specific optimizations as well as several “generic” tools for FHE evaluation. These last tools include (among others) a different variant of the Brakerski-Vaikuntanathan key-switching technique that does not require reducing the norm of the ciphertext vector, and a method of implementing the Brakerski-Gentry-Vaikuntanathan modulus-switching transformation on ciphertexts in CRT representation.

1 Introduction

In his breakthrough result [9], Gentry demonstrated that fully-homomorphic encryption was theoretically possible, assuming the hardness of some problems in integer lattices. Since then, many different improvements have been made, for example authors have proposed new variants, improved efficiency, suggested other hardness assumptions, etc. Some of these works were accompanied by implementation [19,10,6,20,14,7], but all the implementations so far were either “proofs of concept” that can compute only one basic operation at a time (at great cost), or special-purpose implementations limited to evaluating very simple functions. In this work we report on the first implementation powerful enough to support an “interesting real world circuit”. Specifically, we implemented a variant of the leveled FHE-without-bootstrapping scheme of Brakerski, Gentry, and Vaikuntanathan [3] (BGV), with support for deep enough circuits so that we can evaluate an entire AES-128 encryption operation.

Why AES? We chose to shoot for an evaluation of AES since it seems like a natural benchmark: AES is widely deployed and used extensively in security-aware applications (so it is “practically relevant” to implement it), and the AES circuit is nontrivial on one hand, but on the other hand not astronomical. Moreover the AES circuit has a regular (and quite “algebraic”) structure , which is amenable to parallelism and

optimizations. Indeed, for these same reasons AES is often used as a benchmark for implementations of protocols for secure multi-party computation (MPC), for example [17,8,12,13]. Using the same yardstick to measure FHE and MPC protocols is quite natural, since these techniques target similar application domains and in some cases both techniques can be used to solve the same problem.

Beyond being a natural benchmark, homomorphic evaluation of AES decryption also has interesting applications: When data is encrypted under AES and we want to compute on that data, then homomorphic AES decryption would transform this AES-encrypted data into an FHE-encrypted data, and then we could perform whatever computation we wanted. (Such applications were alluded to in [14,20,4]).

Why BGV? Our implementation is based on the (ring-LWE-based) BGV cryptosystem [3], which at present is one of three variants that seem the most likely to yield “somewhat practical” homomorphic encryption. The other two are the NTRU-like cryptosystem of López-Alt et al. [15] and the ring-LWE-based fixed-modulus cryptosystem of Brakerski [2]. (These two variants were not yet available when we started our implementation effort.) These three different variants offer somewhat different implementation tradeoffs, but they all have similar performance characteristics. At present we do not know which of them will end up being faster in practice, but the differences are unlikely to be very significant. Moreover, we note that most of our optimizations for BGV are useful also for the other two variants.

Our Contributions. Our implementation is based on a variant of the BGV scheme [3,5,4] (based on ring-LWE [16]), using the techniques of Smart and Vercauteren (SV) [20] and Gentry, Halevi and Smart (GHS) [11], and we introduce many new optimizations. Some of our optimizations are specific to AES, these are described in Section 4. Most of our optimization, however, are more general-purpose and can be used for homomorphic evaluation of other circuits, these are described in Section 3.

Many of our general-purpose optimizations are aimed at reducing the number of FFTs and CRTs that we need to perform, by reducing the number of times that we need to convert polynomials between coefficient and evaluation representations. Since the cryptosystem is defined over a polynomial ring, many of the operations involve various manipulation of integer polynomials, such as modular multiplications and additions and Frobenius maps. Most of these operations can be performed more efficiently in evaluation representation, when a polynomial is represented by the vector of values that it assumes in all the roots of the ring polynomial (for example polynomial multiplication is just point-wise multiplication of the evaluation values). On the other hand some operations in BGV-type cryptosystems (such as key switching and modulus switching) seem to require coefficient representation, where a polynomial is represented by listing all its coefficients.¹ Hence a “naive implementation” of FHE would need to convert the polynomials back and forth between the two representations, and these conversions turn out to be the most time-consuming part of the execution. In our implementation we keep ciphertexts in evaluation representation at all times, converting to coefficient representation only when needed for some operation, and then converting back.

¹ The need for coefficient representation ultimately stems from the fact that the noise in the ciphertexts is small in coefficient representation but not in evaluation representation.

We describe variants of key switching and modulus switching that can be implemented while keeping almost all the polynomials in evaluation representation. Our key-switching variant has another advantage, in that it significantly reduces the size of the key-switching matrices in the public key. This is particularly important since the main limiting factor for evaluating deep circuits turns out to be the ability to keep the key-switching matrices in memory. Other optimizations that we present are meant to reduce the number of modulus switching and key switching operations that we need to do. This is done by tweaking some operations (such as multiplication by constant) to get a slower noise increase, by “batching” some operations before applying key switching, and by attaching to each ciphertext an estimate of the “noisiness” of this ciphertext, in order to support better noise bookkeeping.

Our Implementation. Our implementation was based on the NTL C++ library running over GMP, we utilized a machine which consisted of a processing unit of Intel Xeon CPUs running at 2.0 GHz with 18MB cache, and most importantly with 256GB of RAM.²

Memory was our main limiting factor in the implementation. With this machine it took us just under two days to compute a single block AES encryption using an implementation choice which minimizes the amount of memory required; this is roughly two orders of magnitude faster than what could be done with the Gentry-Halevi implementation [10]. The computation was performed on ciphertexts that could hold 864 plaintext slots each; where each slot holds an element of \mathbb{F}_{2^8} . This means that we can compute $\lfloor 864/16 \rfloor = 54$ AES operations in parallel, which gives an amortized time per block of roughly forty minutes. A second (byte-sliced) implementation, requiring more memory, completed an AES operation in around five days; where ciphertexts could hold 720 different \mathbb{F}_{2^8} slots (hence we can evaluate 720 blocks in parallel). This results in an amortized time per block of roughly five minutes.

We note that there are a multitude of optimizations that one can perform on our basic implementation. Most importantly, we believe that by using the “bootstrapping as optimization” technique from BGV [3] we can speedup the AES performance by an additional order of magnitude. Also, there are great gains to be had by making better use of parallelism: Unfortunately, the NTL library (which serves as our underlying software platform) is not thread safe, which severely limits our ability to utilize the multi-core functionality of modern processors (our test machine has 24 cores). We expect that by utilizing many threads we can speed up some of our (higher memory) AES variants by as much as a 16x factor; just by letting each thread compute a different S-box lookup.

Organization. In Section 2 we review the main features of BGV-type cryptosystems [4,3], and briefly survey the techniques for homomorphic computation on packed ciphertexts from SV and GHS [20,11]. Then in Section 3 we describe our “general-purpose” optimizations on a high level, with additional details provided in the full version of the paper. A brief overview of AES and a high-level description and performance numbers is provided in Section 4.

² This machine was BlueCrystal Phase 2; and the authors would like to thank the University of Bristol’s Advanced Computing Research Centre (<https://www.acrc.bris.ac.uk/>) for access to this facility.

2 Background

For an integer q we identify the ring $\mathbb{Z}/q\mathbb{Z}$ with the interval $(-q/2, q/2] \cap \mathbb{Z}$, and use $[z]_q$ to denote the reduction of the integer z modulo q into that interval. Our implementation utilizes polynomial rings defined by cyclotomic polynomials, $\mathbb{A} = \mathbb{Z}[X]/\Phi_m(X)$. The ring \mathbb{A} is the ring of integers of a the m th cyclotomic number field $\mathbb{Q}(\zeta_m)$. We let $\mathbb{A}_q \stackrel{\text{def}}{=} \mathbb{A}/q\mathbb{A} = \mathbb{Z}[X]/(\Phi_m(X), q)$ for the (possibly composite) integer q , and we identify \mathbb{A}_q with the set of integer polynomials of degree upto $\phi(m) - 1$ reduced modulo q .

Coefficient vs. Evaluation Representation. Let m, q be two integers such that $\mathbb{Z}/q\mathbb{Z}$ contains a primitive m -th root of unity, and denote one such primitive m -th root of unity by $\zeta \in \mathbb{Z}/q\mathbb{Z}$. Recall that the m 'th cyclotomic polynomial splits into linear terms modulo q , $\Phi_m(X) = \prod_{i \in (\mathbb{Z}/m\mathbb{Z})^*} (X - \zeta^i) \pmod{q}$.

We consider two ways of representing an element $a \in \mathbb{A}_q$: Viewing a as a degree- $(\phi(m) - 1)$ polynomial, $a(X) = \sum_{i < \phi(m)} a_i X^i$, the *coefficient representation* of a just lists all the coefficients in order $\mathbf{a} = \langle a_0, a_1, \dots, a_{\phi(m)-1} \rangle \in (\mathbb{Z}/q\mathbb{Z})^{\phi(m)}$. For the other representation we consider the values that the polynomial $a(X)$ assumes on all primitive m -th roots of unity modulo q , $b_i = a(\zeta^i) \pmod{q}$ for $i \in (\mathbb{Z}/m\mathbb{Z})^*$. The b_i 's in order also yield a vector $\mathbf{b} \in (\mathbb{Z}/q\mathbb{Z})^{\phi(m)}$, which we call the *evaluation representation* of a . Clearly these two representations are related via $\mathbf{b} = V_m \cdot \mathbf{a}$, where V_m is the Vandermonde matrix over the primitive m -th roots of unity modulo q . We remark that for all i we have the equality $(a \pmod{(X - \zeta^i)}) = a(\zeta^i) = b_i$, hence the evaluation representation of a is just a polynomial Chinese-Remaindering representation.

In both representations, an element $a \in \mathbb{A}_q$ is represented by a $\phi(m)$ -vector of integers in $\mathbb{Z}/q\mathbb{Z}$. If q is a composite then each of these integers can itself be represented either using the standard binary encoding of integers or using Chinese-Remaindering relative to the factors of q . We usually use the standard binary encoding for the coefficient representation and Chinese-Remaindering for the evaluation representation. (Hence the latter representation is really a *double CRT* representation, relative to both the polynomial factors of $\Phi_m(X)$ and the integer factors of q .)

2.1 BGV-Type Cryptosystems

Our implementation uses a variant of the BGV cryptosystem due to Gentry, Halevi and Smart, specifically the one described in [11, Appendix D] (in the full version). In this cryptosystem both ciphertexts and secret keys are vectors over the polynomial ring \mathbb{A} , and the native plaintext space is the space of binary polynomials \mathbb{A}_2 . (More generally it could be \mathbb{A}_p for some fixed $p \geq 2$, but in our case we will always use \mathbb{A}_2 .)

At any point during the homomorphic evaluation there is some “current integer modulus q ” and “current secret key \mathbf{s} ”, that change from time to time. A ciphertext \mathbf{c} is decrypted using the current secret key \mathbf{s} by taking inner product over \mathbb{A}_q (with q the current modulus) and then reducing the result modulo 2 *in coefficient representation*. Namely, the decryption formula is

$$a \leftarrow [\underbrace{[\langle \mathbf{c}, \mathbf{s} \rangle \bmod \Phi_m(X)]_q}_{\text{noise}}]_2. \quad (1)$$

The polynomial $[\langle \mathbf{c}, \mathbf{s} \rangle \bmod \Phi_m(X)]_q$ is called the “noise” in the ciphertext \mathbf{c} . Informally, \mathbf{c} is a *valid ciphertext* with respect to secret key \mathbf{s} and modulus q if this noise has “sufficiently small norm” relative to q . The meaning of “sufficiently small norm” is whatever is needed to ensure that the noise does not wrap around q when performing homomorphic operations, in our implementation we keep the norm of the noise always below some pre-set bound (which is determined in the full version of the paper).

Following [16,11], the specific norm that we use to evaluate the magnitude of the noise is the “canonical embedding norm reduced mod q ”, specifically we use the conventions as described in [11, Appendix D] (in the full version). This is useful to get smaller parameters, but for the purpose of presentation the reader can think of the norm as the Euclidean norm of the noise in coefficient representation. More details are given in the Appendices. We refer to the norm of the noise as *the noise magnitude*.

The central feature of BGV-type cryptosystems is that the current secret key and modulus evolve as we apply operations to ciphertexts. We apply five different operations to ciphertexts during homomorphic evaluation. Three of them — addition, multiplication, and automorphism — are “semantic operations” that we use to evolve the plaintext data which is encrypted under those ciphertexts. The other two operations — key-switching and modulus-switching — are used for “maintenance”: These operations do not change the plaintext at all, they only change the current key or modulus (respectively), and they are mainly used to control the complexity of the evaluation. Below we briefly describe each of these five operations on a high level. For reasons of space, key generation and encryption are described in the full version of the paper, with even more details being provided in [11, Appendix D].

Addition. Homomorphic addition of two ciphertext vectors with respect to the same secret key and modulus q is done just by adding the vectors over \mathbb{A}_q . If the two arguments were encrypting the plaintext polynomials $a_1, a_2 \in \mathbb{A}_2$ then the sum will be an encryption of $a_1 + a_2 \in \mathbb{A}_2$. This operation has no effect on the current modulus or key, and the norm of the noise is at most the sum of norms from the noise in the two arguments.

Multiplication. Homomorphic multiplication is done via tensor product over \mathbb{A}_q . In principle, if the two arguments have dimension n over \mathbb{A}_q then the product ciphertext has dimension n^2 , each entry in the output computed as the product of one entry from the first argument and one entry from the second.³

This operation does not change the current modulus, but it changes the current key: If the two input ciphertexts are valid with respect to the dimension- n secret key vector \mathbf{s} , encrypting the plaintext polynomials $a_1, a_2 \in \mathbb{A}_2$, then the output is valid with respect to the dimension- n^2 secret key \mathbf{s}' which is the tensor product of \mathbf{s} with itself, and it encrypts the polynomial $a_1 \cdot a_2 \in \mathbb{A}_2$. The norm of the noise in the product ciphertext can be bounded in terms of the product of norms of the noise in the two arguments. For our choice of norm function, the norm of the product is no larger than the product of the norms of the two arguments.

Key Switching. The public key of BGV-type cryptosystems includes additional components to enable converting a valid ciphertext with respect to one key into a valid

³ It was shown in [5] that over polynomial rings this operation can be implemented while increasing the dimension only to $2n - 1$ rather than to n^2 .

ciphertext encrypting the same plaintext with respect to another key. For example, this is used to convert the product ciphertext which is valid with respect to a high-dimension key back to a ciphertext with respect to the original low-dimension key.

To allow conversion from dimension- n' key \mathbf{s}' to dimension- n key \mathbf{s} (both with respect to the same modulus q), we include in the public key a matrix $W = W[\mathbf{s}' \rightarrow \mathbf{s}]$ over \mathbb{A}_q , where the i 'th column of W is roughly an encryption of the i 'th entry of \mathbf{s}' with respect to \mathbf{s} (and the current modulus). Then given a valid ciphertext \mathbf{c}' with respect to \mathbf{s}' , we roughly compute $\mathbf{c} = W \cdot \mathbf{c}'$ to get a valid ciphertext with respect to \mathbf{s} .

In some more detail, the BGV key switching transformation first ensures that the norm of the ciphertext \mathbf{c}' itself is sufficiently low with respect to q . In [3] this was done by working with the binary encoding of \mathbf{c}' , and one of our main optimization in this work is a different method for achieving the same goal (cf. Section 3.1). Then, if the i 'th entry in \mathbf{s}' is $\mathbf{s}'_i \in \mathbb{A}$ (with norm smaller than q), then the i 'th column of $W[\mathbf{s}' \rightarrow \mathbf{s}]$ is an n -vector \mathbf{w}_i such that $[\langle \mathbf{w}_i, \mathbf{s} \rangle \bmod \Phi_m(X)]_q = 2e_i + \mathbf{s}'_i$ for a low-norm polynomial $e_i \in \mathbb{A}$. Denoting $\mathbf{e} = (e_1, \dots, e_{n'})$, this means that we have $\mathbf{s}W = \mathbf{s}' + 2\mathbf{e}$ over \mathbb{A}_q . For any ciphertext vector \mathbf{c}' , setting $\mathbf{c} = W \cdot \mathbf{c}' \in \mathbb{A}_q$ we get the equation

$$[\langle \mathbf{c}, \mathbf{s} \rangle \bmod \Phi_m(X)]_q = [\mathbf{s}W\mathbf{c}' \bmod \Phi_m(X)]_q = [\langle \mathbf{c}', \mathbf{s}' \rangle + 2\langle \mathbf{c}', \mathbf{e} \rangle \bmod \Phi_m(X)]_q$$

Since \mathbf{c}' , \mathbf{e} , and $[\langle \mathbf{c}', \mathbf{s}' \rangle \bmod \Phi_m(X)]_q$ all have low norm relative to q , then the addition on the right-hand side does not cause a wrap around q , hence we get $[[\langle \mathbf{c}, \mathbf{s} \rangle \bmod \Phi_m(X)]_q]_2 = [[\langle \mathbf{c}', \mathbf{s}' \rangle \bmod \Phi_m(X)]_q]_2$, as needed. The key-switching operation changes the current secret key from \mathbf{s}' to \mathbf{s} , and does not change the current modulus. The norm of the noise is increased by at most an additive factor of $2\|\langle \mathbf{c}', \mathbf{e} \rangle\|$.

Modulus Switching. The modulus switching operation is intended to reduce the norm of the noise, to compensate for the noise increase that results from all the other operations. To convert a ciphertext \mathbf{c} with respect to secret key \mathbf{s} and modulus q into a ciphertext \mathbf{c}' encrypting the same thing with respect to the same secret key but modulus q' , we roughly just scale \mathbf{c} by a factor q'/q (thus getting a fractional ciphertext), then round appropriately to get back an integer ciphertext. Specifically \mathbf{c}' is a ciphertext vector satisfying (a) $\mathbf{c}' = \mathbf{c} \pmod{2}$, and (b) the “rounding error term” $\tau \stackrel{\text{def}}{=} \mathbf{c}' - (q'/q)\mathbf{c}$ has low norm. Converting \mathbf{c} to \mathbf{c}' is easy in coefficient representation, and one of our optimizations is a method for doing the same in evaluation representation (cf. Section 3.2). This operation leaves the current key \mathbf{s} unchanged, changes the current modulus from q to q' , and the norm of the noise is changed as $\|n'\| \leq (q'/q)\|n\| + \|\tau \cdot s\|$. Note that if the key \mathbf{s} has low norm and q' is sufficiently smaller than q , then the noise magnitude decreases by this operation.

A BGV-type cryptosystem has a chain of moduli, $q_0 < q_1 \cdots < q_{L-1}$, where fresh ciphertexts are with respect to the largest modulus q_{L-1} . During homomorphic evaluation every time the (estimated) noise grows too large we apply modulus switching from q_i to q_{i-1} in order to decrease it back. Eventually we get ciphertexts with respect to the smallest modulus q_0 , and we cannot compute on them anymore (except by using bootstrapping).

Automorphisms. In addition to adding and multiplying polynomials, another useful operation is converting the polynomial $a(X) \in \mathbb{A}$ to $a^{(i)}(X) \stackrel{\text{def}}{=} a(X^i) \bmod \Phi_m(X)$.

Denoting by κ_i the transformation $\kappa_i : a \mapsto a^{(i)}$, it is a standard fact that the set of transformations $\{\kappa_i : i \in (\mathbb{Z}/m\mathbb{Z})^*\}$ forms a group under composition (which is the Galois group $\text{Gal}(\mathbb{Q}(\zeta_m)/\mathbb{Q})$), and this group is isomorphic to $(\mathbb{Z}/m\mathbb{Z})^*$. In [3,11] it was shown that applying the transformations κ_i to the plaintext polynomials is very useful, some more examples of its use can be found in our Section 4.

Denoting by $\mathbf{c}^{(i)}, \mathbf{s}^{(i)}$ the vector obtained by applying κ_i to each entry in \mathbf{c}, \mathbf{s} , respectively, it was shown in [3,11] that if \mathbf{s} is a valid ciphertext encrypting a with respect to key \mathbf{s} and modulus q , then $\mathbf{c}^{(i)}$ is a valid ciphertext encrypting $a^{(i)}$ with respect to key $\mathbf{s}^{(i)}$ and the same modulus q . Moreover the norm of noise remains the same under this operation. We remark that we can apply key-switching to $\mathbf{c}^{(i)}$ in order to get an encryption of $a^{(i)}$ with respect to the original key \mathbf{s} .

2.2 Computing on Packed Ciphertexts

Smart and Vercauteren observed [19,20] that the plaintext space \mathbb{A}_2 can be viewed as a vector of “plaintext slots”, by an application the polynomial Chinese Remainder Theorem. Specifically, if the ring polynomial $\Phi_m(X)$ factors modulo 2 into a product of irreducible factors $\Phi_m(X) = \prod_{j=0}^{\ell-1} F_j(X) \pmod{2}$, then a plaintext polynomial $a(X) \in \mathbb{A}_2$ can be viewed as encoding ℓ different small polynomials, $a_j = a \pmod{F_j}$. Just like for integer Chinese Remaindering, addition and multiplication in \mathbb{A}_2 correspond to element-wise addition and multiplication of the vectors of slots.

The effect of the automorphisms is a little more involved. When i is a power of two then the transformations $\kappa_i : a \mapsto a^{(i)}$ is just applied to each slot separately. When i is not a power of two the transformation κ_i has the effect of roughly shifting the values between the different slots. For example, for some parameters we could get a cyclic shift of the vector of slots: If a encodes the vector $(a_0, a_1, \dots, a_{\ell-1})$, then $\kappa_i(a)$ (for some i) could encode the vector $(a_{\ell-1}, a_0, \dots, a_{\ell-2})$. This was used in [11] to devise efficient procedures for applying arbitrary permutations to the plaintext slots.

We note that the values in the plaintext slots are not just bits, rather they are polynomials modulo the irreducible F_j 's, so they can be used to represent elements in extension fields $\text{GF}(2^d)$. In particular, in some of our AES implementations we used the plaintext slots to hold elements of $\text{GF}(2^8)$, and encrypt one byte of the AES state in each slot. Then we can use an adaption of the techniques from [11] to permute the slots when performing the AES row-shift and column-mix.

3 General-Purpose Optimizations

Below we summarize our optimizations that are not tied directly to the AES circuit and can be used also in homomorphic evaluation of other circuits. Underlying many of these optimizations is our choice of keeping ciphertext and key-switching matrices in evaluation (double-CRT) representation. Our chain of moduli is defined via a set of primes of roughly the same size, p_0, \dots, p_{L-1} , all chosen such that $\mathbb{Z}/p_i\mathbb{Z}$ has a m 'th roots of unity. (In other words, $m|p_i - 1$ for all i .) For $i = 0, \dots, L-1$ we then define our i 'th modulus as $q_i = \prod_{j=0}^i p_j$. The primes p_0 and p_{L-1} are special (p_0 is chosen to ensure decryption works, and p_{L-1} is chosen to control noise immediately

after encryption), however all other primes p_i are of size $2^{17} \leq p_i \leq 2^{20}$ if $L < 100$, see the full version for further details.

In the t -th level of the scheme we have ciphertexts consisting of elements in \mathbb{A}_{q_t} (i.e., polynomials modulo $(\Phi_m(X), q_t)$). We represent an element $c \in \mathbb{A}_{q_t}$ by a $\phi(m) \times (t+1)$ “matrix” of its evaluations at the primitive m -th roots of unity modulo the primes p_0, \dots, p_t . Computing this representation from the coefficient representation of c involves reducing c modulo the p_i ’s and then $t+1$ invocations of the FFT algorithm, modulo each of the p_i (picking only the FFT coefficients corresponding to $(\mathbb{Z}/m\mathbb{Z})^*$). To convert back to coefficient representation we invoke the inverse FFT algorithm $t+1$ times, each time padding the $\phi(m)$ -vector of evaluation point with $m - \phi(m)$ zeros (for the evaluations at the non-primitive roots of unity). This yields the coefficients of $t+1$ polynomials modulo $(X^m - 1, p_i)$ for $i = 0, \dots, t$, we then reduce each of these polynomials modulo $(\Phi_m(X), p_i)$ and apply Chinese Remainder interpolation. We stress that we try to perform these transformations as rarely as we can.

3.1 A New Variant of Key Switching

As described in Section 2, the key-switching transformation introduces an additive factor of $2 \langle c', e \rangle$ in the noise, where c' is the input ciphertext and e is the noise component in the key-switching matrix. To keep the noise magnitude below the modulus q , it seems that we need to ensure that the ciphertext c' itself has low norm. In BGV [3] this was done by representing c' as a fixed linear combination of small vectors, i.e. $c' = \sum_i 2^i c'_i$ with c'_i the vector of i ’th bits in c' . Considering the high-dimension ciphertext $c^* = (c'_0 | c'_1 | c'_2 | \dots)$ and secret key $s^* = (s' | 2s' | 4s' | \dots)$, we note that we have $\langle c^*, s^* \rangle = \langle c', s' \rangle$, and c^* has low norm (since it consists of 0-1 polynomials). BGV therefore included in the public key the matrix $W = W[s^* \rightarrow s]$ (rather than $W[s' \rightarrow s]$), and had the key-switching transformation computes c^* from c' and sets $c = W \cdot c^*$.

When implementing key-switching, there are two drawbacks to the above approach. First, this increases the dimension (and hence the size) of the key switching matrix. This drawback is fatal when evaluating deep circuits, since having enough memory to keep the key-switching matrices turns out to be the limiting factor in our ability to evaluate these deep circuits. In addition, for this key-switching we must first convert c' to coefficient representation (in order to compute the c'_i ’s), then convert each of the c'_i ’s back to evaluation representation before multiplying by the key-switching matrix. In level t of the circuit, this seem to require $\Omega(t \log q_t)$ FFTs.

In this work we propose a different variant: Rather than manipulating c' to decrease its norm, we instead temporarily increase the modulus q . We recall that for a valid ciphertext c' , encrypting plaintext a with respect to s' and q , we have the equality $\langle c', s' \rangle = 2e' + a$ over A_q , for a low-norm polynomial e' . This equality, we note, implies that for every odd integer p we have the equality $\langle c', ps' \rangle = 2e'' + a$, holding over A_{pq} , for the “low-norm” polynomial e'' (namely $e'' = p \cdot e' + \frac{p-1}{2}a$). Clearly, when considered relative to secret key ps and modulus pq , the noise in c' is p times larger than it was relative to s and q . However, since the modulus is also p times larger, we maintain that the noise has norm sufficiently smaller than the modulus. In other words, c' is still a valid ciphertext that encrypts the same plaintext a with respect to secret key

ps and modulus pq . By taking p large enough, we can ensure that the norm of \mathbf{c}' (which is independent of p) is sufficiently small relative to the modulus pq .

We therefore include in the public key a matrix $W = W[ps' \rightarrow \mathbf{s}]$ modulo pq for a large enough odd integer p . (Specifically we need $p \approx q\sqrt{m}$.) Given a ciphertext \mathbf{c}' , valid with respect to \mathbf{s} and q , we apply the key-switching transformation simply by setting $\mathbf{c} = W \cdot \mathbf{c}'$ over \mathbb{A}_{pq} . The additive noise term $\langle \mathbf{c}', \mathbf{e} \rangle$ that we get is now small enough relative to our large modulus pq , thus the resulting ciphertext \mathbf{c} is valid with respect to \mathbf{s} and pq . We can now switch the modulus back to q (using our modulus switching routine), hence getting a valid ciphertext with respect to \mathbf{s} and q .

We note that even though we no longer break \mathbf{c}' into its binary encoding, it seems that we still need to recover it in coefficient representation in order to compute the evaluations of $\mathbf{c}' \bmod p$. However, since we do not increase the dimension of the ciphertext vector, this procedure requires only $O(t)$ FFTs in level t (vs. $O(t \log q_t) = O(t^2)$ for the original BGV variant). Also, the size of the key-switching matrix is reduced by roughly the same factor of $\log q_t$.

Our new variant comes with a price tag, however: We use key-switching matrices relative to a larger modulus, but still need the noise term in this matrix to be small. This means that the LWE problem underlying this key-switching matrix has larger ratio of modulus/noise, implying that we need a larger dimension to get the same level of security than with the original BGV variant. In fact, since our modulus is more than squared (from q to pq with $p > q$), the dimension is increased by more than a factor of two. This translates to more than doubling of the key-switching matrix, partly negating the size and running time advantage that we get from this variant.

We comment that a hybrid of the two approaches could also be used: we can decrease the norm of \mathbf{c}' only somewhat by breaking it into digits (as opposed to binary bits as in [3]), and then increase the modulus somewhat until it is large enough relative to the smaller norm of \mathbf{c}' . We speculate that the optimal setting in terms of runtime is found around $p \approx \sqrt{q}$, but so far did not try to explore this tradeoff.

3.2 Modulus Switching in Evaluation Representation

Given an element $c \in \mathbb{A}_{q_t}$ in evaluation (double-CRT) representation relative to $q_t = \prod_{j=0}^t p_j$, we want to modulus-switch to q_{t-1} – i.e., scale down by a factor of p_t ; we call this operation $\text{Scale}(c, q_t, q_{t-1})$. The output should be $c' \in \mathbb{A}$, represented via the same double-CRT format (with respect to p_0, \dots, p_{t-1}), such that (a) $c' \equiv c \pmod{2}$, and (b) the “rounding error term” $\tau = c' - (c/p_t)$ has a very low norm. As p_t is odd, we can equivalently require that the element $c^\dagger \stackrel{\text{def}}{=} p_t \cdot c'$ satisfy

- (i) c^\dagger is divisible by p_t ,
- (ii) $c^\dagger \equiv c \pmod{2}$, and
- (iii) $c^\dagger - c$ (which is equal to $p_t \cdot \tau$) has low norm.

Rather than computing c' directly, we will first compute c^\dagger and then set $c' \leftarrow c^\dagger / p_t$. Observe that once we compute c^\dagger in double-CRT format, it is easy to output also c' in double-CRT format: given the evaluations for c^\dagger modulo p_j ($j < t$), simply multiply them by $p_t^{-1} \pmod{p_j}$. The algorithm to output c^\dagger in double-CRT format is as follows:

1. Set \bar{c} to be the coefficient representation of $c \bmod p_t$. (Computing this requires a single “small FFT” modulo the prime p_t .)
2. Add or subtract p_t from every odd coefficient of \bar{c} , thus obtaining a polynomial δ with coefficients in $(-p_t, p_t]$ such that $\delta \equiv \bar{c} \equiv c \pmod{p_t}$ and $\delta \equiv 0 \pmod{2}$.
3. Set $c^\dagger = c - \delta$, and output it in double-CRT representation.

Since we already have c in double-CRT representation, we only need the double-CRT representation of δ , which requires t more “small FFTs” modulo the p_j ’s.

As all the coefficients of c^\dagger are within p_t of those of c , the “rounding error term” $\tau = (c^\dagger - c)/p_t$ has coefficients of magnitude at most one, hence it has low norm.

The procedure above uses $t + 1$ small FFTs in total. This should be compared to the naive method of just converting everything to coefficient representation modulo the primes ($t + 1$ FFTs), CRT-interpolating the coefficients, dividing and rounding appropriately the large integers (of size $\approx q_t$), CRT-decomposing the coefficients, and then converting back to evaluation representation ($t + 1$ more FFTs). The above approach makes explicit use of the fact that we are working in a plaintext space modulo 2; in the full version we present a technique which works when the plaintext space is defined modulo a larger modulus.

3.3 Dynamic Noise Management

As described in the literature, BGV-type cryptosystems tacitly assume that each homomorphic operation operation is followed a modulus switch to reduce the noise magnitude. In our implementation, however, we attach to each ciphertext an estimate of the noise magnitude in that ciphertext, and use these estimates to decide dynamically when a modulus switch must be performed.

Each modulus switch consumes a level, and hence a goal is to reduce, over a computation, the number of levels consumed. By paying particular attention to the parameters of the scheme, and by carefully analyzing how various operations affect the noise, we are able to control the noise much more carefully than in prior work. In particular, we note that modulus-switching is really only necessary just prior to multiplication (when the noise magnitude is about to get squared), in other times it is acceptable to keep the ciphertexts at a higher level (with higher noise).

3.4 Randomized Multiplication by Constants

Our implementation of the AES round function uses just a few multiplication operations (only seven per byte!), but it requires a relatively large number of multiplications of encrypted bytes by constants. Hence it becomes important to try and squeeze down the increase in noise when multiplying by a constant. To that end, we encode a constant polynomial in \mathbb{A}_2 as a polynomial with coefficients in $\{-1, 0, 1\}$ rather than in $\{0, 1\}$. Namely, we have a procedure $\text{Randomize}(\alpha)$ that takes a polynomial $\alpha \in \mathbb{A}_2$ and replaces each non-zero coefficients with a coefficients chosen uniformly from $\{-1, 1\}$. By Chernoff bound, we expect that for α with h nonzero coefficients, the canonical embedding norm of $\text{Randomize}(\alpha)$ to be bounded by $O(\sqrt{h})$ with high probability (assuming that h is large enough for the bound to kick in). This yields a better bound on the noise increase than the trivial bound of h that we would get if we just multiply

by α itself. (In the full version we present a heuristic argument that we use to bound the noise, which yields the same asymptotic bounds but slightly better constants.)

4 Homomorphic Evaluation of AES

Next we describe our homomorphic implementation of AES-128. We implemented three distinct implementation possibilities; we first describe the “packed implementation”, in which the entire AES state is packed in just one ciphertext. Two other implementations (of byte-slice and bit-slice AES) are described later in Section 4.2. The “packed” implementation uses the least amount of memory (which turns out to be the main constraint in our implementation), and also the fastest running time for a single evaluation. The other implementation choices allow more SIMD parallelism, on the other hand, so they can give better amortized running time when evaluating AES on many blocks in parallel.

A Brief Overview of AES. The AES-128 cipher consists of ten applications of the same keyed round function (with different round keys). The round function operates on a 4×4 matrix of bytes, which are sometimes considered as element of \mathbb{F}_{2^8} . The basic operations that are performed during the round function are AddKey, SubBytes, ShiftRows, MixColumns. The AddKey is simply an XOR operation of the current state with 16 bytes of key; the SubBytes operation consists of an inversion in the field \mathbb{F}_{2^8} followed by a fixed \mathbb{F}_2 -linear map on the bits of the element (relative to a fixed polynomial representation of \mathbb{F}_{2^8}); the ShiftRows rotates the entries in the row i of the 4×4 matrix by $i - 1$ places to the left; finally the MixColumns operations pre-multiplies the state matrix by a fixed 4×4 matrix.

Our Packed Representation of the AES state. For our implementation we chose the native plaintext space of our homomorphic encryption so as to support operations on the finite field \mathbb{F}_{2^8} . To this end we choose our ring polynomial as $\Phi_m(X)$ that factors modulo 2 into degree- d irreducible polynomials such that $8|d$. (In other words, the smallest integer d such that $m|(2^d - 1)$ is divisible by 8.) This means that our plaintext slots can hold elements of \mathbb{F}_{2^d} , and in particular we can use them to hold elements of \mathbb{F}_{2^8} which is a sub-field of \mathbb{F}_{2^d} . Since we have $\ell = \phi(m)/d$ plaintext slots in each ciphertext, we can represent upto $\lfloor \ell/16 \rfloor$ complete AES state matrices per ciphertext.

Moreover, we choose our parameter m so that there exists an element $g \in \mathbb{Z}_m^*$ that has order 16 in both \mathbb{Z}_m^* and the quotient group $\mathbb{Z}_m^*/\langle 2 \rangle$. This condition means that if we put 16 plaintext bytes in slots t, tg, tg^2, tg^3, \dots (for some $t \in \mathbb{Z}_m^*$), then the conjugation operation $X \mapsto X^g$ implements a cyclic right shift over these sixteen plaintext bytes.

In the computation of the AES round function we use several constants. Some constants are used in the S-box lookup phase to implement the AES bit-affine transformation, these are denoted γ and γ_{2^j} for $j = 0, \dots, 7$. In the row-shift/col-mix part we use a constant C_{slot} that has 1 in slots corresponding to $t \cdot g^i$ for $i = 0, 4, 8, 12$, and 0 in all the other slots of the form $t \cdot g^i$. (Here slot t is where we put the first AES byte.) We also use ‘X’ to denote the constant that has the element X in all the slots.

4.1 Homomorphic Evaluation of the Basic Operations

We now examine each AES operation in turn, and describe how it is implemented homomorphically. For each operation we denote the plaintext polynomial underlying a given input ciphertext c by a , and the corresponding content of the ℓ plaintext slots are denoted as an ℓ -vector $(\alpha_i)_{i=1}^\ell$, with each $\alpha_i \in \mathbb{F}_{2^8}$.

AddKey and SubBytes. The AddKey is just a simple addition of ciphertexts, which yields a 4×4 matrix of bytes in the input to the SubBytes operation. We place these 16 bytes in plaintext slots $t g^i$ for $i = 0, 1, \dots, 15$, using column-ordering to decide which byte goes in what slot, namely we have

$$a \approx [\alpha_{00} \alpha_{10} \alpha_{20} \alpha_{30} \alpha_{01} \alpha_{11} \alpha_{21} \alpha_{31} \alpha_{02} \alpha_{12} \alpha_{22} \alpha_{32} \alpha_{03} \alpha_{13} \alpha_{23} \alpha_{33}],$$

encrypting the input plaintext matrix

$$A = (\alpha_{ij})_{i,j} = \begin{pmatrix} \alpha_{00} & \alpha_{01} & \alpha_{02} & \alpha_{03} \\ \alpha_{10} & \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{20} & \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{30} & \alpha_{31} & \alpha_{32} & \alpha_{33} \end{pmatrix}.$$

During S-box lookup, each plaintext byte α_{ij} should be replaced by $\beta_{ij} = S(\alpha_{ij})$, where $S(\cdot)$ is a fixed permutation on the bytes. Specifically, $S(x)$ is obtained by first computing $y = x^{-1}$ in \mathbb{F}_{2^8} (with 0 mapped to 0), then applying a bitwise affine transformation $z = T(y)$ where elements in \mathbb{F}_{2^8} are treated as bit strings with representation polynomial $G(X) = x^8 + x^4 + x^3 + x + 1$.

We implement \mathbb{F}_{2^8} inversion followed by the \mathbb{F}_2 affine transformation using the Frobenius automorphisms, $X \rightarrow X^{2^j}$. Recall that for a power of two $k = 2^j$, the transformation $\kappa_k(a(X)) = (a(X^k) \bmod \Phi_m(X))$ is applied separately to each slot, hence we can use it to transform the vector $(\alpha_i)_{i=1}^\ell$ into $(\alpha_i^k)_{i=1}^\ell$. We note that applying the Frobenius automorphisms to ciphertexts has almost no influence on the noise magnitude, and hence it does not consume any levels.⁴

Inversion over \mathbb{F}_{2^8} is done using essentially the same procedure as Algorithm 2 from [18] for computing $\beta = \alpha^{-1} = \alpha^{254}$. This procedure takes only three Frobenius automorphisms and four multiplications, arranged in a depth-3 circuit (see details below.) To apply the AES \mathbb{F}_2 affine transformation, we use the fact that any \mathbb{F}_2 affine transformation can be computed as a \mathbb{F}_{2^8} affine transformation over the conjugates. Thus there are constants $\gamma_0, \gamma_1, \dots, \gamma_7, \delta \in \mathbb{F}_{2^8}$ such that the AES affine transformation $T_{\text{AES}}(\cdot)$ can be expressed as $T_{\text{AES}}(\beta) = \delta + \sum_{j=0}^7 \gamma_j \cdot \beta^{2^j}$ over \mathbb{F}_{2^8} . We therefore again apply the Frobenius automorphisms to compute eight ciphertexts encrypting the polynomials $\kappa_k(b)$ for $k = 1, 2, 4, \dots, 128$, and take the appropriate linear combination (with coefficients the γ_j 's) to get an encryption of the vector $(T_{\text{AES}}(\alpha_i^{-1}))_{i=1}^\ell$. For our parameters, a multiplication-by-constant operation consumes roughly half a level in terms of added noise.

⁴ It does increase the noise magnitude somewhat, because we need to do key switching after these automorphisms. But this is only a small influence, and we will ignore it here.

One subtle implementation detail to note here, is that although our plaintext slots all hold elements of the same field \mathbb{F}_{2^8} , they hold these elements with respect to different polynomial encodings. The AES affine transformation, on the other hand, is defined with respect to one particular fixed polynomial encoding. This means that we must implement in the i 'th slot not the affine transformation $T_{\text{AES}}(\cdot)$ itself but rather the projection of this transformation onto the appropriate polynomial encoding: When we take the affine transformation of the eight ciphertexts encrypting $b_j = \kappa_{2^j}(b)$, we therefore multiply the encryption of b_j not by a constant that has γ_j in all the slots, but rather by a constant that has in slot i the projection of γ_j to the polynomial encoding of slot i .

Below we provide a pseudo-code description of our S-box lookup implementation, together with an approximation of the levels that are consumed by these operations. (These approximations are somewhat under-estimates, however.)

	Level
Input: ciphertext c	t
// Compute $c_{254} = c^{-1}$	
1. $c_2 \leftarrow c \gg 2$	t // Frobenius $X \mapsto X^2$
2. $c_3 \leftarrow c \times c_2$	$t + 1$ // Multiplication
3. $c_{12} \leftarrow c_3 \gg 4$	$t + 1$ // Frobenius $X \mapsto X^4$
4. $c_{14} \leftarrow c_{12} \times c_2$	$t + 2$ // Multiplication
5. $c_{15} \leftarrow c_{12} \times c_3$	$t + 2$ // Multiplication
6. $c_{240} \leftarrow c_{15} \gg 16$	$t + 2$ // Frobenius $X \mapsto X^{16}$
7. $c_{254} \leftarrow c_{240} \times c_{14}$	$t + 3$ // Multiplication
// Affine transformation over \mathbb{F}_2	
8. $c'_{2^j} \leftarrow c_{254} \gg 2^j$ for $j = 0, 1, 2, \dots, 7$	$t + 3$ // Frobenius $X \mapsto X^{2^j}$
9. $c'' \leftarrow \gamma + \sum_{j=0}^7 \gamma_j \times c'_{2^j}$	$t + 3.5$ // Linear combination over \mathbb{F}_{2^8}

ShiftRows and MixColumns. As commonly done, we interleave the ShiftRows/MixColumns operations, viewing both as a single linear transformation over vectors from $(\mathbb{F}_{2^8})^{16}$. As mentioned above, by a careful choice of the parameter m and the placement of the AES state bytes in our plaintext slots, we can implement a rotation-by- i of the rows of the AES matrix as a single automorphism operations $X \mapsto X^{g^i}$ (for some element $g \in (\mathbb{Z}/m\mathbb{Z})^*$). Given the ciphertext c'' after the SubBytes step, we use these operations (in conjunction with ℓ -SELECT operations, as described in [11]) to compute four ciphertexts corresponding to the appropriate permutations of the 16 bytes (in each of the $\ell/16$ different input blocks). These four ciphertexts are combined via a linear operation (with coefficients 1, X , and $(1 + X)$) to obtain the final result of this round function. Below is a pseudo-code of this implementation and an approximation for the levels that it consumes (starting from $t - 3.5$). We note that the permutations are implemented using automorphisms and multiplication by constant, thus we expect them to consume roughly 1/2 level.

	Level
Input: ciphertext \mathbf{c}''	$t + 3.5$
10. $\mathbf{c}_j^* \leftarrow \pi_j(\mathbf{c}'')$ for $j = 1, 2, 3, 4$	$t + 4.0$ // Permutations
11. Output $X \cdot \mathbf{c}_1^* + (X + 1) \cdot \mathbf{c}_2^* + \mathbf{c}_3^* + \mathbf{c}_4^*$	$t + 4.5$ // Linear combination

The Cost of One Round Function. The above description yields an estimate of 5 levels for implementing one round function. This is however, an underestimate. The actual number of levels depends on details such as how sparse the scalars are with respect to the embedding via Φ_m in a given parameter set, as well as the accumulation of noise with respect to additions, Frobenius operations etc. Running over many different parameter sets we find the average number of levels per round for this method varies between 5.0 and 6.0.

We mention that the byte-slice and bit-slice implementations, given in Section 4.2 below, can consume less levels per round function, since they do not need to permute slots inside a single ciphertext. Specifically, for our byte-sliced implementation, we only need 4.5-5.0 levels per round on average. However, since we need to manipulate many more ciphertexts, the implementation takes much more time per evaluation and requires much more memory. On the other hand it offers wider parallelism, so yields better amortized time per block. Our bit-sliced implementation should theoretical consume the least number of levels (by purely counting multiplication gates), but the noise introduced by additions means the average number of levels consumed per round varies from 5.0 upto 10.0.

4.2 Byte- and Bit-Slice Implementations

In the byte sliced implementation we use sixteen distinct ciphertexts to represent a single state matrix. (But since each ciphertext can hold ℓ plaintext slots, then these 16 ciphertexts can hold the state of ℓ different AES blocks). In this representation there is no interaction between the slots, thus we operate with pure ℓ -fold SIMD operations. The AddKey and SubBytes steps are exactly as above (except applied to 16 ciphertexts rather than a single one). The permutations in the ShiftRows/MixColumns step are now “for free”, but the scalar multiplication in MixColumns still consumes another level in the modulus chain.

Using the same estimates as above, we expect the number of levels per round to be roughly four (as opposed to the 4.5 of the packed implementation). In practice, again over many parameter sets, we find the average number of levels consumed per round is between 4.5 and 5.0.

For the bit sliced implementation we represent the entire round function as a binary circuit, and we use 128 distinct ciphertexts (one per bit of the state matrix). However each set of 128 ciphertexts is able to represent a total of ℓ distinct blocks. The main issue here is how to create a circuit for the round function which is as shallow, in terms of number of multiplication gates, as possible. Again the main issue is the SubBytes operation as all operations are essentially linear. To implement the SubBytes we used the “depth-16” circuit of Boyar and Peralta [1], which consumes four levels. The rest of the round function can be represented as a set of bit-additions, Thus, implementing

this method means that we consumes a minimum of four levels on computing an entire round function. However, the extensive additions within the Boyar–Peralta circuit mean that we actually end up consuming a lot more. On average this translates into actually consuming between 5.0 and 10.0 levels per round.

4.3 Performance Details

As remarked in the introduction, we implemented the above variant of evaluating AES homomorphically on a very large memory machine; namely a machine with 256 GB of RAM. Firstly parameters were selected, for details see the full version, to cope with 60 levels of computation, and a public/private key pair was generated; along with the key-switching data for multiplication operations and conjugation with-respect-to the Galois group.

As input to the actual computation was an AES plaintext block and the eleven round keys; each of which was encrypted using our homomorphic encryption scheme. Thus the input consisted of eleven packed ciphertexts. Producing the encrypted key schedule took around half an hour. To evaluate the entire ten rounds of AES took just over 36 hours; however each of our ciphertexts could hold 864 plaintext slots of elements in \mathbb{F}_{2^8} , thus we could have processed 54 such AES blocks in this time period. This would result in a throughput of around forty minutes per AES block.

We note that as the algorithm progressed the operations became faster. The first round of the AES function took 7 hours, whereas the penultimate round took 2 hours and the last round took 30 minutes. Recall, the last AES round is somewhat simpler as it does not involve a MixColumns operation.

Whilst our other two implementation choices (given in Section 4.2 below) may seem to yield better amortized per-block timing, the increase in memory requirements and data actually makes them less attractive when encrypting a single block. For example just encrypting the key schedule in the Byte-Sliced variant takes just under 5 hours (with 50 levels), with an entire encryption taking 65 hours (12 hours for the first round, with between 4 and 5 hours for both the penultimate and final rounds). This however equates to an amortized time of just over five minutes per block.

The Bit-Sliced variant requires over 150 hours to just encrypt the key schedule (with 60 levels), and evaluating a single round takes so long that our program is timed out before even a single round is evaluated.

Acknowledgments. We thank Jean-Sebastien Coron for pointing out to us the efficient implementation from [18] of the AES S-box lookup.

The first and second authors are sponsored by DARPA under agreement number FA8750-11-C-0096. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government. Distribution Statement “A” (Approved for Public Release, Distribution Unlimited).

The third author is sponsored by DARPA and AFRL under agreement number FA8750-11-2-0079. The same disclaimers as above apply. He is also supported by the

European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II and via an ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, by EPSRC via grant COED-EP/I03126X, and by a Royal Society Wolfson Merit Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the European Commission or EPSRC.

References

1. Boyar, J., Peralta, R.: A depth-16 circuit for the AES S-box (2011) (manuscript), <http://eprint.iacr.org/2011/332>
2. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP (2012) (manuscript), <http://eprint.iacr.org/2012/078>
3. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. In: Innovations in Theoretical Computer Science, ITCS 2012 (2012), <http://eprint.iacr.org/2011/277>
4. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS 2011. IEEE Computer Society (2011)
5. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
6. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011)
7. Coron, J.-S., Naccache, D., Tibouchi, M.: Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (2012)
8. Damgård, I., Keller, M.: Secure Multiparty AES. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 367–374. Springer, Heidelberg (2010)
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178. ACM (2009)
10. Gentry, C., Halevi, S.: Implementing Gentry’s Fully-Homomorphic Encryption Scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
11. Gentry, C., Halevi, S., Smart, N.P.: Fully Homomorphic Encryption with Polylog Overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012), Full version at <http://eprint.iacr.org/2011/566>
12. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium (2011)
13. Orlandi, C., Nielsen, J.B., Nordholt, P.S., Sheshank, S.: A new approach to practical active-secure two-party computation (2011) (manuscript)
14. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: CCSW, pp. 113–124. ACM (2011)
15. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC. ACM (2012)
16. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)

17. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation Is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
18. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
19. Smart, N.P., Vercauteren, F.: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
20. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations (2011), Manuscript at <http://eprint.iacr.org/2011/133>

A Other Optimizations

Some other optimizations that we encountered during our implementation work are discussed next. Not all of these optimizations are useful for our current implementation, but they may be useful in other contexts.

Three-way Multiplications. Sometime we need to multiply several ciphertexts together, and if their number is not a power of two then we do not have a complete binary tree of multiplications, which means that at some point in the process we will have three ciphertexts that we need to multiply together.

The standard way of implementing this 3-way multiplication is via two 2-argument multiplications, e.g., $x \cdot (y \cdot z)$. But it turns out that here it is better to use “raw multiplication” to multiply these three ciphertexts (as done in [5]), thus getting an “extended” ciphertext with four elements, then apply key-switching (and later modulus switching) to this ciphertext. This takes only six ring-multiplication operations (as opposed to eight according to the standard approach), three modulus switching (as opposed to four), and only one key switching (applied to this 4-element ciphertext) rather than two (which are applied to 3-element extended ciphertexts). All in all, this three-way multiplication takes roughly 1.5 times a standard two-element multiplication.

We stress that this technique is not useful for larger products, since for more than three multiplicands the noise begins to grow too large. But with only three multiplicands we get noise of roughly B^3 after the multiplication, which can be reduced to noise $\approx B$ by dropping two levels, and this is also what we get by using two standard two-element multiplications.

Commuting Automorphisms and Multiplications. Recalling that the automorphisms $X \mapsto X^i$ commute with the arithmetic operations, we note that some ordering of these operations can sometimes be better than others. For example, it may be better perform the multiplication-by-constant before the automorphism operation whenever possible. The reason is that if we perform the multiply-by-constant after the key-switching that follows the automorphism, then added noise term due to that key-switching is multiplied by the same constant, thereby making the noise slightly larger. We note that to move the multiplication-by-constant before the automorphism, we need to multiply by a different constant.

Switching to higher-level moduli. We note that it may be better to perform automorphisms at a higher level, in order to make the added noise term due to key-switching small with respect to the modulus. On the other hand operations at high levels are more expensive than the same operations at a lower level. A good rule of thumb is to perform the automorphism operations one level above the lowest one. Namely, if the naive strategy that never switches to higher-level moduli would perform some Frobenius operation at level q_i , then we perform the key-switching following this Frobenius operation at level Q_{i+1} , and then switch back to level q_{i+1} (rather than using Q_i and q_i).

Commuting Addition and Modulus-switching. When we need to add many terms that were obtained from earlier operations (and their subsequent key-switching), it may be better to first add all of these terms relative to the large modulus Q_i before switching the sum down to the smaller q_i (as opposed to switching all the terms individually to q_i and then adding).

Reducing the number of key-switching matrices. When using many different automorphisms $\kappa_i : X \mapsto X^i$ we need to keep many different key-switching matrices in the public key, one for every value of i that we use. We can reduce this memory requirement, at the expense of taking longer to perform the automorphisms. We use the fact that the Galois group $\mathcal{G}\text{al}$ that contains all the maps κ_i (which is isomorphic to $(\mathbb{Z}/m\mathbb{Z})^*$) is generated by a relatively small number of generators. (Specifically, for our choice of parameters the group $(\mathbb{Z}/m\mathbb{Z})^*$ has two or three generators.) It is therefore enough to store in the public key only the key-switching matrices corresponding to κ_{g_j} 's for these generators g_j of the group $\mathcal{G}\text{al}$. Then in order to apply a map κ_i we express it as a product of the generators and apply these generators to get the effect of κ_i . (For example, if $i = g_1^2 \cdot g_2$ then we need to apply κ_{g_1} twice followed by a single application of κ_{g_2} .)

Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP

Zvika Brakerski*

Stanford University
zvika@stanford.edu

Abstract. We present a new tensoring technique for LWE-based fully homomorphic encryption. While in all previous works, the ciphertext noise grows quadratically ($B \rightarrow B^2 \cdot \text{poly}(n)$) with every multiplication (before “refreshing”), our noise only grows linearly ($B \rightarrow B \cdot \text{poly}(n)$).

We use this technique to construct a *scale-invariant* fully homomorphic encryption scheme, whose properties only depend on the ratio between the modulus q and the initial noise level B , and not on their absolute values.

Our scheme has a number of advantages over previous candidates: It uses the same modulus throughout the evaluation process (no need for “modulus switching”), and this modulus can take arbitrary form. In addition, security can be *classically* reduced from the worst-case hardness of the GapSVP problem (with quasi-polynomial approximation factor), whereas previous constructions could only exhibit a quantum reduction from GapSVP.

1 Introduction

Fully homomorphic encryption has been the focus of extensive study since the first candidate scheme was introduced by Gentry [9]. In a nutshell, fully homomorphic encryption allows to perform arbitrary computation on encrypted data. It can thus be used, for example, to outsource a computation to a remote server without compromising data privacy.

The first generation of fully homomorphic schemes [4, 6, 7, 9, 10, 26] that started with Gentry’s seminal work, all followed a similar and fairly complicated methodology, often relying on relatively strong computational assumptions. A second generation of schemes started with the work of Brakerski and Vaikuntanathan [5], who established full homomorphism in a simpler way, based on the *learning with errors* (LWE) assumption. Using known reductions [20, 22], the security of their construction is based on the (often quantum) hardness of approximating some short vector problems in worst-case lattices. Their scheme was then improved by Brakerski, Gentry and Vaikuntanathan [3], as we describe below.

In LWE-based schemes such as [3, 5], ciphertexts are represented as vectors in \mathbb{Z}_q , for some modulus q . The decryption process is essentially computing an

* Supported by a Simons Postdoctoral Fellowship and by DARPA.

inner product of the ciphertext and the secret key vector, which produces a noisy version of the message (the noise is added at encryption for security purposes). The noise increases with every homomorphic operation, and correct decryption is guaranteed if the final noise magnitude is below $q/4$. Homomorphic addition roughly doubles the noise, while homomorphic multiplication roughly squares it.

In the [5] scheme, after L levels of multiplication (e.g. evaluating a depth L multiplication tree), the noise grows from an initial magnitude of B , to B^{2^L} . Hence, to enable decryption, a very large modulus $q \approx B^{2^L}$ was required. This affected both efficiency and security (the security of the scheme depends inversely on the ratio q/B , so bigger q for the same B means less security).

The above was improved by [3], who suggested to *scale down* the ciphertext vector after every multiplication (they call this “modulus switching”, see below).¹ That is, to go from a vector \mathbf{c} over \mathbb{Z}_q , into the vector \mathbf{c}/w over $\mathbb{Z}_{q/w}$ (for some scaling factor w). Scaling “switches” the modulus q to a smaller q/w , but also reduces the noise by the same factor (from B to B/w). To see why this change of scale is effective, consider scaling by a factor B after every multiplication (as indeed suggested by [3]): After the first multiplication, the noise goes up to B^2 , but scaling brings it back down to B , at the cost of reducing the modulus to q/B . With more multiplications, the noise magnitude always goes back to B , but the modulus keeps reducing. After L levels of multiplication-and-scaling, the noise magnitude is still B , but the modulus is down to q/B^L . Therefore it is sufficient to use $q \approx B^{L+1}$, which is significantly lower than before. However, this process results in a complicated homomorphic evaluation process that “climbs down the ladder of moduli”.

The success of the scaling methodology teaches us that *perspective* matters: scaling does not change the ratio between the modulus and noise, but it still manages the noise better by changing the perspective in which we view the ciphertext. In this work, we suggest to work in an *invariant perspective* where only the ratio q/B matters (and not the absolute values of q, B as in previous works). We derive a scheme that is superior to the previous best known in simplicity, noise management and security. Details follow.

1.1 Our Results

As explained above, we present a *scale invariant* scheme, by finding an invariant perspective. The idea is very natural based on the outlined motivation: if we scale down the ciphertext by a factor of q , we get a fractional ciphertext modulo 1, with noise magnitude B/q . In this perspective, all choices of q, B with the same B/q ratio will look the same. It turns out that in this perspective, homomorphic multiplication does not square the noise, but rather multiplies it by a polynomial factor $p(n)$ that depends only on the security parameter.² After L levels of multiplication, the noise will grow from B/q to $(B/q) \cdot p(n)^L$, which means that we only need to use $q \approx B \cdot p(n)^L$.

¹ A different scaling technique was already suggested in [5] as a way to simplify decryption and improve efficiency, but not to manage noise.

² More accurately, a polynomial $p(n, \log q)$, but w.l.o.g $q \leq 2^n$.

Interestingly, the idea of working modulo 1 goes back to the early works of Ajtai and Dwork [1], and Regev [21], and to the first formulation of LWE [22]. In a sense, we are “going back to the roots” and showing that these early ideas are instrumental in the construction of homomorphic encryption.

For technical reasons, we don’t implement the scheme over fractions, but rather mimic the invariant perspective over \mathbb{Z}_q (see Section 1.2 for more details). Perhaps surprisingly, the resulting scheme is exactly Regev’s original LWE-based scheme, with additional auxiliary information for the purpose of homomorphic evaluation. The properties of our scheme are summarized in the following theorem:

Theorem. *There exists a homomorphic encryption scheme for depth L circuits, based on the $\text{DLWE}_{n,q,\chi}$ assumption (n -dimensional decision-LWE modulo q , with noise χ), so long as*

$$q/B \geq (O(n \log q))^{L+O(1)},$$

where B is a bound on the values of χ .

The resulting scheme has a number of interesting properties:

1. **Scale invariance.** Homomorphic properties only depend on q/B (as explained above).
2. **No modulus switching.** We work with a single modulus q . We don’t need to switch moduli as in [3,5]. This leads to a simpler description of the scheme (and hopefully better implementations).
3. **No restrictions on the modulus.** Our modulus q can take any form (so long as it satisfies the size requirement). This is achieved by putting the message bit in the most significant bit of the ciphertext, rather than least significant as in previous homomorphic schemes (this can be interpreted as making the *message* scale invariant). We note that for odd q , the least and most significant bit representations are interchangeable.

In particular, in our scheme q can be a power of 2, which can simplify implementation of arithmetics.³ In previous schemes, such q could not be used for binary message spaces.⁴

This, again, is going back to the roots: Early schemes such as [22], and in a sense also [1, 14], encoded ciphertexts in the most significant bits. Switching to least significant bit encoding was (perhaps ironically) motivated by improving homomorphism.

4. **No restrictions on the secret key distribution.** While [3] requires that the secret key is drawn from the noise distribution (LWE in Hermite normal form), our scheme works under any secret key distribution for which the LWE assumption holds.

³ On the downside, such q might reduce efficiency when using ring-LWE (see below) due to FFT embedding issues.

⁴ [11] gain on efficiency by using moduli that are “almost” a power of 2.

5. **Classical Reduction from GapSVP.** One of the appeals of LWE-based cryptography is the known quantum (Regev [22]) and classical (Peikert [20]) reductions from the worst case hardness of lattice problems. Specifically to GapSVP_γ , which is the problem of deciding, given an n dimensional lattice and a number d , between the following two cases: either the lattice has a vector shorter than d , or it doesn't have any vector shorter than $\gamma(n) \cdot d$. The value of γ depends on the ratio q/B (essentially $\gamma = (q/B) \cdot \tilde{O}(n)$), and the smaller γ is, the better the reduction ($\text{GapSVP}_{2^{\Omega(n)}}$ is an easy problem). Peikert's classical reduction requires that $q \approx 2^{n/2}$, which makes his reduction unusable for previous homomorphic schemes, since γ becomes exponential. For example, in [3], $q/B = q/q^{1/(L+1)} = q^{1-1/(L+1)}$ which translates to $\gamma \approx 2^{n/2}$ for the required q .⁵

In our scheme, this problem does not arise. We can instantiate our scheme with any q while hardly affecting the ratio q/B . We can therefore set $q \approx 2^{n/2}$ and get a classical reduction from $\text{GapSVP}_{n^{\mathcal{O}(\log n)}}$, which is currently solvable only in $2^{\tilde{\Omega}(n)}$ time. (This is mostly of theoretical interest, though, since efficiency considerations will favor the smallest possible q .)

Using our scheme as a building block we achieve:

1. **Fully homomorphic encryption using bootstrapping.** Using Gentry's bootstrapping theorem, we present a leveled fully homomorphic scheme based on the classical worst case $\text{GapSVP}_{n^{\mathcal{O}(\log n)}}$ problem. As usual, an additional circular security assumption is required to get a non-leveled scheme.
2. **Leveled fully homomorphic encryption without bootstrapping.** As in [3], our scheme can be used to achieve leveled homomorphism without bootstrapping.
3. **Increased efficiency using ring-LWE (RLWE).** RLWE ([15]) is a version of LWE that works over polynomial rings rather than the integers. Its hardness is quantumly related to short vector problems in *ideal* lattices.

RLWE is a stronger assumption than LWE, but it can dramatically improve the efficiency of schemes [3, 4, 12]. Our methods are readily portable to the RLWE world.

In summary, our construction carries conceptual significance in its simplicity and in a number of theoretical aspects. Its practical usefulness compared to other schemes is harder to quantify, though, since it will vary greatly with the specific implementation and optimizations chosen.

1.2 Our Techniques

Our starting point is Regev's public key encryption scheme. There, the encryption of a message $m \in \{0, 1\}$ is an integer vector \mathbf{c} such that $\langle \mathbf{c}, \mathbf{s} \rangle = \lfloor \frac{q}{2} \rfloor \cdot m + e + qI$, for an integer I and for $|e| \leq E$, for some bound $E < q/4$.

⁵ Peikert suggests to classically base small- q LWE on a new lattice problem that he introduces.

The secret key vector \mathbf{s} is also over the integers. We can assume w.l.o.g that the elements of \mathbf{c}, \mathbf{s} are in the segment $(-q/2, q/2]$. (We note that previous homomorphic constructions used a different variant of Regev's scheme, where $\langle \mathbf{c}, \mathbf{s} \rangle = m + 2e + qI$.)

In this work, we take the invariant perspective on the scheme, and consider the fractional ciphertext $\tilde{\mathbf{c}} = \mathbf{c}/q$. It holds that $\langle \tilde{\mathbf{c}}, \mathbf{s} \rangle = \frac{1}{2} \cdot m + \tilde{e} + I$, where $I \in \mathbb{Z}$ and $|\tilde{e}| \leq E/q = \epsilon$. The elements of \mathbf{c} are now rational numbers in $(-1/2, 1/2]$. Note that the secret key does not change and is still over \mathbb{Z} .

Additive homomorphism is immediate: if \mathbf{c}_1 encrypts m_1 and \mathbf{c}_2 encrypts m_2 , then $\mathbf{c}_{\text{add}} = \mathbf{c}_1 + \mathbf{c}_2$ encrypts $[m_1 + m_2]_2$. The noise grows from ϵ to $\approx 2\epsilon$. Multiplicative homomorphism is achieved by *tensoring* the input ciphertexts:

$$\mathbf{c}_{\text{mult}} = 2 \cdot \mathbf{c}_1 \otimes \mathbf{c}_2 .$$

The tensored ciphertext can be decrypted using a tensored secret key because

$$\left\langle \underbrace{2 \cdot \mathbf{c}_1 \otimes \mathbf{c}_2}_{\mathbf{c}_{\text{mult}}}, \mathbf{s} \otimes \mathbf{s} \right\rangle = 2 \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle .$$

A “key switching” mechanism developed in [5] and generalized in [3] allows to switch back from a tensored secret key into a “normal” one without much additional noise. The details of this mechanism are immaterial for this discussion. We focus on the noise growth in the tensored ciphertext.

We want to show that $2 \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle \approx \frac{1}{2}m_1m_2 + e' + I'$, for a small e' . To do this, we let $I_1, I_2 \in \mathbb{Z}$ be integers such that $\langle \mathbf{c}_1, \mathbf{s} \rangle = \frac{1}{2}m_1 + e_1 + I_1$, and likewise for \mathbf{c}_2 . It can be verified that $|I_1|, |I_2|$ are bounded by $\approx \|\mathbf{s}\|_1$. We therefore get:

$$\begin{aligned} 2 \cdot \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle &= 2 \cdot \left(\frac{1}{2}m_1 + e_1 + I_1 \right) \cdot \left(\frac{1}{2}m_2 + e_2 + I_2 \right) \\ &= \frac{1}{2}m_1m_2 + 2(e_1I_2 + e_2I_1) + e_1m_2 + e_2m_1 + 2e_1e_2 \\ &\quad + \underbrace{(m_1I_2 + m_2I_1 + 2I_1I_2)}_{\in \mathbb{Z}} . \end{aligned}$$

Interestingly, the cross-term e_1e_2 that was responsible for the squaring of the noise in previous schemes, is now practically insignificant since $\epsilon^2 \ll \epsilon$. The significant noise term in the above expression is $2(e_1I_2 + e_2I_1)$, which is bounded by $O(\|\mathbf{s}\|_1) \cdot \epsilon$. All that is left to show now is that $\|\mathbf{s}\|_1$ is independent of B, q and only depends on n (recall that we allow dependence on $\log q \leq n$).

On the face of it, $\|\mathbf{s}\|_1 \approx n \cdot q$, since the elements of \mathbf{s} are integers in the segment $(-q/2, q/2]$. In order to reduce the norm, we use *binary decomposition* (which was used in [3,5] for different purposes). Let $\mathbf{s}^{(j)}$ denote the binary vector that contains the j^{th} bit from each element of \mathbf{s} . Namely $\mathbf{s} = \sum_j 2^j \mathbf{s}^{(j)}$. Then

$$\langle \mathbf{c}, \mathbf{s} \rangle = \sum_j 2^j \langle \mathbf{c}, \mathbf{s}^{(j)} \rangle = \langle (\mathbf{c}, 2\mathbf{c}, \dots), (\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \dots) \rangle .$$

This means that we can convert a ciphertext \mathbf{c} that corresponds to a secret key \mathbf{s} in \mathbb{Z} , into a modified ciphertext $(\mathbf{c}, 2\mathbf{c}, \dots)$ that corresponds to a *binary* secret

key $(\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \dots)$. The norm of the binary key is at most its dimension, which is polynomial in n as required.⁶

We point out that an alternative solution to the norm problem follows by using the dual-Regev scheme of [13] as the basic building block. There, the secret key is natively binary and of low norm. (In addition, as noticed in previous works, working with dual-Regev naturally implies a weak form of homomorphic identity based encryption.) However, the ciphertexts and some other parameters will need to grow.

Finally, working with fractional ciphertexts brings about issues of precision in representation and other problems. We thus implement our scheme over \mathbb{Z} with appropriate scaling: Each rational number x in the above description will be represented by the integer $y = \lfloor qx \rfloor$ (which determines x up to an additive factor of $1/2q$). The addition operation $x_1 + x_2$ is mimicked by $y_1 + y_2 \approx \lfloor q(x_1 + x_2) \rfloor$. To mimic multiplication, we take $\lfloor (y_1 \cdot y_2)/q \rfloor \approx \lfloor x_1 \cdot x_2 \cdot q \rfloor$. Our tensored ciphertext for multiplication will thus be defined as $\left[\frac{2}{q} \cdot \mathbf{c}_1 \otimes \mathbf{c}_2 \right]$, where $\mathbf{c}_1, \mathbf{c}_2$ are integer vectors and the tensoring operation is over the integers. In this representation, encryption and decryption become identical to Regev's original scheme.

1.3 Notation

For an integer q , we define the set $\mathbb{Z}_q \triangleq (-q/2, q/2] \cap \mathbb{Z}$. We stress that in this work, \mathbb{Z}_q is *not synonymous* with the ring $\mathbb{Z}/q\mathbb{Z}$. In particular, all arithmetics is performed over \mathbb{Z} (or \mathbb{Q} when division is used) and not over any sub-ring. For any $x \in \mathbb{Q}$, we let $y = [x]_q$ denote the unique value $y \in (-q/2, q/2]$ such that $y = x \pmod{q}$ (i.e. y is congruent to x modulo q).⁷

A distribution χ over the integers is B -bounded, denoted $|\chi| \leq B$, if χ is only supported on $[-B, B]$.

We denote vectors in bold lowercase (\mathbf{x}) and matrices in bold uppercase (\mathbf{A}). Slightly abusing notation, we denote the concatenation of vectors \mathbf{x}, \mathbf{y} by (\mathbf{x}, \mathbf{y}) . The tensor product of two vectors \mathbf{v}, \mathbf{w} of dimension n , denoted $\mathbf{v} \otimes \mathbf{w}$, is the n^2 dimensional vector containing all elements of the form $\mathbf{v}[i]\mathbf{w}[j]$. Note that $\langle \mathbf{v} \otimes \mathbf{w}, \mathbf{x} \otimes \mathbf{y} \rangle = \langle \mathbf{v}, \mathbf{x} \rangle \cdot \langle \mathbf{w}, \mathbf{y} \rangle$.

1.4 Paper Organization

Section 2 introduces the LWE assumption and defines homomorphic encryption and related terms. Section 3 introduces our building blocks: Regev's encryption scheme, binary decomposition of vectors and the key switching mechanism. Finally, in Section 4 we present and analyze our scheme, and discuss several possible optimizations.

⁶ Reducing the norm of \mathbf{s} was also an issue in [3]. There it was resolved by using LWE in Hermite normal form, where \mathbf{s} is sampled from the noise distribution and thus $\|\mathbf{s}\|_1 \approx n \cdot B$. This suffices when B must be very small, as in [3], but not in our setting.

⁷ For example, if $x = 2, y = -3 \in \mathbb{Z}_7$, then $x \cdot y = -6 \notin \mathbb{Z}_7$, however $[x \cdot y]_7 = 1 \in \mathbb{Z}_7$.

2 Preliminaries

2.1 Learning With Errors (LWE)

The LWE problem was introduced by Regev [22] as a generalization of “learning parity with noise”. For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on \mathbb{Z} , let $A_{\mathbf{s}, \chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly at random and a noise term $e \xleftarrow{\$} \chi$, and outputting $(\mathbf{a}, [\langle \mathbf{a}, \mathbf{s} \rangle + e]_q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Decisional LWE (DLWE) is defined as follows.

Definition 2.1 (DLWE). *For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over \mathbb{Z} , the (average-case) decision learning with errors problem, denoted $\text{DLWE}_{n,m,q,\chi}$, is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s}, \chi}$ (for uniformly random $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote by $\text{DLWE}_{n,q,\chi}$ the variant where the adversary gets oracle access to $A_{\mathbf{s}, \chi}$, and is not a-priori bounded in the number of samples.*

There are known quantum (Regev [22]) and classical (Peikert [20]) reductions between $\text{DLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be (discretized versions of) the Gaussian distribution, which is statistically indistinguishable from B -bounded, for an appropriate B . Since the exact distribution χ does not matter for our results, we state a corollary of the results of [20, 22] (in conjunction with the search to decision reduction of Micciancio and Mol [16] and Micciancio and Peikert [17]) in terms of the bound B . These results also extend to additional forms of q (see [16, 17]).

Corollary 2.2 ([16, 17, 20, 22]). *Let $q = q(n) \in \mathbb{N}$ be either a prime power $q = p^r$, or a product of co-prime numbers $q = \prod q_i$ such that for all i , $q_i = \text{poly}(n)$, and let $B \geq \omega(\log n) \cdot \sqrt{n}$. Then there exists an efficiently sampleable B -bounded distribution χ such that if there is an efficient algorithm that solves the (average-case) $\text{DLWE}_{n,q,\chi}$ problem. Then:*

- There is an efficient quantum algorithm that solves $\text{GapSVP}_{\tilde{O}(n \cdot q/B)}$ (and $\text{SIVP}_{\tilde{O}(n \cdot q/B)}$) on any n -dimensional lattice.
- If in addition $q \geq \tilde{O}(2^{n/2})$, then there is an efficient classical algorithm for $\text{GapSVP}_{\tilde{O}(n \cdot q/B)}$ on any n -dimensional lattice.

In both cases, if one also considers distinguishers with sub-polynomial advantage, then we require $B \geq \tilde{O}(n)$ and the resulting approximation factor is slightly larger $\tilde{O}(n\sqrt{n} \cdot q/B)$.

Recall that GapSVP_γ is the (promise) problem of distinguishing, given a basis for a lattice and a parameter d , between the case where the lattice has a vector shorter than d , and the case where the lattice doesn’t have any vector shorter than $\gamma \cdot d$. SIVP is the search problem of finding a set of “short” vectors. We refer the reader to [20, 22] for more information.

The best known algorithms for GapSVP_γ ([18, 25]) require at least $2^{\tilde{\Omega}(n/\log \gamma)}$ time. The scheme we present in this work reduces from $\gamma = n^{O(\log n)}$, for which the best known algorithms run in time $2^{\tilde{\Omega}(n)}$.

As a final remark, we mention that Peikert also shows a classical reduction in the case of small values of q , but this reduction is from a newly defined “ ζ -to- γ decisional shortest vector problem”, which is not as extensively studied as GapSVP .

2.2 Homomorphic Encryption and Bootstrapping

We now define homomorphic encryption and introduce Gentry’s bootstrapping theorem. Our definitions are mostly taken from [3, 5].

A homomorphic (public-key) encryption scheme $\mathbf{HE} = (\mathbf{HE.Keygen}, \mathbf{HE.Enc}, \mathbf{HE.Dec}, \mathbf{HE.Eval})$ is a quadruple of PPT algorithms as follows (n is the security parameter):

- **Key generation** $(pk, evk, sk) \leftarrow \mathbf{HE.Keygen}(1^n)$: Outputs a public encryption key pk , a public evaluation key evk and a secret decryption key sk .⁸
- **Encryption** $c \leftarrow \mathbf{HE.Enc}_{pk}(m)$: Using the public key pk , encrypts a single bit message $m \in \{0, 1\}$ into a ciphertext c .
- **Decryption** $m \leftarrow \mathbf{HE.Dec}_{sk}(c)$: Using the secret key sk , decrypts a ciphertext c to recover the message $m \in \{0, 1\}$.
- **Homomorphic evaluation** $c_f \leftarrow \mathbf{HE.Eval}_{evk}(f, c_1, \dots, c_\ell)$: Using the evaluation key evk , applies a function $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ to c_1, \dots, c_ℓ , and outputs a ciphertext c_f .

As in previous works, we represent f as an arithmetic circuit over $GF(2)$ with addition and multiplication gates. Thus it is customary to “break” $\mathbf{HE.Eval}$ into homomorphic addition $c_{\text{add}} \leftarrow \mathbf{HE.Add}_{evk}(c_1, c_2)$ and homomorphic multiplication $c_{\text{mult}} \leftarrow \mathbf{HE.Mult}_{evk}(c_1, c_2)$.

A homomorphic encryption scheme is said to be secure if it is semantically secure (note that the adversary is given both pk and evk).

Homomorphism w.r.t depth-bounded circuits and full homomorphism are defined next:

Definition 2.3 (L-homomorphism). A scheme \mathbf{HE} is L-homomorphic, for $L = L(n)$, if for any depth L arithmetic circuit f (over $GF(2)$) and any set of inputs m_1, \dots, m_ℓ , it holds that

$$\Pr[\mathbf{HE.Dec}_{sk}(\mathbf{HE.Eval}_{evk}(f, c_1, \dots, c_\ell)) \neq f(m_1, \dots, m_\ell)] = \text{negl}(n),$$

where $(pk, evk, sk) \leftarrow \mathbf{HE.Keygen}(1^n)$ and $c_i \leftarrow \mathbf{HE.Enc}_{pk}(m_i)$.

Definition 2.4 (compactness and full homomorphism). A homomorphic scheme is compact if its decryption circuit is independent of the evaluated function. A compact scheme is fully homomorphic if it is L-homomorphic for any

⁸ We adopt the terminology of [5] that treats the evaluation key as a separate entity from the public key.

polynomial L . The scheme is leveled fully homomorphic if it takes 1^L as additional input in key generation.

Gentry's bootstrapping theorem shows how to go from L -homomorphism to full homomorphism:

Theorem 2.5 (bootstrapping [8, 9]). *If there is an L -homomorphic scheme whose decryption circuit depth is less than L , then there exists a leveled fully homomorphic encryption scheme.*

Furthermore, if the aforementioned L -homomorphic scheme is also weak circular secure (remains secure even against an adversary who gets encryptions of the bits of the secret key), then there exists a fully homomorphic encryption scheme.

3 Building Blocks

In this section, we present building blocks from previous works that are used in our construction. Specifically, like all LWE-based fully homomorphic schemes, we rely on Regev's [22] basic public-key encryption scheme (Section 3.1). We also use the key-switching methodology of [3, 5] (Section 3.2).

3.1 Regev's Encryption Scheme

Let $q = q(n)$ be an integer function and let $\chi = \chi(n)$ be a distribution ensemble over \mathbb{Z} . The scheme Regev is defined as follows:

- Regev.SecretKeygen(1^n): Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. Output $sk = \mathbf{s}$.
- Regev.PublicKeygen(\mathbf{s}): Let $N \triangleq (n+1) \cdot (\log q + O(1))$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{N \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^N$. Compute $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and define

$$\mathbf{P} := [\mathbf{b} \| -\mathbf{A}] \in \mathbb{Z}_q^{N \times (n+1)}.$$

Output $pk = \mathbf{P}$.

- Regev.Enc_{pk}(m): To encrypt a message $m \in \{0, 1\}$ using $pk = \mathbf{P}$, sample $\mathbf{r} \in \{0, 1\}^N$ and output ciphertext

$$\mathbf{c} := \left[\mathbf{P}^T \cdot \mathbf{r} + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathbf{m} \right]_q \in \mathbb{Z}_q^{n+1},$$

where $\mathbf{m} \triangleq (m, 0, \dots, 0) \in \{0, 1\}^{n+1}$.

- Regev.Dec_{sk}(\mathbf{c}): To decrypt $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ using secret key $sk = \mathbf{s}$, compute

$$m := \left[\left\lfloor 2 \cdot \frac{[\langle \mathbf{c}, (1, \mathbf{s}) \rangle]_q}{q} \right\rfloor \right]_2.$$

Correctness. We analyze the noise magnitude at encryption and decryption. We start with a lemma regarding the noise magnitude of properly encrypted ciphertexts (the proof is omitted).

Lemma 3.1 (encryption noise). Let $q, n, N, |\chi| \leq B$ be parameters for Regev. Let $\mathbf{s} \in \mathbb{Z}^n$ be any vector and $m \in \{0, 1\}$ be a bit. Set $\mathbf{P} \leftarrow \text{Regev.PublicKeygen}(\mathbf{s})$ and $\mathbf{c} \leftarrow \text{Regev.Enc}_{\mathbf{P}}(m)$. Then for some e with $|e| \leq N \cdot B$ it holds that

$$\langle \mathbf{c}, (1, \mathbf{s}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot m + e \pmod{q}.$$

We proceed to state the correctness of decryption for low-noise ciphertexts. The proof easily follows by assignment into the definition of Regev.Dec and is omitted.

Lemma 3.2 (decryption noise). Let $\mathbf{s} \in \mathbb{Z}^n$ be some vector, and let $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ be such that

$$\langle \mathbf{c}, (1, \mathbf{s}) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot m + e \pmod{q},$$

with $m \in \{0, 1\}$ and $|e| < \lfloor q/2 \rfloor / 2$. Then

$$\text{Regev.Dec}_{\mathbf{s}}(\mathbf{c}) = m.$$

Security. The following lemma states the security of Regev. The proof is standard (see e.g. [22]) and is omitted.

Lemma 3.3. Let n, q, χ be some parameters such that $\text{DLWE}_{n, q, \chi}$ holds. Then for any $m \in \{0, 1\}$, if $\mathbf{s} \leftarrow \text{Regev.SecretKeygen}(1^n)$, $\mathbf{P} \leftarrow \text{Regev.PublicKeygen}(\mathbf{s})$, $\mathbf{c} \leftarrow \text{Regev.Enc}_{\mathbf{P}}(m)$, it holds that the joint distribution (\mathbf{P}, \mathbf{c}) is computationally indistinguishable from uniform over $\mathbb{Z}_q^{N \times (n+1)} \times \mathbb{Z}_q^{n+1}$.

3.2 Vector Decomposition and Key Switching

We show how to decompose vectors in a way that preserves inner product and how to generate and use key switching parameters. Our notation is generally adopted from [3].

Vector Decomposition. We often break vectors into their bit representations as defined below:

- $\text{BitDecomp}_q(\mathbf{x})$: For $\mathbf{x} \in \mathbb{Z}^n$, let $\mathbf{w}_i \in \{0, 1\}^n$ be such that $\mathbf{x} = \sum_{i=0}^{\lceil \log q \rceil - 1} 2^i \cdot \mathbf{w}_i \pmod{q}$. Output the vector

$$(\mathbf{w}_0, \dots, \mathbf{w}_{\lceil \log q \rceil - 1}) \in \{0, 1\}^{n \cdot \lceil \log q \rceil}.$$

- $\text{PowersOfTwo}_q(\mathbf{y})$: For $\mathbf{y} \in \mathbb{Z}^n$, output

$$\left[(\mathbf{y}, 2 \cdot \mathbf{y}, \dots, 2^{\lceil \log q \rceil - 1} \cdot \mathbf{y}) \right]_q \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil}.$$

We will usually omit the subscript q when it is clear from the context.

Claim 3.4. For all $q \in \mathbb{Z}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, it holds that

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \text{BitDecomp}_q(\mathbf{x}), \text{PowersOfTwo}_q(\mathbf{y}) \rangle \pmod{q}.$$

Key Switching. In the functions below, q is an integer and χ is a distribution over \mathbb{Z} :

- $\text{SwitchKeyGen}_{q,\chi}(\mathbf{s}, \mathbf{t})$: For a “source” key $\mathbf{s} \in \mathbb{Z}^{n_s}$ and “target” key $\mathbf{t} \in \mathbb{Z}^{n_t}$, we define a set of parameters that allow to switch ciphertexts under \mathbf{s} into ciphertexts under $(1, \mathbf{t})$.

Let $\hat{n}_s \triangleq n_s \cdot \lceil \log q \rceil$ be the dimension of $\text{PowersOfTwo}_q(\mathbf{s})$. Sample a uniform matrix $\mathbf{A}_{\mathbf{s}:\mathbf{t}} \xleftarrow{\$} \mathbb{Z}_q^{\hat{n}_s \times n_t}$ and a noise vector $\mathbf{e} \xleftarrow{\$} \chi^{\hat{n}_s}$. The function’s output is a matrix

$$\mathbf{P}_{\mathbf{s}:\mathbf{t}} = [\mathbf{b}_{\mathbf{s}:\mathbf{t}} \| -\mathbf{A}_{\mathbf{s}:\mathbf{t}}] \in \mathbb{Z}_q^{\hat{n}_s \times (n_t+1)},$$

where

$$\mathbf{b}_{\mathbf{s}:\mathbf{t}} := \left[\mathbf{A}_{\mathbf{s}:\mathbf{t}} \cdot \mathbf{t} + \mathbf{e}_{\mathbf{s}:\mathbf{t}} + \text{PowersOfTwo}_q(\mathbf{s}) \right]_q \in \mathbb{Z}_q^{\hat{n}_s}.$$

This is similar, although not identical, to encrypting $\text{PowersOfTwo}_q(\mathbf{s})$ (the difference is that $\text{PowersOfTwo}_q(\mathbf{s})$ contains non-binary values).

- $\text{SwitchKey}_q(\mathbf{P}_{\mathbf{s}:\mathbf{t}}, \mathbf{c}_s)$: To switch a ciphertext from a secret key \mathbf{s} to $(1, \mathbf{t})$, output

$$\mathbf{c}_t := [\mathbf{P}_{\mathbf{s}:\mathbf{t}}^T \cdot \text{BitDecomp}_q(\mathbf{c}_s)]_q.$$

Again, we usually omit the subscripts when they are clear from the context. Correctness and security are stated below, the proofs are by definition.

Lemma 3.5 (correctness). Let $\mathbf{s} \in \mathbb{Z}^{n_s}$, $\mathbf{t} \in \mathbb{Z}^{n_t}$ and $\mathbf{c}_s \in \mathbb{Z}_q^{n_s}$ be any vectors. Let $\mathbf{P}_{\mathbf{s}:\mathbf{t}} \leftarrow \text{SwitchKeyGen}(\mathbf{s}, \mathbf{t})$ and set $\mathbf{c}_t \leftarrow \text{SwitchKey}(\mathbf{P}_{\mathbf{s}:\mathbf{t}}, \mathbf{c}_s)$. Then

$$\langle \mathbf{c}_s, \mathbf{s} \rangle = \langle \mathbf{c}_t, (1, \mathbf{t}) \rangle - \langle \text{BitDecomp}_q(\mathbf{c}_s), \mathbf{e}_{\mathbf{s}:\mathbf{t}} \rangle \pmod{q}.$$

Lemma 3.6 (security). Let $\mathbf{s} \in \mathbb{Z}^{n_s}$ be arbitrary, $\mathbf{t} \leftarrow \text{Regev.SecretKeygen}(1^n)$ and $\mathbf{P} \leftarrow \text{SwitchKeyGen}_{q,\chi}(\mathbf{s}, \mathbf{t})$. Then \mathbf{P} is computationally indistinguishable from uniform over $\mathbb{Z}_q^{\hat{n}_s \times (n_t+1)}$, assuming $\text{DLWE}_{n,q,\chi}$.

4 A Scale Invariant Homomorphic Encryption Scheme

We present our scale invariant L -homomorphic scheme as outlined in Section 1.2. Homomorphic properties are discussed in Section 4.1, implications and optimizations are discussed in Section 4.2.

Let $q = q(n)$ be an integer function, let $L = L(n)$ be a polynomial and let $\chi = \chi(n)$ be a distribution ensemble over \mathbb{Z} . The scheme SI-HE is defined as follows:

- $\text{SI-HE.Keygen}(1^L, 1^n)$: Sample vectors $\mathbf{s}_0, \dots, \mathbf{s}_L \leftarrow \text{Regev.SecretKeygen}(1^n)$. Compute a Regev public key for the first one: $\mathbf{P}_0 \leftarrow \text{Regev.PublicKeygen}(\mathbf{s}_0)$. For all $i \in [L]$, define

$$\tilde{\mathbf{s}}_{i-1} := \text{BitDecomp}((1, \mathbf{s}_{i-1})) \otimes \text{BitDecomp}((1, \mathbf{s}_{i-1})) \in \{0, 1\}^{((n+1)\lceil \log q \rceil)^2}.$$

and compute

$$\mathbf{P}_{(i-1):i} \leftarrow \text{SwitchKeyGen}(\tilde{\mathbf{s}}_{i-1}, \mathbf{s}_i).$$

Output $pk = \mathbf{P}_0$, $evk = \{\mathbf{P}_{(i-1):i}\}_{i \in [L]}$ and $sk = \mathbf{s}_L$.

- SI-HE. $\text{Enc}_{pk}(m)$: Identical to Regev’s, output $\mathbf{c} \leftarrow \text{Regev}.\text{Enc}_{pk}(m)$.
- SI-HE. $\text{Eval}_{evk}(\cdot)$: As usual, we describe homomorphic addition and multiplication over GF(2), which allows to evaluate depth L arithmetic circuits in a gate-by-gate manner. The convention for a gate at level i of the circuit is that the operand ciphertexts are decryptable using \mathbf{s}_{i-1} , and the output of the homomorphic operation is decryptable using \mathbf{s}_i .

Since evk contains key switching parameters from $\tilde{\mathbf{s}}_{i-1}$ to \mathbf{s}_i , homomorphic addition and multiplication both first produce an intermediate output $\tilde{\mathbf{c}}$ that corresponds to $\tilde{\mathbf{s}}_{i-1}$, and then use key switching to obtain the final output.⁹

- SI-HE. $\text{Add}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$: Assume w.l.o.g that both input ciphertexts are encrypted under the same secret key \mathbf{s}_{i-1} . First compute

$$\tilde{\mathbf{c}}_{\text{add}} := \text{PowersOfTwo}(\mathbf{c}_1 + \mathbf{c}_2) \otimes \text{PowersOfTwo}((1, 0, \dots, 0)) ,$$

then output

$$\mathbf{c}_{\text{add}} \leftarrow \text{SwitchKey}(\mathbf{P}_{(i-1):i}, \tilde{\mathbf{c}}_{\text{add}}) \in \mathbb{Z}_q^{n+1} .$$

Let us explain what we did: We first added the ciphertext vectors (as expected) to obtain $\mathbf{c}_1 + \mathbf{c}_2$. This already implements the homomorphic addition, but provides an output that corresponds to \mathbf{s}_{i-1} and not \mathbf{s}_i as required. We thus generate $\tilde{\mathbf{c}}_{\text{add}}$ by tensoring with a “trivial” ciphertext. The result corresponds to $\tilde{\mathbf{s}}_{i-1}$, and allows to finally use key switching to obtain an output corresponding to \mathbf{s}_i . We use powers-of-two representation in order to control the norm of the secret key (as we explain in Section 1.2).

- SI-HE. $\text{Mult}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$: Assume w.l.o.g that both input ciphertexts are encrypted under the same secret key \mathbf{s}_{i-1} . First compute

$$\tilde{\mathbf{c}}_{\text{mult}} := \left\lfloor \frac{2}{q} \cdot \left(\text{PowersOfTwo}(\mathbf{c}_1) \otimes \text{PowersOfTwo}(\mathbf{c}_2) \right) \right\rfloor ,$$

then output

$$\mathbf{c}_{\text{mult}} \leftarrow \text{SwitchKey}(\mathbf{P}_{(i-1):i}, \tilde{\mathbf{c}}_{\text{mult}}) \in \mathbb{Z}_q^{n+1} .$$

As we explain in Section 1.2, The tensored ciphertext $\tilde{\mathbf{c}}_{\text{mult}}$ mimics tensoring in the “invariant perspective”, which produces an encryption of the product of the plaintexts under the tensored secret key $\tilde{\mathbf{s}}_{i-1}$. We then switch keys to obtain an output corresponding to \mathbf{s}_i .

- Decryption SI-HE. $\text{Dec}_{sk}(\mathbf{c})$: Assume w.l.o.g that \mathbf{c} is a ciphertext that corresponds to $\mathbf{s}_L (=sk)$. Then decryption is again identical to Regev’s, output

$$m \leftarrow \text{Regev}.\text{Dec}_{sk}(\mathbf{c}) .$$

⁹ The final key switching replaces the more complicated “refresh” operation of [3].

Security. The security of the scheme follows in a straightforward way, very similarly to the proof of [5, Theorem 4.1]. The proof is omitted.

Lemma 4.1. *Let n, q, χ be some parameters such that $\text{DLWE}_{n,q,\chi}$ holds, and let $L = L(n)$ be polynomially bounded. Let $(pk, evk, sk) \leftarrow \text{SI-HE.Keygen}(1^L, 1^n)$, $\mathbf{c} \leftarrow \text{SI-HE.Enc}_{pk}(m)$ (for any $m \in \{0, 1\}$), then it holds that the joint distribution (pk, evk, \mathbf{c}) is computationally indistinguishable from uniform.*

4.1 Homomorphic Properties of SI-HE

The following theorem summarizes the homomorphic properties of our scheme.

Theorem 4.2. *The scheme SI-HE with parameters $n, q, |\chi| \leq B, L$ for which*

$$q/B \geq (O(n \log q))^{L+O(1)},$$

is L -homomorphic.

The theorem is proven using the following lemma, which bounds the growth of the noise in gate evaluation.

Lemma 4.3. *Let $q, n, |\chi| \leq B, L$ be parameters for SI-HE. Set (pk, evk, sk) by calling $\text{SI-HE.Keygen}(1^L, 1^n)$ and let $\mathbf{c}_1, \mathbf{c}_2$ be such that*

$$\begin{aligned} \langle \mathbf{c}_1, (1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 \pmod{q} \\ \langle \mathbf{c}_2, (1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_2 \pmod{q}, \end{aligned} \quad (1)$$

with $|e_1|, |e_2| \leq E < \lfloor q/2 \rfloor / 2$. Consider ciphertexts $\mathbf{c}_{\text{add}} \leftarrow \text{SI-HE.Add}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$, $\mathbf{c}_{\text{mult}} \leftarrow \text{SI-HE.Mult}_{evk}(\mathbf{c}_1, \mathbf{c}_2)$. Then

$$\begin{aligned} \langle \mathbf{c}_{\text{add}}, (1, \mathbf{s}_i) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot ([m_1 + m_2]_2) + e_{\text{add}} \pmod{q} \\ \langle \mathbf{c}_{\text{mult}}, (1, \mathbf{s}_i) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 m_2 + e_{\text{mult}} \pmod{q}, \end{aligned}$$

where

$$|e_{\text{add}}|, |e_{\text{mult}}| \leq O(n \log q) \cdot \max \{E, (n \log^2 q) \cdot B\}.$$

We remark that, as usual, homomorphic addition increases noise much more moderately than multiplication, but the coarse bound we show in the lemma is sufficient for our purposes.

Theorem 4.2 follows from Lemma 4.3 in a straightforward manner. The proof of Lemma 4.3 appears in Section 4.3.

4.2 Implications and Optimizations

Fully Homomorphic Encryption using Bootstrapping. Fully homomorphic encryption follows using the bootstrapping theorem (Theorem 2.5). In order to use bootstrapping, we need to bound the depth of the decryption circuit. The following lemma has been proven in a number of previous works (e.g. [5, Lemma 4.5]):

Lemma 4.4. *For all \mathbf{c} , the function $f_{\mathbf{c}}(\mathbf{s}) = \text{SI-HE.Dec}_{\mathbf{s}}(\mathbf{c})$ can be implemented by a circuit of depth $O(\log n + \log \log q)$.*

An immediate corollary follows from Theorem 2.5, Theorem 4.2, Lemma 4.4:

Corollary 4.5. *Let $n, q, |\chi| \leq B$ be such that $q/B \geq (n \log q)^{O(\log n + \log \log q)}$. Then there exists a (leveled) fully homomorphic encryption scheme based on the $\text{DLWE}_{n,q,\chi}$ assumption.*

Furthermore, if SI-HE is weak circular secure, then the same assumption implies full (non leveled) homomorphism.

Finally, we can classically reduce security from GapSVP using Corollary 2.2, by choosing $q = \tilde{O}(2^{n/2})$ and $B = q/(n \log q)^{O(\log n + \log \log q)} = q/n^{O(\log n)}$:

Corollary 4.6. *There exists a (leveled) fully-homomorphic encryption scheme based on the classical worst case hardness of the $\text{GapSVP}_{n^{O(\log n)}}$ problem.*

(Leveled) Fully Homomorphic Encryption without Bootstrapping. As in [3], our scheme implies a leveled fully homomorphic encryption without bootstrapping. Plugging our scheme into the [3] framework, we obtain a (leveled) fully homomorphic encryption without bootstrapping, based on the classical worst case hardness of $\text{GapSVP}_{2^{n^\epsilon}}$, for any $\epsilon > 0$.

Optimizations. So far, we chose to present our scheme in the cleanest possible way. However, there are a few techniques that can somewhat improve performance. While the asymptotic advantage of some of these methods is not great, a real life implementation can benefit from them.

1. Our tensored secret key $\tilde{\mathbf{s}}_{i-1}$ is obtained by tensoring a vector with itself. Such a vector can be represented by only $\binom{n_s}{2}$ (as opposed to our n_s^2), saving a factor of (almost) 2 in the representation length.
2. When $B \ll q$, some improvement can be achieved by using LWE in Hermite normal form. It is known (see e.g. [2]) that the hardness of LWE remains essentially the same if we sample $\mathbf{s} \xleftarrow{\$} \chi^n$ (instead of uniformly in \mathbb{Z}_q^n). Sampling our keys this way, we only need $O(n \log B)$ bits to represent $\text{BitDecomp}(\mathbf{s})$, and its norm goes down accordingly.

We can therefore reduce the size of the evaluation key (which depends quadratically on the bit length of the secret key), and more importantly, we can prove a tighter version of Lemma 4.3. When using Hermite normal form, the noise grows from E to $O(n \log B) \cdot \max\{E, (n \log B \log q) \cdot B\}$. Therefore, L -homomorphism is achieved whenever

$$q/B \geq (O(n \log B))^{L+O(1)} \cdot \log q .$$

3. The least significant bits of the ciphertext can sometimes be truncated without much harm, which can lead to significant saving in ciphertext and key length, especially when B/q is large: Let \mathbf{c} be an integer ciphertext vector

and define $\mathbf{c}' = \lfloor 2^{-i} \cdot \mathbf{c} \rfloor$. Then \mathbf{c}' , which can be represented with $n \cdot i$ fewer bits than \mathbf{c} , implies a good approximation for \mathbf{c} since

$$|\langle \mathbf{c}, \mathbf{s} \rangle - \langle 2^i \cdot \mathbf{c}', \mathbf{s} \rangle| \leq 2^{i-1} \|\mathbf{s}\|_1 .$$

This means that $2^i \cdot \mathbf{c}'$ can be used instead of \mathbf{c} , at the cost of an additive increase in the noise magnitude.

Consider a case where $q, B \gg q/B$ (which occurs when we artificially increase q in order for the classical reduction to work). Recall that $\|\mathbf{s}\|_1 \approx n \log q$ and consider truncating with $i \approx \log(B/(n \log q))$. Then the additional noise incurred by using \mathbf{c}' instead of \mathbf{c} is only an insignificant $\approx B$. The number of bits required to represent each element in \mathbf{c}' however now becomes $\log q - i \approx \log(q/B) + \log(n \log q)$. In conclusion, we hardly lose anything in ciphertext length compared to the case of working with smaller q, B to begin with (with similar q/B ratio). The ciphertext length can, therefore, be made invariant to the absolute values of q, B , and depend only on their ratio. This of course applies also to the vectors in evk .

4.3 Proof of Lemma 4.3

We start with the analysis for addition, which is simpler and will also serve as good warm-up towards the analysis for multiplication.

Analysis for Addition. By Lemma 3.5, it holds that

$$\langle \mathbf{c}_{\text{add}}, (1, \mathbf{s}_i) \rangle = \langle \tilde{\mathbf{c}}_{\text{add}}, \tilde{\mathbf{s}}_i \rangle + \underbrace{\langle \text{BitDecomp}(\tilde{\mathbf{c}}), \mathbf{e}_{i-1:i} \rangle}_{\triangleq \delta_1} \pmod{q} .$$

where $\mathbf{e}_{i-1:i} \sim \chi^{(n+1)^2 \cdot (\lceil \log q \rceil)^3}$. That is, δ_1 is the noise inflicted by the key switching process.

We bound $|\delta_1|$ using the bound on χ :

$$|\delta_1| = |\langle \text{BitDecomp}(\tilde{\mathbf{c}}_{\text{add}}), \mathbf{e}_{i-1:i} \rangle| \leq (n+1)^2 \cdot (\lceil \log q \rceil)^3 \cdot B = O(n^2 \log^3 q) \cdot B .$$

Next, we expand the term $\langle \tilde{\mathbf{c}}_{\text{add}}, \tilde{\mathbf{s}}_i \rangle$, by breaking an inner product of tensors into a product of inner products (one of which is trivially equal to 1):

$$\begin{aligned} \langle \tilde{\mathbf{c}}_{\text{add}}, \tilde{\mathbf{s}}_i \rangle &= \left\langle \text{PowersOfTwo}(\mathbf{c}_1 + \mathbf{c}_2) \otimes \text{PowersOfTwo}((1, 0, \dots, 0)), \right. \\ &\quad \left. \text{BitDecomp}((1, \mathbf{s}_{i-1})) \otimes \text{BitDecomp}((1, \mathbf{s}_{i-1})) \right\rangle \\ &= \langle \text{PowersOfTwo}(\mathbf{c}_1 + \mathbf{c}_2), \text{BitDecomp}((1, \mathbf{s}_{i-1})) \rangle \cdot 1 \\ &= \langle (\mathbf{c}_1 + \mathbf{c}_2), (1, \mathbf{s}_{i-1}) \rangle \pmod{q} \\ &= \langle \mathbf{c}_1, (1, \mathbf{s}_{i-1}) \rangle + \langle \mathbf{c}_2, (1, \mathbf{s}_{i-1}) \rangle \pmod{q} . \end{aligned}$$

We can now plug in what we know about $\mathbf{c}_1, \mathbf{c}_2$ from Eq. (1) in the lemma statement:

$$\begin{aligned}\langle \tilde{\mathbf{c}}_{\text{add}}, \tilde{\mathbf{s}}_i \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 + \left\lfloor \frac{q}{2} \right\rfloor \cdot m_2 + e_2 \pmod{q} \\ &= \left\lfloor \frac{q}{2} \right\rfloor \cdot [m_1 + m_2]_2 \underbrace{- \tilde{m} + e_1 + e_2}_{\triangleq \delta_2}, \pmod{q}\end{aligned}$$

where $\tilde{m} \in \{0, 1\}$ is defined as:

$$\tilde{m} \triangleq \begin{cases} 0, & \text{if } q \text{ is even,} \\ \frac{1}{2} \cdot (m_1 + m_2 - [m_1 + m_2]_2), & \text{if } q \text{ is odd,} \end{cases}$$

and $|\delta_2| \leq 1 + 2E$.

Putting it all together,

$$\langle \mathbf{c}_{\text{add}}, (1, \mathbf{s}_i) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot [m_1 + m_2]_2 + \underbrace{\delta_1 + \delta_2}_{=e_{\text{add}}} \pmod{q}.$$

Where the bound on e_{add} is

$$|e_{\text{add}}| = |\delta_1 + \delta_2| \leq O(n^2 \log^3 q) \cdot B + O(1) \cdot E \leq O(n \log q) \cdot \max \{E, (n \log^2 q) \cdot B\}$$

which finishes the argument for addition.

Analysis for Multiplication. The analysis for multiplication starts very similarly to addition:

$$\langle \mathbf{c}_{\text{mult}}, (1, \mathbf{s}_i) \rangle = \langle \tilde{\mathbf{c}}_{\text{mult}}, \tilde{\mathbf{s}}_i \rangle + \underbrace{\langle \text{BitDecomp}(\tilde{\mathbf{c}}_{\text{mult}}), \mathbf{e}_{i-1:i} \rangle}_{\triangleq \delta_1} \pmod{q},$$

and as before

$$|\delta_1| = O(n^2 \log^3 q) \cdot B.$$

Let us now focus on $\langle \tilde{\mathbf{c}}_{\text{mult}}, \tilde{\mathbf{s}}_i \rangle$. We want to use the properties of tensoring to break the inner product into two smaller inner products, as we did before. This time, however, $\tilde{\mathbf{c}}_{\text{mult}}$ is a *rounded* tensor:

$$\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}_i \rangle = \left\langle \left[\frac{2}{q} \cdot (\text{PowersOfTwo}(\mathbf{c}_1) \otimes \text{PowersOfTwo}(\mathbf{c}_2)) \right], \tilde{\mathbf{s}}_{i-1} \right\rangle \pmod{q}.$$

We start by showing that the rounding does not add much noise. Intuitively this is because $\tilde{\mathbf{s}}_{i-1}$ is a binary vector and thus has low norm. We define

$$\begin{aligned}\delta_2 \triangleq & \left\langle \left[\frac{2}{q} \cdot (\text{PowersOfTwo}(\mathbf{c}_1) \otimes \text{PowersOfTwo}(\mathbf{c}_2)) \right], \tilde{\mathbf{s}}_{i-1} \right\rangle \\ & - \left\langle \frac{2}{q} \cdot (\text{PowersOfTwo}(\mathbf{c}_1) \otimes \text{PowersOfTwo}(\mathbf{c}_2)), \tilde{\mathbf{s}}_{i-1} \right\rangle,\end{aligned}$$

and for convenience we also define

$$\begin{aligned} \mathbf{c}' &\triangleq \left[\frac{2}{q} \cdot (\text{PowersOfTwo}(\mathbf{c}_1) \otimes \text{PowersOfTwo}(\mathbf{c}_2)) \right] \\ &\quad - \frac{2}{q} \cdot (\text{PowersOfTwo}(\mathbf{c}_1) \otimes \text{PowersOfTwo}(\mathbf{c}_2)) . \end{aligned}$$

By definition, $\delta_2 = \langle \mathbf{c}', \tilde{\mathbf{s}}_{i-1} \rangle$. We know that $\|\mathbf{c}'\|_\infty \leq 1/2$ and $\|\tilde{\mathbf{s}}_{i-1}\|_1 \leq ((n+1) \lceil \log q \rceil)^2 = O(n^2 \log^2 q)$, and therefore

$$|\delta_2| \leq \|\mathbf{c}'\|_\infty \cdot \|\tilde{\mathbf{s}}_{i-1}\|_1 = O(n^2 \log^2 q) .$$

We can now break the inner product using the properties of tensoring:

$$\begin{aligned} \langle \tilde{\mathbf{c}}_{\text{mult}}, \tilde{\mathbf{s}}_{i-1} \rangle - \delta_2 &= \frac{2}{q} \cdot \langle \text{PowersOfTwo}(\mathbf{c}_1), \text{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle \\ &\quad \cdot \langle \text{PowersOfTwo}(\mathbf{c}_2), \text{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle . \end{aligned} \quad (2)$$

We keep $\mathbf{c}_1, \mathbf{c}_2$ in powers-of-two form on purpose (it is useful to have ciphertexts that relate to low-norm secrets).

Going back to Eq. (1) from the lemma statement and using Claim 3.4, there exist $I_1, I_2 \in \mathbb{Z}$ such that

$$\begin{aligned} \langle \text{PowersOfTwo}(\mathbf{c}_1), \text{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 + q \cdot I_1 \\ \langle \text{PowersOfTwo}(\mathbf{c}_2), \text{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_2 + e_2 + q \cdot I_2 . \end{aligned} \quad (3)$$

Let us bound the absolute value of I_1 (obviously the same holds for I_2):

$$\begin{aligned} |I_1| &= \frac{|\langle \text{PowersOfTwo}(\mathbf{c}_1), \text{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle - \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 - e_1|}{q} \\ &\leq \frac{|\langle \text{PowersOfTwo}(\mathbf{c}_1), \text{BitDecomp}(1, \mathbf{s}_{i-1}) \rangle|}{q} + 1 \\ &\leq \frac{\|\text{PowersOfTwo}(\mathbf{c}_1)\|_\infty}{q} \cdot \|\text{BitDecomp}(1, \mathbf{s}_{i-1})\|_1 + 1 \\ &\leq \frac{1}{2} \cdot \|\text{BitDecomp}(1, \mathbf{s}_{i-1})\|_1 + 1 \\ &\leq \frac{1}{2} \cdot (n+1) \lceil \log q \rceil + 1 \\ &= O(n \log q) . \end{aligned} \quad (4)$$

Plugging Eq. (3) into Eq. (2), we get

$$\begin{aligned} \langle \tilde{\mathbf{c}}_{\text{mult}}, \tilde{\mathbf{s}}_{i-1} \rangle - \delta_2 &= \frac{2}{q} \cdot \left(\left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 + e_1 + q \cdot I_1 \right) \cdot \left(\left\lfloor \frac{q}{2} \right\rfloor \cdot m_2 + e_2 + q \cdot I_2 \right) \\ &= \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 \cdot m_2 + \delta_3 + q \cdot (m_1 I_2 + m_2 I_1 + 2I_1 I_2) , \end{aligned}$$

where δ_3 is defined as

$$\delta_3 \triangleq \begin{cases} 2e_2 \cdot I_1 + 2e_1 \cdot I_2 + (e_1 m_2 + e_2 m_1) + \frac{2e_1 \cdot e_2}{q}, & \text{if } q \text{ is even,} \\ (2e_2 - m_2) \cdot I_1 + (2e_1 - m_1) \cdot I_2 \\ \quad + \frac{q-1}{q} \cdot (e_1 m_2 + e_2 m_1) - \frac{m_1 \cdot m_2}{2q} + \frac{2e_1 \cdot e_2}{q}, & \text{if } q \text{ is odd.} \end{cases}$$

In particular (recall that $E < \lfloor q/2 \rfloor / 2 \leq q/4$):

$$|\delta_3| \leq 2(2E+1)O(n \log q) + 2E + \frac{1+2E^2}{q} = O(n \log q) \cdot E.$$

Putting everything together, we get that

$$\langle \mathbf{c}_{\text{mult}}, (1, \mathbf{s}_i) \rangle = \left\lfloor \frac{q}{2} \right\rfloor \cdot m_1 m_2 + \underbrace{\delta_1 + \delta_2 + \delta_3}_{=e_{\text{mult}}} \pmod{q},$$

where

$$|e_{\text{mult}}| = |\delta_1 + \delta_2 + \delta_3| \leq O(n \log q) \cdot E + O(n^2 \log^3 q) \cdot B,$$

and the lemma follows. \square

Acknowledgments. We thank Vinod Vaikuntanathan for fruitful discussions and advice, and Dan Boneh for his comments on an earlier version of this manuscript. We thank the reviewers of CRYPTO 2012 for their constructive comments. In addition, we thank various readers for pointing out typos in earlier versions of this manuscript.

References

1. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Leighton, F.T., Shor, P.W. (eds.) STOC, pp. 284–293. ACM (1997)
2. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
3. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS (2012), <http://eprint.iacr.org/2011/277>
4. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
5. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky (ed.) [19], pp. 97–106, References are to full version, <http://eprint.iacr.org/2011/344>
6. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway (ed.) [24], pp. 487–504
7. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
8. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), <http://crypto.stanford.edu/craig>

9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
10. Gentry, C., Halevi, S.: Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In: Ostrovsky (ed.) [19], pp. 107–109
11. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. IACR Cryptology ePrint Archive, 2011:680 (2011)
12. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. IACR Cryptology ePrint Archive, 2011:566 (2011)
13. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) STOC, pp. 197–206. ACM (2008)
14. Goldreich, O., Goldwasser, S., Halevi, S.: Eliminating Decryption Errors in the Ajtai-Dwork Cryptosystem. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 105–111. Springer, Heidelberg (1997)
15. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010); Draft of full version was provided by the authors
16. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In: Rogaway (ed.) [24], pp. 465–484
17. Micciancio, D., Peikert, C.: Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012)
18. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In: Schulman, L.J. (ed.) STOC, pp. 351–358. ACM (2010)
19. Ostrovsky, R.: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22–25. IEEE (2011)
20. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: STOC, pp. 333–342 (2009)
21. Regev, O.: New lattice based cryptographic constructions. In: Larmore, L.L., Goemans, M.X. (eds.) STOC, pp. 407–416. ACM (2003); Full version in J. ACM 51(6) (2004)
22. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) STOC, pp. 84–93. ACM (2005); Full version in J. ACM 56(6) (2009)
23. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 169–177. Academic Press (1978)
24. Rogaway, P. (ed.): CRYPTO 2011. LNCS, vol. 6841. Springer, Heidelberg (2011)
25. Schnorr, C.-P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. 53, 201–224 (1987)
26. Smart, N.P., Vercauteren, F.: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)

Author Index

- Abdelraheem, Mohamed Ahmed 50
Agrawal, Shweta 443
Ågren, Martin 50
Alwen, Joël 124
Augier, Maxime 626

Bardou, Romain 608
Bay, Ash 741
Beelen, Peter 50
Beimel, Amos 144
Bellare, Mihir 294, 312
Ben-Sasson, Eli 663
Berta, Mario 776
Bitansky, Nir 255
Bos, Joppe W. 626
Brakerski, Zvika 868
Brickell, Ernie 570
Burra, Sai Sheshank 681

Chase, Melissa 236
Chiesa, Alessandro 255

Dachman-Soled, Dana 533
Damgård, Ivan 643
Dinur, Itai 719
Dodis, Yevgeniy 348, 497
Duc, Alexandre 832
Dunkelman, Orr 719
Dupuis, Frédéric 794

Farràs, Oriol 144
Fawzi, Omar 776
Fehr, Serge 663
Focardi, Riccardo 608

Garg, Sanjam 105, 424
Gehrke, Johannes 479
Gentry, Craig 850
Gorbunov, Sergey 162
Goyal, Vipul 443

Halevi, Shai 850
Hanaoka, Goichiro 812
Hay, Michael 479
Hoang, Viet Tung 1

Hofheinz, Dennis 590
Hughes, James P. 626

Iwata, Tetsu 31

Jager, Tibor 273, 590
Jain, Abhishek 443
Jetchev, Dimitar 832

Kalai, Yael Tauman 533
Katz, Jonathan 124
Kawamoto, Yusuke 608
Keller, Nathan 719
Khovratovich, Dmitry 367
Kleinjung, Thorsten 626
Knellwolf, Simon 367
Kohlar, Florian 273
Kumarasubramanian, Abishek 424

Landecker, Will 14
Leander, Gregor 50
Lenstra, Arjen K. 626
Lewko, Allison 180
Libert, Benoît 571
Lin, Huijia 461
Liu, Feng-Hao 517
López-Alt, Adriana 497
Lui, Edward 479
Lysyanskaya, Anna 517

Mahmoody, Mohammad 701
Mashatan, Atefeh 741
Matsuda, Takahiro 812
Maurer, Ueli 124
Mennink, Bart 330
Miles, Eric 68
Minematsu, Kazuhiko 31
Mintz, Yuval 144
Mironov, Ilya 497
Morris, Ben 1

Nielsen, Jesper Buus 681, 794
Nordholt, Peter Sebastian 681

Ohashi, Keisuke 31
Orlandi, Claudio 681
Ostrovsky, Rafail 424, 663

- Pass, Rafael 461, 479, 701
Pastro, Valerio 643
Peters, Thomas 571
Prabhakaran, Manoj 443
Preneel, Bart 330
- Ristenpart, Thomas 312, 348
Rogaway, Phillip 1
Rosulek, Mike 87, 406
Rothblum, Guy N. 552
- Sahai, Amit 105, 199, 443
Salvail, Louis 794
Schäge, Sven 273
Schuldt, Jacob C.N. 812
Schwenk, Jörg 273
Seyalioglu, Hakan 199
Shamir, Adi 719
Shrimpton, Thomas 14
Simionato, Lorenzo 608
Smart, Nigel P. 643, 850
Steel, Graham 608
Steinberger, John 348, 384
Sun, Xiaoming 384
- Terashima, R. Seth 14
Tessaro, Stefano 294, 312, 348
Tsay, Joe-Kai 608
- Vadhan, Salil 497
Vaikuntanathan, Vinod 162
Vardy, Alexander 294
Vaudenay, Serge 741
Viola, Emanuele 68
Visconti, Ivan 236, 424
- Wachter, Christophe 626
Waters, Brent 180, 199, 218
Wee, Hoeteck 162
Wehner, Stephanie 776
- Yang, Zhe 384
Yung, Moti 571
- Zakarias, Sarah 643
Zhandry, Mark 758
Zikas, Vassilis 124
Zittrain, Jonathan 86