LNCS 6558

Aggelos Kiayias (Ed.)

Topics in Cryptology – CT-RSA 2011

The Cryptographers' Track at the RSA Conference 2011
San Francisco, CA, USA, February 2011
Proceedings

RSACONFERENCE2011

Springer

Aggelos Kiayias (Ed.)

Topics in Cryptology – CT-RSA 2011

The Cryptographers' Track at the RSA Conference 2011
San Francisco, CA, USA, February 14-18, 2011
Proceedings

Springer

Volume Editor

Aggelos Kiayias
National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
Panepistimiopolis, Ilisia, Athens 15784, Greece
E-mail: aggelos@kiayias.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-19073-5 e-ISBN 978-3-642-19074-2
DOI 10.1007/978-3-642-19074-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2010943110

CR Subject Classification (1998): E.3, D.4.6, K.6.5, C.2, K.4.4

LNCS Sublibrary: SL 4 – Security and Cryptology

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

The RSA conference was initiated in 1991 and is a major international event for cryptography and information security researchers as well as the industry related to these disciplines. It is an annual event that attracts hundreds of vendors and thousands of participants from industry and academia. Since 2001, the RSA conference has included the Cryptographers' Track (called the CT-RSA), which enables the forefront of cryptographic research to be presented within the formal program of the conference. CT-RSA has become a major publication venue for cryptographers worldwide.

This year the RSA conference was held in San Francisco, California, during February 14–18, 2011. The CT-RSA conference servers were running in the University of Athens, Greece, and we received 82 submissions out of which 3 were withdrawn. Every paper was reviewed by at least three committee members. The Program Committee members were also allowed to submit up to one paper for inclusion in the program. Such papers were reviewed by at least five committee members. The reviewing of the submissions proceeded in two stages: in the first stage papers were read individually by committee members without knowledge of other committee members' opinions. In the second stage, all reviews were made available to committee members and discussion through a Web bulletin board ensued. After a total of seven weeks the committee work concluded and a selection of 24 papers was made for inclusion in the conference program. In a small number of cases a final round of reviewing took place as some of the papers were accepted conditionally on specific changes that were requested by the Program Committee. The final revised versions of the accepted papers is what you will find in this volume.

We were very pleased this year to have three keynote talks included in the CT-RSA program. Dan Boneh from Stanford University gave a talk on computing with signed data. Dickie George of the Information Assurance Directorate at NSA spoke on NSA's role in the development of DES. Adi Shamir from the Weizmann Institute of Science gave a talk on the role of academia and industry in the design and analysis of DES. The talk also featured a mini-talk by Martin Hellman on that subject.

A number of people played key roles in the success of the conference this year. First and foremost I would like to thank the authors of all submitted papers; without their contributions the conference would not have been possible. Second, a special thanks is due to the members of the Program Committee and the subreviewers that invested a lot of their time in carefully reading the submitted papers and contributing to the discussion of each paper. The submission and review process was supported by the Web submission software written by Shai Halevi. I would also like to thank Bree LaBollita and Amy Szymanski, who worked very hard to properly organize the conference this year.

December 2010 Aggelos Kiayias

CT-RSA 2011
The 11th Cryptographers' Track – RSA 2011

San Francisco, California, USA
February 14–18, 2011

Program Chair

Aggelos Kiayias	University of Athens, Greece

Steering Committee

Masayuki Abe	NTT, Japan
Ari Juels	RSA Laboratories, USA
Tal Malkin	Columbia University, USA
Josef Pieprzyk	Macquarie University, Australia
Ron Rivest	MIT, USA
Moti Yung	Google, USA

Program Committee

Giuseppe Ateniese	Johns Hopkins University, USA
Sasha Boldyreva	Georgia Tech, USA
Xavier Boyen	Université de Liège, Belgium
Christophe De Cannière	KU Leuven, Belgium
Jung Hee Cheon	Seoul National University, Korea
Joo Yeon Cho	Nokia, Denmark
Orr Dunkelman	Weizmann Institute, Israel
Steven Galbraith	University of Auckland, New Zealand
Craig Gentry	IBM Research, USA
Philippe Golle	Google, USA
Louis Goubin	Université de Versailles, France
Iftach Haitner	Tel Aviv University, Israel
Amir Herzberg	Bar Ilan University, Israel
Dennis Hofheinz	Karlsruhe University, Germany
Stanislaw Jarecki	UC Irvine, USA
Marc Joye	Technicolor, France
Ralf Küsters	University of Trier, Germany
Anna Lysyanskaya	Brown University, USA

Alexander May	Ruhr University Bochum, Germany
Daniele Micciancio	UCSD, USA
Tal Moran	Harvard University, USA
Antonio Nicolosi	Stevens Institute of Technology, USA
Tatsuaki Okamoto	NTT, Japan
Rafail Ostrovsky	UCLA, USA
Josef Pieprzyk	Macquarie University, Australia
David Pointcheval	ENS, France
Berry Schoenmakers	TU Eindhoven, The Netherlands
Alice Silverberg	UC Irvine, USA
Martijn Stam	Royal Holloway, University of London, UK
François-Xavier Standaert	UCL, Belgium
Berk Sunar	WPI, USA
Nikos Triandopoulos	RSA Laboratories and Boston University, USA
Huaxiong Wang	NTU, Singapore
Bogdan Warinschi	University of Bristol, UK

External Reviewers

Onur Aciicmez
Kahraman Akdemir
Martin Albrecht
Gilad Asharov
Roberto Avanzi
Foteini Baldimtsi
Lejla Batina
Carl Bosley
Ran Canetti
Sherman S.M. Chow
Léonard Dallot
Yevgeniy Dodis
Gwenaël Doërr
Nelly Fazio
Nicolas Gama
Hossein Ghodosi
Yossi Gilad
Choudary Gorantla
Jian Guo
Ghaith Hammouri
Malin Md Mokammel Haque
Mathias Herrmann
Susan Hohenberger
Yin Hu
Xinyi Huang
Malika Izabachene
Dimitar Jetchev
Deniz Karakoyunlu
HongTae Kim
Jihye Kim
Jinsoo Kim
Minkyu Kim
Taechan Kim
Mikkel Krøigård
Ranjit Kumaresan
Hyung Tae Lee
Anja Lehmann
Helger Lipmaa
David M'Raihi
Ilya Mironov
Daisuke Moriyama
Erdinc Ozturk
Charalampos Papamanthou
Olivier Pereira
Ludovic Perret
Christophe Petit
Nicolas Prigent
Tal Rabin
Francesco Regazzoni
Andy Rupp
Jae Hong Seo
Elaine Shi

Haya Shulman	Max Tuengerthal
Radu Sion	Nicolas Veyrat-Charvillon
Ron Steinfeld	Andreas Vogt
Hung-Min Sun	Liang Feng Zhang
Stefano Tessaro	Hong-Sheng Zhou
Tomasz Truderung	Vassilis Zikas

Table of Contents

Secure Set Intersection with Untrusted Hardware Tokens

Marc Fischlin[1], Benny Pinkas[2], Ahmad-Reza Sadeghi[1,3], Thomas Schneider[3], and Ivan Visconti[4]

[1] Darmstadt University of Technology, Germany
marc.fischlin@gmail.com
[2] Bar Ilan University, Ramat Gan, Israel
benny@pinkas.net
[3] Ruhr-University Bochum, Germany
{ahmad.sadeghi,thomas.schneider}@trust.rub.de
[4] University of Salerno, Italy
visconti@dia.unisa.it

Abstract. Secure set intersection protocols are the core building block for a manifold of privacy-preserving applications.

In a recent work, Hazay and Lindell (ACM CCS 2008) introduced the idea of using trusted hardware tokens for the set intersection problem, devising protocols which improve over previous (in the standard model of two-party computation) protocols in terms of efficiency and secure composition. Their protocol uses only a linear number of symmetric-key computations and the amount of data stored in the token does not depend on the sizes of the sets. The security proof of the protocol is in the universal composability model and is based on the strong assumption that the token is trusted by *both* parties.

In this paper we revisit the idea and model of hardware-based secure set intersection, and in particular consider a setting where tokens are not necessarily trusted by both participants to additionally cover threats like side channel attacks, firmware trapdoors and malicious hardware. Our protocols are very efficient and achieve the same level of security as those by Hazay and Lindell for trusted tokens. For untrusted tokens, our protocols ensure privacy against malicious adversaries, and correctness facing covert adversaries.

Keywords: cryptographic protocols, set intersection, untrusted hardware.

1 Introduction

A variety of applications with sophisticated privacy requirements can be based on secure set operations, in particular secure set intersection. Examples are versatile and range from government agencies comparing their databases of suspects on a national and international basis, to competing enterprises evaluating their performance on various aspects (items, deployed processes), to dating services.

The underlying protocols typically involve two mistrusting parties who compute an intersection of their respective sets (or some function of them). As we elaborate in §1.1 on related work, cryptographic research has proposed several solutions to this problem, each having its own strengths and weaknesses; in particular, the efficiency aspect is crucial for deployment in real-life scenarios: While software-based solutions use expensive public-key operations, it is also possible to incorporate a tamper-proof hardware token into the protocol, yielding more efficient schemes and/or avoiding impossibility results. However, this hardware-based model requires a strong trust model, i.e., a token trusted by all parties.

Background. In this paper we will focus on a recent proposal by Hazay and Lindell [1] that aims to design *truly practical* and secure set intersection protocols by introducing a new party, a (tamper-proof) hardware token $\mathcal{T}$. Here, one party, called the issuer $\mathcal{A}$, programs a key into the token $\mathcal{T}$ which protects this key from being accessible by the other party $\mathcal{B}$. At the same time, the manufacturer of the token ensures that the token correctly computes the intended function, i.e., $\mathcal{A}$ can only choose the secret key but cannot interfere with the token's program. The protocol is very efficient and requires the involved parties and the token to perform a few pseudorandom permutation evaluations, thus disposing of any public-key operations and/or random oracles as in previous efforts (cf. §1.1).

The use of the token in [1] is justified when trusted hardware manufacturers are available (e.g., manufacturers which produce high-end smartcards that have FIPS 140-2, level 3 or 4 certification). The security of the scheme is proven in the Universal Composability (UC) model [2], guaranteeing security even when composed with other protocols. It is important to note that today's high-end smartcards may have a sufficient amount of resources for executing the entire ideal functionality in a relatively simple use-case such as set intersection, although probably not on relatively large inputs. However, doing so would require to program the smartcard to implement this specific functionality. The protocols of [1] as well as the protocols we propose, on the other hand, can be run in practice by using cheap smartcards: they assume limited computation capabilities (only symmetric-key operations) and *constant* storage (see also [1]).

Motivation. The security proof of the scheme of [1] considers the universal composability framework inherently relying on the trustworthiness of the token, since it is assumed that both parties fully trust the token. This assumption, though, is critical with regard to several aspects regarding to what level tokens can be trusted in practice.

First, even extensive testing of the token cannot provide protection against errors and backdoors, introduced accidentally or deliberately in the underlying hardware and software stack running on it. A well-known example is the "Pentium bug" which caused the floating point division unit of the Intel Pentium$^{\mathrm{TM}}$ processor to compute slightly wrong results for specific inputs [3]. Such flaws in the hardware can be exploited in so called "bug attacks" [4] to break the security of the underlying protocol. Moreover, although appropriate certification might help to ensure, to some degree, that at least the design of the token is backdoor-free, it is still unclear how to protect against hardware Trojans being maliciously introduced into the hardware during the manufacturing process, particularly because chip production is increasingly outsourced to other countries which are potentially untrusted or have their own evaluation standards.

Another threat concerns hardware and side-channel attacks allowing to break hardware protection mechanisms. Modern commercial smartcards have been equipped with a variety of measures to counter standard side-channel attacks. However, the severeness of attacks depends of course on the effort (see, e.g., the recently reported hardware attack on the Trusted Platform Module (TPM) [5]).

Our Contribution and Outline. After summarizing related works on set intersection and token-based protocols in §1.1, we introduce our setting and the employed primitives in §2, and review the basic protocol of [1] in §3. Afterwards, we present the following contributions.

We revisit the model of a fully trusted hardware token and provide several protocols for secure set intersection that make use of untrusted hardware tokens and fulfill different security targets. In our protocols only one party $\mathcal{A}$ trusts (some of) the hardware token(s) but the other party $\mathcal{B}$ does not. More concretely, we present a stepwise design of token-based set intersection protocols:

1. Guaranteeing the privacy of $\mathcal{B}$'s inputs in the *malicious* adversary model, using a *single* token trusted only by the issuer $\mathcal{A}$ (§4).
2. Additionally guaranteeing the correctness of $\mathcal{B}$'s outputs in the *covert* adversary model, using a *single* token trusted only by the issuer (§5).
3. Additionally preserving the privacy of $\mathcal{A}$'s inputs in the *malicious* adversary model, using *multiple* tokens of which at least one is trusted by issuer $\mathcal{A}$ (§6).

Moreover, our protocols have the "fall-back" security guarantees to the protocol of [1]: in case both parties fully trust the token, our protocols still provide the same security properties as [1]. While the original protocol of [1] does not provide any security guarantees in the case of untrusted token, our protocols achieve input privacy for malicious adversaries and output correctness for a covert token, i.e., any cheating attempt of the token may breach correctness (but not privacy) and is detectable with high probability.

1.1 Related Work

Set Intersection without Hardware Tokens. Several protocols for two-party set intersection secure in the *semi-honest* model have been proposed [6, 7, 8, 9, 10]. Protocols with security against *malicious* adversaries are given in [6,7,11,12,8,13,14,15,16,17]. A detailed summary and performance comparison of most of these protocols is given in [9]. Protocols with *covert* security are given in [12,16]. All these protocols that do not employ hardware tokens need a non-negligible number of computationally expensive public-key operations [6]. In contrast, the protocols of [1] and our protocols perform a linear number of fast symmetric-key operations only.

Set Intersection with Hardware Tokens Trusted by Both Parties. HW tokens with limited capabilities that are trusted by both parties have been used to construct more efficient protocols for verifiable encryption and fair exchange [18], and secure function evaluation [19,20]. Additionally, government-issued signature cards have been proposed as setup assumption for UC [21]. Further, semi-honest tamper-proof hardware tokens can serve as basis for non-interactive oblivious transfer and hence non-interactive secure two-party computation, called one-time programs [22,23]. Our tokens need not to be trusted by both parties. In the rest of the paper we will extend the token-based set intersection model and protocol proposed recently in [1] which we summarize in §3.

Set Intersection with Hardware Tokens Trusted by the Issuer Only. HW tokens trusted by their issuer only were used as setup assumption for constructing UC commitments [24,25,26,27], and information-theoretic one-time programs [28]. These protocols use HW tokens merely to overcome known impossibility results, but do not claim to yield efficient protocols for practical applications.

To improve the performance of practical two-party secure function evaluation protocols, garbled circuits can be generated efficiently using a HW token trusted by its issuer only [29]. Furthermore, truly efficient oblivious transfer protocols with security against covert adversaries were proposed in [30]. We adapt techniques of [30] for constructing our protocols for secure set intersection.

2 Preliminaries

We denote the security parameter for symmetric schemes by t. A *pseudorandom permutation* (PRP) F is an algorithm which takes as input a key $k \in \{0,1\}^t$ and describes a "random-looking" permutation $F_k(\cdot)$ over $D = \{0,1\}^t$. If we drop the requirement on F being a permutation, then we have a *pseudorandom function* (PRF) instead. If it also holds that it is hard to distinguish permutation F_k from a random permutation given access to both the permutation and its inverse, then F is called a *strong* pseudorandom permutation (SPRP). Note that AES, for example, is believed to be a strong PRP.

2.1 The Setting for Token-Based Set Intersection Protocols

The general setting for the set intersection protocols we consider is as follows: Two parties, $\mathcal{A}$ and $\mathcal{B}$ would like to compute the intersection $\mathcal{F}_\cap(X,Y) = X \cap Y$ on their input sets $X = \{x_1, \ldots, x_{n_A}\}$ and $Y = \{y_1, \ldots, y_{n_B}\}$ such that only $\mathcal{B}$ obtains the output (while $\mathcal{A}$ learns nothing). Note that we assume that the set sizes are known to both parties. We further assume that elements from X and Y are from a domain $D = \{0,1\}^t$, i.e., $X, Y \subseteq D$. If needed, larger input data can be hashed to shorter strings with a collision-resistant hash function.

Our protocols have the following general structure: party $\mathcal{A}$ issues, i.e., buys, one or more hardware tokens $\mathcal{T}_1, \ldots, \mathcal{T}_n$, where $\mathcal{T}_i$ is manufactured by the hardware manufacturer $\mathcal{M}_i$. It initializes the tokens $\mathcal{T}_i$, and sends them to $\mathcal{B}$. In the case of protocols with a single token we simply call the token $\mathcal{T}$ and its manufacturer $\mathcal{M}$. In our model, any of the participating parties may be dishonest (where a dishonest token $\mathcal{T}$ refers to a maliciously produced token), and all malicious parties are controlled by a single adversary. We say that *a party trusts* $\mathcal{T}$ iff the other party cannot collude with $\mathcal{M}$ to produce a dishonest or breakable token. We consider static corruptions only.

To model hardware-based access we assume that, once a token is in possession of $\mathcal{B}$, $\mathcal{A}$ cannot communicate with the token anymore. In particular, the adversary may construct a malicious token, but may not interact with the token anymore, once it is sent to $\mathcal{B}$. The adversary can only communicate with the token through messages sent to and received from $\mathcal{B}$. Analogously, two tokens cannot communicate directly.

2.2 Security Models

While we denote by $\mathcal{A}, \mathcal{B}$, and $\mathcal{T}$ respectively the first (left) player, the second (right) player and the token, we will denote by $\mathcal{A}_I$ and $\mathcal{B}_I$ the players of the ideal world where parties just send their inputs to a set intersection functionality that then sends the intersection of the received inputs to $\mathcal{B}_I$.

We use different security notions. First, we consider *unconditional privacy* of the input of a player, i.e., regardless of the actions of the other malicious player, the input of an honest player will remain private in the sense that anything that can be computed about it can also be computed in the ideal world.

When we can carry a real-world attack mounted by an adversary during a protocol run into an ideal world attack, we achieve *simulation-based security.* If simulation cannot be achieved, we will instead downgrade to the weaker *indistinguishability-based security* notion. This last notion means that a malicious player cannot guess which input the other player has used during a protocol run, even when the honest player uses one of two inputs determined by the adversary.

The traditional notion of security through realizing an ideal functionality requires the simulation of any real-world attack into an ideal-world attack, and that the outputs of honest players do not deviate in the two worlds. We then say that the protocol *securely computes* (or *evaluates) the functionality* $\mathcal{F}_\cap(X, Y)$, and often specify the adversary's capabilities further, e.g., that the token is trusted or that it cannot be compromised by $\mathcal{B}$. This classical notion implicitly includes a *correctness* requirement: the output of a honest player depends only on its input and the implicit input used by the adversary in the protocol run.

When our protocols cannot achieve the *correctness* and *simulation* requirements simultaneously, we will downgrade the standard security notion to *covert security* [31], which means that the adversarial behavior can be detected by the honest player with some non-negligible probability ϵ, called the deterrence factor.[1] In all applications where the reputation of a player is more important than

[1] In addition the protocol must be *detection accurate* in the sense that in real-world executions no honest party accuses another honest party of cheating. All our protocols obey this property, albeit we do not mention this explicitly.

the output correctness of another player (e.g., where established enterprises offering services to citizens), this notion of covert security suffices, since there is a deterrence factor that discourages malicious actions.

We note that our protocols provide stronger security guarantees than security against the strongest notion of covert adversaries defined in [31], as no information about honest players' inputs is leaked, independently of whether cheating was detected or not. That is, in our case the ideal-world adversary can issue a `cheat` command (in case he wants to cheat) and this is announced to the parties with probability ϵ – but unlike in [31] the ideal-world adversary here does not get to learn the honest parties' inputs in case no cheat is announced. Still, in such a case we provide no correctness guarantee whatsoever.

3 Both Parties Trust Token [1]

We now review the model and protocol of [1]. Our models and protocols presented later extend on these to cope with untrusted hardware.

Model of [1]. In the model of [1], the hardware token $\mathcal{T}$ is assumed to honestly compute the intended functionality. The authors of [1] argue that this assumption is justified if highly trusted hardware manufacturers are available, e.g., manufacturers which produce high-end smartcards that have FIPS 140-2, level 3 or 4 certification. The token $\mathcal{T}$ is as reliable as its manufacturer $\mathcal{M}$ and, as only $\mathcal{T}$ is involved in the protocol but not $\mathcal{M}$, this security assumption is weaker than using $\mathcal{M}$ as a trusted third party.[2]

Set Intersection Protocol of [1]. The set intersection protocol of [1], depicted in Fig. 1, works as follows: In the setup phase, $\mathcal{A}$ initializes the HW token $\mathcal{T}$ with a random key k, a random message OK, and an upper bound on the size of $\mathcal{B}$'s input set n_B; $\mathcal{A}$ sends $\mathcal{T}$ to $\mathcal{B}$. In the online phase, $\mathcal{B}$ can query the token to evaluate F_k (where F is a SPRP as defined in §2) on each of its inputs. If $\mathcal{T}$ has been queried n_B times, it invalidates k (e.g., by deleting it)[3] and outputs OK to $\mathcal{B}$ who forwards it to $\mathcal{A}$. If OK is correct, $\mathcal{A}$ sends the evaluation of F_k on each of his inputs to $\mathcal{B}$. Finally, $\mathcal{B}$ computes the intersection by comparing the values obtained from $\mathcal{T}$ with those from $\mathcal{A}$. (Note that at that point $\mathcal{B}$ cannot query $\mathcal{T}$ anymore, i.e., all queries to $\mathcal{T}$ were independent of $\mathcal{A}$'s inputs.)

Security. According to Theorem 3 of [1], the above protocol UC-securely realizes the set intersection functionality when $\mathcal{T}$ is honest.

[2] This model is somewhat related to the common reference string (CRS) model in which a party trusted by all players generates a string according to a given distribution. The string is later used in the protocol. While a CRS is a static information generated before protocol executions, the trusted token will offer a trusted functionality *during* the execution of a protocol.

[3] This ensures that $\mathcal{B}$ gains no advantage when querying $\mathcal{T}$ in an invalid way.

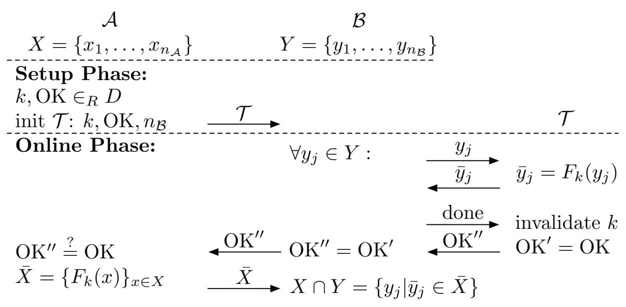

$\mathcal{A}$
$X = \{x_1, \ldots, x_{n_\mathcal{A}}\}$
$\mathcal{B}$
$Y = \{y_1, \ldots, y_{n_\mathcal{B}}\}$
Setup Phase:
$k, \mathrm{OK} \in_R D$
init $\mathcal{T}$: $k, \mathrm{OK}, n_\mathcal{B}$
$\mathcal{T}$
$\mathcal{T}$
Online Phase:
$\forall y_j \in Y:$
y_j
$\bar{y}_j$
$\bar{y}_j = F_k(y_j)$
done
invalidate k
OK''
$\mathrm{OK}' = \mathrm{OK}$
$\mathrm{OK}'' \overset{?}{=} \mathrm{OK}$
OK''
$\mathrm{OK}'' = \mathrm{OK}'$
$\bar{X} = \{F_k(x)\}_{x \in X}$
$\bar{X}$
$X \cap Y = \{y_j | \bar{y}_j \in \bar{X}\}$

Fig. 1. Set Intersection Protocol of [1]: token $\mathcal{T}$ is trusted by both parties

Efficiency. $\mathcal{T}$ performs n_B evaluations of F. The communication in the online phase contains the OK message from $\mathcal{B}$ to $\mathcal{A}$, and a message containing $n_A t$ bits from $\mathcal{A}$ to $\mathcal{B}$. The overall online communication complexity is therefore $\mathcal{O}(n_A t)$.

4 Only Issuer Trusts Token: Privacy of $\mathcal{B}$'s Input

The protocol of [1] assumes that $\mathcal{T}$ is fully trusted by both parties. Obviously, when one of the parties can break into $\mathcal{T}$ (e.g., by physical attacks or by colluding with its manufacturer $\mathcal{M}$), they can break the correctness or the privacy of the protocol. In the following we extend the protocol of [1] to make it non-interactive and guarantee privacy of $\mathcal{B}$'s inputs even if $\mathcal{A}$ and $\mathcal{T}$ are malicious.

Model. We consider the trust model where $\mathcal{B}$ does not trust $\mathcal{T}$ to behave correctly, i.e., $\mathcal{A}$ can collude with the hardware manufacturer $\mathcal{M}$ to produce a bad token $\mathcal{T}$. This model seems justified, as $\mathcal{B}$ is required to use a hardware token which is provided by $\mathcal{A}$, whom $\mathcal{B}$ might not trust.

Problem 1 ($\mathcal{A}$ colludes with $\mathcal{M}$ to break privacy of $\mathcal{B}$'s inputs). In the protocol of Fig. 1, the only message in which information about $\mathcal{B}$'s inputs can be leaked to $\mathcal{A}$ is the OK message. A corrupt player $\mathcal{A}$ can construct a corrupt token $\mathcal{T}$ that changes the OK message based on the inputs that $\mathcal{B}$ feeds to $\mathcal{T}$ (i.e., OK is used as covert channel), or $\mathcal{T}$ aborts the protocol (e.g., refuses to output OK).

Protocol. Problem 1 arises in the protocol of [1], as $\mathcal{B}$ first provides his input Y to $\mathcal{T}$, $\mathcal{T}$ answers $\mathcal{B}$ and finally $\mathcal{B}$ sends a message to $\mathcal{A}$ which depends on $\mathcal{T}$'s answer (OK). We eliminate this source of leakage from $\mathcal{T}$ to $\mathcal{A}$ in the protocol as shown in Fig. 2, by making the protocol non-interactive: First, $\mathcal{A}$ sends the permutations $\bar{X}$ of its inputs (as before). Afterwards, $\mathcal{B}$ obtains its permuted inputs $\bar{Y}$ from $\mathcal{T}$ by sending its inputs Y to $\mathcal{T}$. In contrast to the original protocol, $\mathcal{T}$ cannot reveal the permuted inputs $\bar{y}_j$ directly to $\mathcal{B}$ as otherwise $\mathcal{B}$, who already knows $\bar{X}$ now, could already compute parts of the intersection $X \cap \{y_1, \ldots, y_j\}$ and adaptively change his input depending on this. Instead, $\mathcal{T}$ encrypts each $\bar{y}_j$ by XORing it with a pseudo-random pad p_j which is derived by computing a pseudo-random function $f_s(j)$ keyed with a fixed secret key s. After having queried for all elements in Y, $\mathcal{B}$ has an encrypted copy of $\bar{Y}$. Now, $\mathcal{T}$ releases the pseudo-random pads p_j with which $\bar{Y}$ is encrypted to $\mathcal{B}$, who can finally recover $\bar{Y}$ and compute $X \cap Y$ as before.

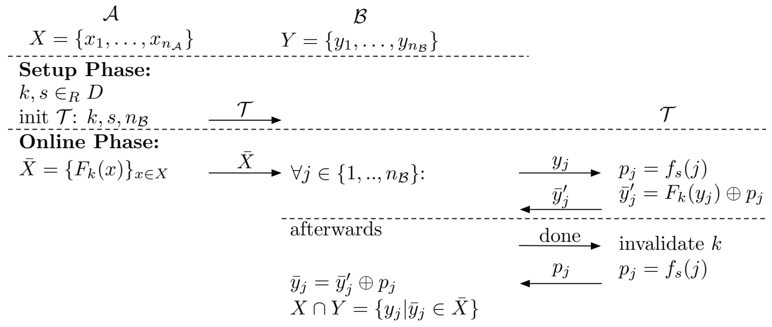

$\mathcal{A}$
$X = \{x_1, \ldots, x_{n_\mathcal{A}}\}$
$\mathcal{B}$
$Y = \{y_1, \ldots, y_{n_\mathcal{B}}\}$
Setup Phase:
$k, s \in_R D$
init $\mathcal{T}$: $k, s, n_\mathcal{B}$
$\mathcal{T}$
$\mathcal{T}$
Online Phase:
$\bar{X} = \{F_k(x)\}_{x \in X}$
$\bar{X}$
$\forall j \in \{1, .., n_\mathcal{B}\}$:
y_j
$\bar{y}'_j$
$p_j = f_s(j)$
$\bar{y}'_j = F_k(y_j) \oplus p_j$
afterwards
done
invalidate k
p_j
$p_j = f_s(j)$
$\bar{y}_j = \bar{y}'_j \oplus p_j$
$X \cap Y = \{y_j | \bar{y}_j \in \bar{X}\}$

Fig. 2. Set Intersection Protocol with Privacy of $\mathcal{B}$'s Inputs (Problem 1) w.r.t. malicious adversaries: token $\mathcal{T}$ is not trusted by $\mathcal{B}$

Theorem 1. *If F is a SPRP and f is a PRF, then the protocol depicted in Fig. 2:*

1. *securely evaluates $\mathcal{F}_\cap(X, Y)$ w.r.t. a malicious $\mathcal{B}$ that cannot break into $\mathcal{T}$;*
2. *keeps $\mathcal{B}$'s input unconditionally private in the indistinguishability sense w.r.t. a malicious $\mathcal{A}$;*
3. *securely evaluates $\mathcal{F}_\cap(X, Y)$ when both parties trust the token.*

Proof. To prove Theorem 1 we treat each corruption case separately.

$\mathcal{A}$ is corrupted and $\mathcal{T}$ is trusted by $\mathcal{A}$ and $\mathcal{B}$. As noted above, non-interactivity implies that $\mathcal{B}$'s input is protected unconditionally from a malicious $\mathcal{A}$. Here however, we can even prove unconditional security in a simulation-based sense, constructing an ideal-world adversary $\mathcal{A}_I$ that simulates in the ideal world the attack carried out by $\mathcal{A}$ in the real world. The difference here that allows us to achieve such a stronger security notion is that since the token is trusted, it has not been produced by $\mathcal{A}$, and therefore $\mathcal{A}$ has only black-box access to it. Thus, given a real-world adversary $\mathcal{A}$, we can construct an ideal-world adversary $\mathcal{A}_I$ that includes $\mathcal{A}$ and is able to read and write on its communication channels, including the ones that are supposed to be used for the communication with the token. Notice that since the token is trusted, from the fact that it answers to $\mathcal{B}$'s queries, it must be the case that $\mathcal{A}$ uploads to $\mathcal{T}$ both k and s – otherwise $\mathcal{T}$ would be in an inconsistent state and would not play with $\mathcal{B}$ (that therefore would just abort). Thus, $\mathcal{A}_I$ will obtain k and s from the initialization of the token performed by $\mathcal{A}$. Then, $\mathcal{A}_I$ reads the vector of messages $\bar{X}$ and inverts each $\bar{x}_j \in \bar{X}$ obtaining the original vector X that corresponds to the set that $\mathcal{A}$ would play in the real world. Then, $\mathcal{A}_I$ plays X in the ideal world. As a consequence,

the ideal-world honest player $\mathcal{B}_I$ will obtain the same input obtained by a real-world honest player $\mathcal{B}$, that plays the protocol with a trusted token. Finally $\mathcal{A}_I$ outputs whatever $\mathcal{A}$ outputs. As the joint distribution of the view of $\mathcal{A}$ and the output of $\mathcal{B}$ in real and ideal world are clearly identical, property 1 holds.

$\mathcal{A}$ is corrupted and $\mathcal{T}$ is trusted by $\mathcal{A}$ but not $\mathcal{B}$. Since the protocol is non-interactive, $\mathcal{A}$ does not get any message from $\mathcal{B}$ and therefore $\mathcal{B}$'s privacy is protected unconditionally. However, we cannot construct and ideal-world adversary $\mathcal{A}_I$ since we cannot extract $\mathcal{A}$'s input. Therefore we obtain unconditional indistinguishability of B's private input, and property 2 holds.

$\mathcal{B}$ is corrupted. To prove that $\mathcal{A}$'s input remains private in a simulation-based sense against a real-world malicious $\mathcal{B}$ we construct an ideal-world adversary $\mathcal{B}_I$ that internally simulates a protocol run to $\mathcal{B}$, extracts its input and plays the extracted input in the ideal world. $\mathcal{B}_I$ has control over the communication channels used by $\mathcal{B}$ to communicate with $\mathcal{T}$, and thus reads all queries y_j performed by $\mathcal{B}$, sending as answer random values $\bar{y}'_j$. Moreover, $\mathcal{B}_I$ sends to $\mathcal{B}$ a random vector $\bar{X}$ therefore simulating the message of the honest real-world $\mathcal{A}$. As soon as all elements of $\mathcal{B}$ have been sent to the (simulated) token, $\mathcal{B}_I$ groups all the elements in a set Y that is sent to the ideal functionality. $\mathcal{B}_I$ then obtains from the ideal functionality the intersection of Y with $\mathcal{A}_I$'s input, where $\mathcal{A}_I$ is the honest player of the ideal model. Let Z be the output of $\mathcal{B}_I$ in the ideal world. $\mathcal{B}_I$ now aims at giving Z to $\mathcal{B}$ in the internal execution of the real-world protocol. To do so, it performs the last n_B steps of the protocol sending values $p_1, \ldots, p_{n_B}$ as follows: if y_j is in Z then set $p_j = y'_j \oplus \bar{y}_j$, else set p_j equal to a random string. Then $\mathcal{B}_I$ outputs whatever $\mathcal{B}$ outputs.

Notice that the only difference in the view of $\mathcal{B}$ between the real-world and the simulated executions is that the former uses the SPRP F and the PRF f, while the latter uses random bits. We now show that any distinguisher between the two views, can be used to build either an adversary for F or an adversary f.

Consider the hybrid experiment G in which the real-world execution is played but F is replaced by random strings, still keeping consistency so that on the same input F produces the same output. Clearly G can be run in polynomial time and is computationally indistinguishable from the real-world execution, otherwise we have immediately a forgery for the SPRP F.

Consider now the next hybrid game G' in which all evaluations of f are replaced by random bits, still keeping consistency as above. Again, any distinguisher between G and G' would immediately produce a forgery for the PRF f.

Finally, consider the simulated execution of the real-world protocol. Both the message sent over the communication channel (i.e., $\bar{X}$) and the first bunch of answers of $\mathcal{T}$ (i.e., $\bar{y}'_j$) have the uniform distribution and are therefore identically distributed in both G' and in the simulated game. The final answers p_j received by $\mathcal{B}$ correspond in both the simulated game and in G' to random messages, with the only exception of the elements that appear in the intersection. In this last case the received messages p_j correspond precisely to the unique values that allow $\mathcal{B}$ to compute the values in the intersection. This holds both in G' and in the simulated execution. This allows us to conclude the proof of property 3. □

Efficiency and Token Reusability. While the round complexity of our protocol is optimal, compared to the 3 rounds of [1], its computational complexity is only by a factor of approximately 3 worse. Overall, the computational and storage requirements for $\mathcal{T}$ are the same in both protocols, namely symmetric-key operations (SPRP and PRF), and a small constant amount of secure storage.

Our protocols can be extended to reuse the same token for multiple protocol runs. For this, all information shared between $\mathcal{A}$ and $\mathcal{T}$ (i.e., the value k and s) is derived pseudo-randomly from a master-key known by $\mathcal{A}$ and $\mathcal{T}$ and some session identifier. The token $\mathcal{T}$ keeps track of the next session id using a strictly monotonic tamper-proof hardware counter which is available in most smartcards today. Also updating the usage counter n_B inside the token is possible via secure messaging as described in [1].

5 Only Issuer Trusts Token: Correctness of $\mathcal{B}$'s Output

In this section we extend the protocol of §4 to guarantee privacy and correctness when $\mathcal{B}$ does not trust the token. This is formalized by the following problem.

Problem 2 ($\mathcal{A}$ colludes with $\mathcal{M}$ to break correctness of $\mathcal{B}$'s output). In the protocols of Fig. 1 and Fig. 2, a corrupt $\mathcal{A}$ can enforce $\mathcal{B}$ to obtain in the protocol wrong outputs, i.e., different from $X \cap Y$: This can be done by creating a malicious token $\mathcal{T}$ that does not compute the permutation F correctly, but computes another function F' which maps multiple values to the same value or even depends on the history of values seen from $\mathcal{B}$.

Although Problem 2 does not affect the privacy of $\mathcal{B}$'s input, the correctness of $\mathcal{B}$'s output is no longer guaranteed. In many application scenarios this is not a problem, as a malicious $\mathcal{A}$ could also provide wrong inputs to the computation. However, a malicious token $\mathcal{T}$ could also compute a completely different function which does not correspond to set intersection at all: For example, a malicious $\mathcal{T}$ could output random values once it has obtained a value $y_i = 0$. In this case, the protocol computes some set $Z \subsetneq X \cap Y$ if $0 \in Y$, and $X \cap Y$ otherwise.

Protocol. We extend the protocol of Fig. 2 and adapt the oblivious transfer protocol of [30] to the set intersection scenario. We will therefore obtain both input privacy against malicious $\mathcal{A}$ and correctness against a covert $\mathcal{A}$ in the covert sense: $\mathcal{A}$ can actually succeed in violating the correctness of B's output with non-negligible probability but at the same time $\mathcal{B}$ can detect the cheating behavior of $\mathcal{A}$ with probability 1/2. The update of the protocol goes as follows: The basic idea is to let $\mathcal{T}$ compute two answers (using two different keys K, K^T), where $\mathcal{B}$ can verify the correctness of one answer ($\mathcal{B}$ obtains one key K^T from $\mathcal{A}$) without $\mathcal{T}$ knowing which one is verified. For this, $\mathcal{B}$ randomly chooses and sends to $\mathcal{A}$ a test value r^T and a distinct value r. Then, $\mathcal{B}$ obtains the test key $K^T = F_k(r^T)$ from $\mathcal{A}$, whereas the other key $K = F_k(r)$ remains unknown to $\mathcal{B}$ (to ensure this, $\mathcal{A}$ checks that $r^T \neq r$). Afterwards, $\mathcal{B}$ sends (r, r^T) to $\mathcal{T}$ in random order such that $\mathcal{T}$ can derive K, K^T without knowing which of them

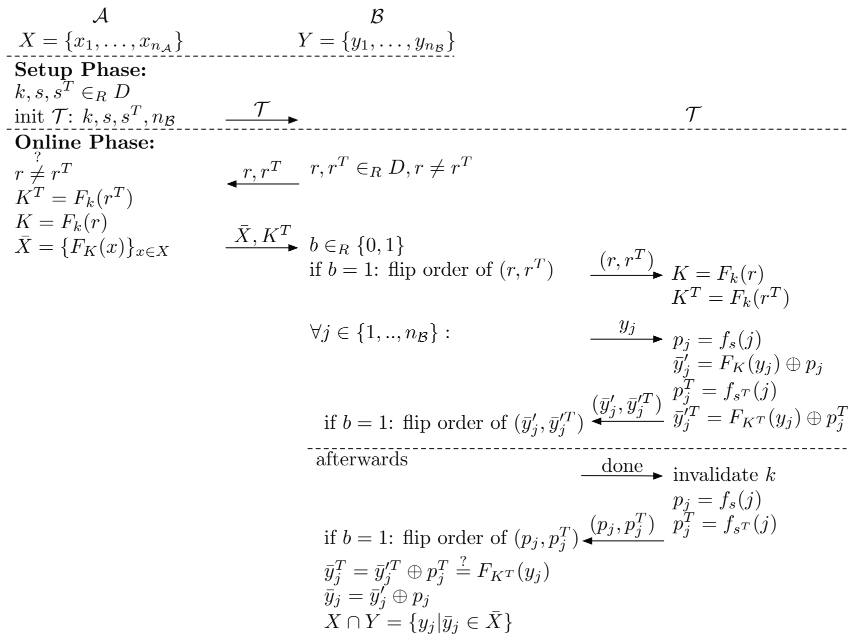

$\mathcal{A}$
$X = \{x_1, \ldots, x_{n_\mathcal{A}}\}$
$\mathcal{B}$
$Y = \{y_1, \ldots, y_{n_\mathcal{B}}\}$
Setup Phase:
$k, s, s^T \in_R D$
init $\mathcal{T}$: $k, s, s^T, n_\mathcal{B}$
$\mathcal{T}$
$\mathcal{T}$
Online Phase:
$r \overset{?}{\neq} r^T$
r, r^T
$r, r^T \in_R D, r \neq r^T$
$K^T = F_k(r^T)$
$K = F_k(r)$
$\bar{X} = \{F_K(x)\}_{x \in X}$
$\bar{X}, K^T$
$b \in_R \{0,1\}$
if $b = 1$: flip order of (r, r^T)
(r, r^T)
$K = F_k(r)$
$K^T = F_k(r^T)$
$\forall j \in \{1, .., n_\mathcal{B}\}$:
y_j
$p_j = f_s(j)$
$\bar{y}'_j = F_K(y_j) \oplus p_j$
$p_j^T = f_{s^T}(j)$
$\bar{y}'^T_j = F_{K^T}(y_j) \oplus p_j^T$
$(\bar{y}'_j, \bar{y}'^T_j)$
if $b = 1$: flip order of $(\bar{y}'_j, \bar{y}'^T_j)$
afterwards
done
invalidate k
$p_j = f_s(j)$
$p_j^T = f_{s^T}(j)$
(p_j, p_j^T)
if $b = 1$: flip order of (p_j, p_j^T)
$\bar{y}_j^T = \bar{y}'^T_j \oplus p_j^T \overset{?}{=} F_{K^T}(y_j)$
$\bar{y}_j = \bar{y}'_j \oplus p_j$
$X \cap Y = \{y_j | \bar{y}_j \in \bar{X}\}$

Fig. 3. Set Intersection Protocol with Privacy of $\mathcal{B}$'s Input and (Covert) Correctness of $\mathcal{B}$'s Output when $\mathcal{T}$ is not trusted by $\mathcal{B}$, and Privacy of $\mathcal{A}$'s input when $\mathcal{A}$ trusts $\mathcal{T}$

is known to $\mathcal{B}$. Then, for each element $y_j \in Y$, $\mathcal{B}$ obtains $\bar{y}_j = F_K(y_j)$ and $\bar{y}_j^T = F_{K^T}(y_j)$ from $\mathcal{T}$ (after removing the pads p_j and p_j^T as in the protocol of Fig. 2). As $\mathcal{B}$ knows the test key K^T it can test the correctness of $\bar{y}_j^T$, whereas $\mathcal{T}$ can only guess whether to cheat on $\bar{y}_j$ or $\bar{y}_j^T$. Finally, $\mathcal{B}$ computes the intersection from $\bar{X}$ and $\bar{Y}$ as before.

The overall protocol shown in Fig. 3 provides $\mathcal{A}$ with input privacy against a malicious $\mathcal{B}$, which cannot break into the token, and provides $\mathcal{B}$ with input privacy (Problem 1) against a malicious $\mathcal{A}$ and $\mathcal{T}$ and output correctness against a covert $\mathcal{A}$ and $\mathcal{T}$ (Problem 2).

Theorem 2. *If F is a SPRP and f is a PRF, then the protocol depicted in Fig. 3:*

1. *securely evaluates $\mathcal{F}_\cap(X, Y)$ w.r.t. a malicious $\mathcal{B}$ that cannot break into $\mathcal{T}$;*
2. *securely evaluates $\mathcal{F}_\cap(X, Y)$ w.r.t. a covert $\mathcal{A}$ with deterrence factor $\epsilon = 1/2$;*
3. *securely evaluates $\mathcal{F}_\cap(X, Y)$ when both parties trust the token.*

$\mathcal{B}$'s input is still (unconditionally) private even w.r.t. malicious $\mathcal{A}$, as in Property 2 of Theorem 1.

Proof (Sketch). To prove Theorem 2 we consider each property individually.

Malicious $\mathcal{B}$ that cannot break into $\mathcal{T}$. We show an ideal world adversary $\mathcal{B}_I$. This adversary $\mathcal{B}_I$ internally runs $\mathcal{B}$ simulating also $\mathcal{T}$'s answers. $\mathcal{B}_I$ sends to $\mathcal{B}$

a random vector of messages $\bar{X}$ and a random key K^T. When simulating $\mathcal{T}$'s answers before `done`, B_I plays honestly when test queries are performed (i.e., using K_T for the test queries along with the pseudorandom function indexed by s_T) and sending random messages otherwise, as already done in the proof of Theorem 1. When message `done` has been received, B_I plays in the ideal world the input extracted from the queries received by $\mathcal{T}$ and gets back the intersection Z. Here B_I proceeds by computing values p_j^T honestly, but adaptively computing all final p_j values so that the view of $\mathcal{B}$ will still be computationally indistinguishable, precisely as in the proof of Theorem 1.

Note that, since $\mathcal{A}$ checks that $r \neq r^T$, the pseudorandom keys K and K^T are computationally independent, and can be essentially replaced by independent random keys. A straightforward hybrid game shows that by the pseudorandomness of F this does not change $\mathcal{B}$'s success probability significantly.

Covert $\mathcal{A}$. Informally, the privacy of $\mathcal{B}$'s input is preserved as $\mathcal{A}$ does not obtain any message from $\mathcal{B}$ besides the random values r, r^T. The same argument which has been applied already in the proof of Theorem 1 about protecting $\mathcal{B}$'s input from a malicious sender, applies here as well. The more interesting difference however consists now in proving correctness of $\mathcal{B}$'s output in the covert sense: showing that a success of $\mathcal{A}$ in violating the correctness of $\mathcal{B}$'s output can be detected by $\mathcal{B}$ with probability $\epsilon = 1/2$, and this is achieved through the cut-and-choose construction of [30].

To formally prove the correctness of $\mathcal{B}$'s output we build a simulator *Sim* which plays as an honest $\mathcal{B}$ against adversaries $Adv_{\mathcal{A}}$ and $Adv_{\mathcal{T}}$ who control $\mathcal{A}$ and $\mathcal{T}$, respectively. As the token is not necessarily honest and hence a cheating $Adv_{\mathcal{A}}$ does not need to initialize $\mathcal{T}$ at all, *Sim* cannot learn the token's keys k, s, s^T from the initialization message sent from $Adv_{\mathcal{A}}$ to $Adv_{\mathcal{T}}$. Instead, *Sim* determines whether the adversary cheats in the protocol as follows: *Sim* obtains both opening keys K^T and K from $Adv_{\mathcal{A}}$, by rewinding $Adv_{\mathcal{A}}$ and swapping the order of (r, r^T). Afterwards, *Sim* can verify whether both values $\bar{y}_j, \bar{y}_j^T$ received from $Adv_{\mathcal{T}}$ are correct. If $Adv_{\mathcal{T}}$ tried to cheat (e.g., if the check of $\bar{y}_j^T$ failed), *Sim* catches $\mathcal{T}$ in doing so and issues the `cheat` instruction. *Sim* aborts in this case (losing any correctness guarantee in case the cheat is not announced). Otherwise, *Sim* continues to play as honest $\mathcal{B}$ and extracts $\mathcal{A}$'s inputs from $\bar{X}$ using K. Note that *Sim* simulates the ideal view of a covert $\mathcal{A}$ with deterrence factor $\epsilon = 1/2$, because for any run in which *Sim* does not receive both keys, $\mathcal{B}$ would detect cheating with probability 1/2 in the actual protocol, in which case it too aborts.

$\mathcal{A}$ and $\mathcal{B}$ trust the token. We now prove that when the token $\mathcal{T}$ is trusted, the protocol actually realizes the set intersection functionality (i.e., both input privacy in the simulation-based sense and output correctness are achieved). The proof follows closely the one of Theorem 1, indeed since $\mathcal{T}$ is honest, both $\mathcal{A}$'s and $\mathcal{B}$'s input can be extracted by receiving the queries to $\mathcal{T}$, moreover there is no issue of correctness since $\mathcal{T}$ never deviates from the protocol. The only issue to mention is that a malicious $\mathcal{A}$ could play a wrong third message, sending a wrong K^T. Therefore, the ideal world simulator $\mathcal{A}_I$ will first check that $\mathcal{A}$'s message is

well formed playing as honest $\mathcal{B}$, and only in case honest $\mathcal{B}$ would have obtained the output, $\mathcal{A}_I$ forwards the extracted input to the ideal functionality. □

Efficiency and Amplifying Deterrence Factor. Overall, the protocol in Fig. 3 approximately doubles the computation performed by $\mathcal{T}$ and the communication between $\mathcal{B}$ and $\mathcal{T}$ compared to the protocol in Fig. 2. The hardware requirements for the token are the same.

In analogy to [30], the deterrence factor ϵ can be increased by using n test elements r_i^T for which $\mathcal{B}$ obtains the corresponding test keys K_i^T from $\mathcal{A}$. Now, $\mathcal{T}$ can only successfully guess the key on which to cheat with probability $p = \frac{1}{n+1}$ s.t. $\epsilon = 1 - p$ is polynomially close to 1 in n. Obviously this is a tradeoff between deterrence factor and efficiency.

6 Only One Token Trusted: Privacy of $\mathcal{A}$'s Input

Model. In this section we extend the model of §4 so that not only $\mathcal{B}$ does not trust the tokens issued by $\mathcal{A}$, but also $\mathcal{B}$ is allowed to collude with all but one hardware manufacturer without $\mathcal{A}$ knowing which one. We show how to detect cheating in this model.

Problem 3 ($\mathcal{B}$ breaks into $\mathcal{T}$ to break privacy of $\mathcal{A}$'s inputs). In the protocols so far, a malicious $\mathcal{B}$ who can break into $\mathcal{T}$ (e.g., by a successful attack or by colluding with $\mathcal{M}$ who builds a trapdoor for $\mathcal{B}$ into $\mathcal{T}$) can obtain k and invert F to recover $\mathcal{A}$'s inputs from $\bar{X}$.

Protocol. To address Problem 3, we extend the protocol of Fig. 3 to multiple tokens as shown in Fig. 4: Instead of using one token, $\mathcal{A}$ uses two hardware tokens $\mathcal{T}_1$ and $\mathcal{T}_2$ manufactured by $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. Then, $\mathcal{A}$ embeds into each token $\mathcal{T}_i$ a different random key and runs the protocol using the sequential composition $F_{K'} = F_{K_2} \circ F_{K_1}$ instead of F_K, i.e., $\mathcal{B}$ communicates first with $\mathcal{T}_1$ and afterwards with $\mathcal{T}_2$. As long as at least one token is resistant against $\mathcal{B}$'s attacks, $\mathcal{B}$ cannot invert $F_{K'}$ and hence cannot recover $\mathcal{A}$'s inputs.

Theorem 3. *If F is a SPRP and f is a PRF, then the protocol depicted in Fig. 4:*

1. *securely evaluates $\mathcal{F}_\cap(X, Y)$ w.r.t. a malicious $\mathcal{B}$ that cannot break into all but one token $\mathcal{T}_i$;*
2. *securely evaluates $\mathcal{F}_\cap(X, Y)$ w.r.t. a covert $\mathcal{A}$ with deterrence factor $\epsilon = 1/2$;*
3. *securely evaluates $\mathcal{F}_\cap(X, Y)$ when both parties can trust all tokens.*

Proof (Sketch). The proof of Theorem 3 follows similarly to that of Theorem 2, but using multiple tokens where $\mathcal{B}$ can break into all but one.

Malicious $\mathcal{B}$ that can break into all but one token $\mathcal{T}_i$. Assume that $\mathcal{B}$ corrupts token $\mathcal{T}_1$ and thus learns k_1, s_1, and s_1^T. Then security for $\mathcal{A}$ follows as in the proof of Theorem 2 from the trustworthiness of $\mathcal{T}_2$, only that we consider the injectively transformed inputs through $F_{k_1}(\cdot)$. Analogously, if $\mathcal{B}$ corrupts $\mathcal{T}_2$ then security follows as before, because the outer function is easy to simulate.

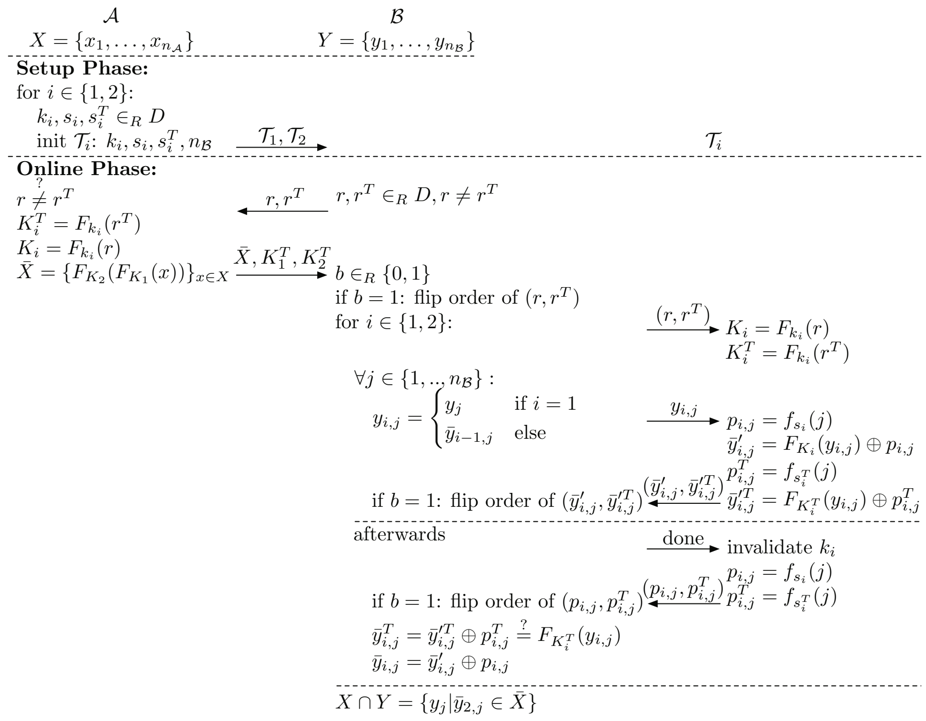

$\mathcal{A}$
$X = \{x_1, \ldots, x_{n_\mathcal{A}}\}$
$\mathcal{B}$
$Y = \{y_1, \ldots, y_{n_\mathcal{B}}\}$
Setup Phase:
for $i \in \{1,2\}$:
$k_i, s_i, s_i^T \in_R D$
init $\mathcal{T}_i$: $k_i, s_i, s_i^T, n_\mathcal{B}$
$\mathcal{T}_1, \mathcal{T}_2$
$\mathcal{T}_i$
Online Phase:
$r \overset{?}{\neq} r^T$
$K_i^T = F_{k_i}(r^T)$
$K_i = F_{k_i}(r)$
$\bar{X} = \{F_{K_2}(F_{K_1}(x))\}_{x \in X}$
r, r^T
$r, r^T \in_R D, r \neq r^T$
$\bar{X}, K_1^T, K_2^T$
$b \in_R \{0,1\}$
if $b = 1$: flip order of (r, r^T)
for $i \in \{1,2\}$:
(r, r^T)
$K_i = F_{k_i}(r)$
$K_i^T = F_{k_i}(r^T)$
$\forall j \in \{1,..,n_\mathcal{B}\}$:
$y_{i,j} = \begin{cases} y_j & \text{if } i = 1 \\ \bar{y}_{i-1,j} & \text{else} \end{cases}$
$y_{i,j}$
$p_{i,j} = f_{s_i}(j)$
$\bar{y}'_{i,j} = F_{K_i}(y_{i,j}) \oplus p_{i,j}$
$p_{i,j}^T = f_{s_i^T}(j)$
$\bar{y}'^T_{i,j} = F_{K_i^T}(y_{i,j}) \oplus p_{i,j}^T$
$(\bar{y}'_{i,j}, \bar{y}'^T_{i,j})$
if $b = 1$: flip order of $(\bar{y}'_{i,j}, \bar{y}'^T_{i,j})$
afterwards
done
invalidate k_i
$p_{i,j} = f_{s_i}(j)$
$p_{i,j}^T = f_{s_i^T}(j)$
$(p_{i,j}, p_{i,j}^T)$
if $b = 1$: flip order of $(p_{i,j}, p_{i,j}^T)$
$\bar{y}_{i,j}^T = \bar{y}'^T_{i,j} \oplus p_{i,j}^T \overset{?}{=} F_{K_i^T}(y_{i,j})$
$\bar{y}_{i,j} = \bar{y}'_{i,j} \oplus p_{i,j}$
$X \cap Y = \{y_j | \bar{y}_{2,j} \in \bar{X}\}$

Fig. 4. Set Intersection Protocol with Privacy of $\mathcal{B}$'s Inputs, (Covert) Correctness of $\mathcal{B}$'s Output and Privacy of $\mathcal{A}$'s Inputs when $\mathcal{A}$ trusts at least one Token

Covert $\mathcal{A}$. The only message $\mathcal{A}$ obtains from $\mathcal{B}$ are the random values r, r^T which do not depend on $\mathcal{B}$'s inputs, and this proves $\mathcal{B}$'s input privacy. For correctness of $\mathcal{B}$'s output, we observe that only one token can cheat while the other behaves correctly such that the probability of being caught remains 1/2. Alternatively, the two tokens could run a combined cheating strategy: token $\mathcal{T}_1$ which is queried first can only guess on which of the two values to cheat without being detected with probability 1/2. In case cheating is not detected, $\mathcal{T}_1$ can transfer information on which value it cheated successfully to $\mathcal{T}_2$ in the value $\bar{y}_{1,j}$. However, the combined cheating strategy will still be caught with probability at least 1/2.

$\mathcal{A}$ and $\mathcal{B}$ trust all tokens. In this case the protocol realizes the set intersection functionalities (i.e., both input privacy in the simulation-based sense and output correctness are achieved). The proof is similar to that of Theorem 2. □

Multiple Tokens and Efficiency. The protocol in Fig. 4 can be generalized to $n \geq 1$ tokens $\mathcal{T}_1, \ldots, \mathcal{T}_n$ manufactured by $\mathcal{M}_1, \ldots, \mathcal{M}_n$, where a malicious $\mathcal{B}$ is able to break all but one token. For $n = 1$, the protocol is equivalent to the protocol of Fig. 3, where $\mathcal{B}$ cannot break into the single token. With n tokens, the protocol in Fig. 4 is essentially a n-times repetition of the protocol in Fig. 3.

Acknowledgments. The work described in this paper was supported in part by the European Commission through the ICT program under contract 238811 UNIQUE, 216676 ECRYPT II, 216499 CACE, and 215270 FRONTS. The first author was supported by an Emmy Noether Grant Fi 940/2-1 of the German Research Foundation (DFG) and by CASED (`www.cased.de`). The second author was supported by the SFEROT project of the European Research Council. The last author was also supported in part by the MIUR Project PRIN "PEPPER: Privacy E Protezione di dati PERsonali" (prot. 2008SY2PH4).

References

1. Hazay, C., Lindell, Y.: Constructions of truly practical secure protocols using standard smartcards. In: CCS 2008, pp. 491–500. ACM, New York (2008)
2. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)
3. Sharangpani, H.P., Barton, M.L.: Statistical analysis of floating point flaw in the Pentium^{TM}processor. White paper, Intel Corporation (1994)
4. Biham, E., Carmeli, Y., Shamir, A.: Bug attacks. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 221–240. Springer, Heidelberg (2008)
5. Security, H.: Hacker extracts crypto key from TPM chip (2010), `http://www.h-online.com/security/news/item/Hacker-extracts-crypto-key-from-TPM-chip-927077.html`
6. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
7. Kissner, L., Song, D.X.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
8. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)
9. De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear computational and bandwidth complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (2010)
10. Ateniese, G., De Cristofaro, E., Tsudik, G.: (If) size matters: Size-hiding private set intersection. Cryptology ePrint Archive, Report 2010/220 (2010), `http://eprint.iacr.org/`
11. Sang, Y., Shen, H.: Privacy preserving set intersection protocol secure against malicious behaviors. In: PDCAT 2007, pp. 461–468. IEEE Computer Society, Los Alamitos (2007)
12. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
13. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient robust private set intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (2009)
14. Jarecki, S., Liu, X.: Fast secure computation of set intersection. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 418–435. Springer, Heidelberg (2010)

15. Hazay, C., Nissim, K.: Efficient set operations in the presence of malicious adversaries. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 312–331. Springer, Heidelberg (2010)
16. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. JoC 23, 422–456 (2010)
17. De Cristofaro, E., Kim, J., Tsudik, G.: Linear-complexity private set intersection protocols secure in malicious model. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 213–231. Springer, Heidelberg (2010)
18. Tate, S., Vishwanathan, R.: Improving cut-and-choose in verifiable encryption and fair exchange protocols using trusted computing technology. In: Gudes, E., Vaidya, J. (eds.) Data and Applications Security XXIII. LNCS, vol. 5645, pp. 252–267. Springer, Heidelberg (2009)
19. Fort, M., Freiling, F.C., Penso, L.D., Benenson, Z., Kesdogan, D.: Trustedpals: Secure multiparty computation implemented with smart cards. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 34–48. Springer, Heidelberg (2006)
20. Iliev, A., Smith, S.: More efficient secure function evaluation using tiny trusted third parties. Technical Report TR2005-551, Dartmouth College, Computer Science, Hanover, NH (2005)
21. Hofheinz, D., Müller-Quade, J., Unruh, D.: Universally composable zero-knowledge arguments and commitments from signature cards. In: MoraviaCrypt 2005 (2005)
22. 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)
23. Järvinen, K., Kolesnikov, V., Sadeghi, A.R., Schneider, T.: Garbled circuits for leakage-resilience: Hardware implementation and evaluation of one-time programs. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 383–397. Springer, Heidelberg (2010)
24. 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)
25. Moran, T., Segev, G.: David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008)
26. 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)
27. Damgård, I., Nielsen, J.B., Wichs, D.: Universally composable multiparty computation with partially isolated parties. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 315–331. Springer, Heidelberg (2009)
28. 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)
29. Järvinen, K., Kolesnikov, V., Sadeghi, A.R., Schneider, T.: Embedded SFE: Offloading server and network using hardware tokens. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 207–221. Springer, Heidelberg (2010)
30. Kolesnikov, V.: Truly efficient string oblivious transfer using resettable tamper-proof tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 327–342. Springer, Heidelberg (2010)
31. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 137–156. Springer, Heidelberg (2007)

Efficient Secure Two-Party Exponentiation*

Ching-Hua Yu[1], Sherman S.M. Chow[2], Kai-Min Chung[3], and Feng-Hao Liu[4]

[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan
[2] Combinatorics and Optimization, University of Waterloo, Ontario, Canada
[3] Department of Computer Science, Cornell University, New York, USA
[4] Department of Computer Science, Brown University, Rhode Island, USA

Abstract. We present a new framework to design secure two-party computation protocols for exponentiation over integers and over Z_Q where Q is a publicly-known prime. Using our framework, we realize efficient protocols in the semi-honest setting. Assuming the base is non-zero, and the exponent is at most $Q/2$ for the Z_Q case, our protocols consist of at most 5 rounds (each party sending 5 messages) and the total communication consists of a small constant number (≤ 18) of encrypted/encoded elements in Z_Q. Without these assumptions, our protocols are still more efficient than a protocol recently proposed by Damgård et al. in TCC 2006 (24 vs. $>$ 114 rounds, $\approx 279\ell + 12t$ for an error rate of 2^{-t} vs. $> 110\ell \log \ell$ secure multiplications, where ℓ is the bit length of the shares).

Our protocols are constructed from different instantiations of our framework with different assumptions (homomorphic encryption or oblivious transfers) to achieve different advantages. Our key idea is to exploit the properties of both additive and multiplicative secret sharing. We also propose efficient transformation protocols between these sharings, which might be of independent interest.

Keywords: two-party exponentiation, additive/multiplicative share.

1 Introduction

Secure two-party computation is one of the central topics in cryptography, where two parties Alice and Bob want to jointly compute a function $f(x_A, x_B)$ from their own secret inputs x_A and x_B without revealing any information about their inputs. General feasibility results have been developed in the 1980s [1–4], which the privacy of the honest party holds even if the other party deviates arbitrarily from the prescribed protocol (the malicious setting). However, the communication complexity of the protocols depends on the *boolean circuit* complexity of f, which is considered too inefficient for most practical applications.

In many cases such as privacy-preserving data-mining/statistical learning [5–8] and distributed generation of cryptographic keys [9], the desired functionality f involves mostly arithmetic operations such as addition, multiplication,

* A major part of the work was done while the second author was at New York University and the third author was at Harvard University. The third author is supported by US-Israel BSF grant 2006060 and NSF grant CNS-0831289.

division, and exponentiation over the integers and/or the finite fields. More efficient protocols for these basic operations can result in an more efficient protocol for f. Indeed, a significant effort has focused on designing protocols for these operations. For examples, Ishai, Prabhakaran, and Sahai [10, 11] studied general solutions for secure arithmetic computations over rings, which correspond to addition/subtraction and multiplication. Bunn and Ostrovsky [6] designed a division protocol, which is the essential part of their k-means clustering protocol. Damgård *et al.* [12] studied the exponentiation operation over integers modulo a public prime Q and gave the only constant-round protocol in this setting.

Information theoretical security is impossible for two-party computation in the plain model even if we only consider security against “semi-honest” adversaries [3, 13]. Hence, sending certain encrypted/encoded messages is necessary. For addition and multiplication, we know secure protocols with a small constant number of rounds involving a constant number of encrypted elements. However, for integer division and exponentiation, no efficient protocol that sends only a constant number of encrypted elements is known. (This is possible using a fully homomorphic encryption scheme [14], but the current candidates [14, 15] are very inefficient.) Indeed, there is no constant round protocol for division, and the best known protocol for exponentiation [12] requires a large constant number of rounds (> 114) and more than $110\ell \log \ell$ secure multiplications[1], where ℓ is the bit-length of the inputs, which are expensive for some applications.

Problem Statement. Motivated by building more efficient protocols for basic operations, we study semi-honest secure exponentiation over different domains with the goal of achieving efficiency comparable to multiplication. We remark that our protocols can be made secure against malicious adversaries by standard but rather expensive techniques such as zero-knowledge proofs. The possibility of an efficient security-strengthening transformation is outside our scope.

Our setting of secure two-party integer exponentiation is as follows. Two parties Alice and Bob receive inputs as secret shares of integers x and y, and the goal is to jointly compute a secret share of x^y. As usual, a predefined upper bound M on the result of computation is necessary.[2] For example, we are guaranteed that $x^y \leq M$. Given M, we can choose a (publicly known) prime Q which is sufficiently large (say $Q > M^2$) and embed the integers into $\mathbb{Z}_Q$. Namely, we associate integers $\{0, 1, 2, \ldots, Q-1\}$ with elements in $\mathbb{Z}_Q$ in a natural way, which induces an ordering on $\mathbb{Z}_Q$. The shares are taken to be additive shares over $\mathbb{Z}_Q$ – the input to Alice and Bob are (x_A, y_A) and (x_B, y_B) respectively such that $x = x_A + x_B \mod Q$ and $y = y_A + y_B \mod Q$, and the output of Alice and Bob is share z_A and z_B respectively such that $z_A + z_B = x^y \mod Q$.

We also consider modular exponentiation over $\mathbb{Z}_Q$, to compute additive shares of $x^y \bmod Q$, from additive shares of x and y over $\mathbb{Z}_Q$. Now, the exponent y can

[1] These come from the bit-decomposition which dominates the complexity [12].

[2] An example is **unsigned int** in C++ which sets $M = 2^{32} - 1$. We need to avoid overflow during the computation.

be any integer in $\{0, 1, \ldots, Q-1\}$, while the exponent for integer exponentiation is restricted to $y \leq M \ll Q$.

Our Results and Techniques. We present a new framework to design semi-honest secure two-party exponentiation protocols for the above two settings. The key idea is to exploit the properties of *both* additive and multiplicative secret sharing. We also propose efficient protocols for transformation between these sharings, which might be of independent interest. Using our framework, the resulting protocols for integer exponentiation (of non-zero base) use at most 5 rounds (each party sends at most 5 messages) and exchange a small constant number of encrypted/encoded elements. For modular exponentiation over $\mathbb{Z}_Q$, the resulting protocols improve the efficiency over the protocol of Damgård *et al.* [12], and achieve essentially the same efficiency as the integer exponentiation setting when the exponent is at most $Q/2$. A summary of our results is given in Table 1.

We present two implementations basing on homomorphic encryption schemes and oblivious transfers (OTs) (the latter uses the noisy encoding technique of Ishai, Prabhakaran, and Sahai [11]). All our protocols share the same framework and hence have similar round complexity. In the following, we elaborate the advantages of our different implementations.

- **Homomorphic encryption**-based approach achieves the best efficiency in terms of communication and computation.
- **Oblivious transfers**-based approach inherits the versatility of OTs [10]. Our protocols can be realized by many different number theoretic assumptions or even reach information-theoretic security with physical assumptions (e.g., binary symmetric channel). Furthermore, OTs can be precomputed [10, 11] which makes the online efficiency better than the encryption-based protocol for certain parameter ranges.

Related Works and Comparison. One can consider secure computation for two-party or multi-party, and there are differences between semi-honest and malicious security. We focus on the semi-honest two-party setting, but our framework can be extended to the multi-party setting (without honest-majority).

There are a variety of settings for exponentiation depending on whether the base x, the exponent y, and the modulus Q are shared or public. For the most general setting where x, y, Q are shared among all parties, existing solution [16] considers (information-theoretic) semi-honest secure multi-party computation with *honest majority*, hence it is not trivial to be adapted to the two-party setting. On the other hand, Damgård *et al.* [12] considered the same setting as us, where x and y are shared and Q is public. They mentioned that their results can be extended to the general setting of shared Q by combining their technique with [16] and [17]. However, they did not explicitly analyze the complexity of their solution in this setting. Their construction is based on the existence of secure multiplication on linear shares over $\mathbb{Z}_Q$, so it works for both multi-party with honest majority and two-party. A simpler setting where only y is private, x and Q are both public, has been considered [5, 18].

We summarize our results and the related existing results [12, 16] in Table 1. The round complexity counts the maximum number of messages Alice or Bob sends, and the communication denotes the total communication complexity of the protocol, which is the total umber of "unit messages" (i.e., ciphertexts/noisy encodings/field elements) sent by the two parties. A "ciphertext" refers to that produced by ElGamal or Paillier encryption schemes. On the other hand, one noisy encoding requires sending roughly $O(k+\ell)$ field elements (i.e., $O(\ell(k+\ell))$ bits) and uses $O(k+\ell)$ calls to OT, where k is a security parameter for a security level of 2^k and $\ell = \log Q$ is the length of elements of the field $\mathbb{Z}_Q$. In practice, depending on the size of field $\mathbb{Z}_Q$, the length of ciphertext k_{Enc} can be larger or smaller than the length of noisy encodings $O(\ell(k+\ell))$. Two-party secure multiplication can also be implemented in different ways as secure exponentiation, which require 4 "unit messages". Our results and the result of Damgård *et al.* [12] are the only constant-round protocols for secure exponentiation (when both x, y are shared), and both results work for integer exponentiation and modular exponentiation modulo a prime[3] Q. The result of Algesheimer, Camenisch, and Shoup [16] is for multi-party with honest majority.

We remark that our results for the general case of modular exponentiation over $\mathbb{Z}_Q$ require a zero-test and a comparison sub-protocol. Both of them are quite expensive in comparison with the rest of our protocol, and in fact, dominate the complexity of our protocols. In particular, the zero-test protocol due to Nishide and Ohta [19] requires 4 rounds and $12t$ secure multiplications to achieve an error rate 2^{-t}. Their comparison protocol [19] requires 15 rounds and $279\ell + 5$ secure multiplications which is the only reason that makes the number of communication elements of our protocol depending on the field size.

2 Our Framework

Our framework exploits the properties of *both* additive and multiplicative secret sharing. A key observation is that exponentiation is very simple when the base x is shared in multiplicative form and the exponent y is shared in additive form (over $\mathbb{Z}_{Q-1}$). All we need is to convert the shares of x and y into the desired form. We formalize our framework in terms of the protocols as described in Figure 1:

1. Alice and Bob convert their additive shares of ($x = x_A + x_B \mod Q$) to multiplicative shares ($x = x'_A \cdot x'_B \mod Q$). This corresponds to the protocol A2M (additive sharing to multiplicative sharing) that converts shares from the additive form to the multiplicative form.
2. Alice and Bob convert their additive shares of ($y = y_A + y_B \mod Q$) to additive shares ($y = y'_A + y'_B \mod (Q-1)$) with $y'_A, y'_B \in \mathbb{Z}_{Q-1}$. This corresponds to the modular reduction protocol ModRed, which converts additive shares of y over $\mathbb{Z}_Q$ to $\mathbb{Z}_{Q-1}$.

[3] Our results extend to a general modulus N if $\varphi(N)$ is available.

Table 1. A summary of our results and existing protocols [12, 16] for computing additive shares of $x^y \mod Q$ from additive shares of x and $y \mod Q$. ($\ell = \log Q$ is the length of elements of the field $\mathbb{Z}_Q$; k denotes the security parameter to achieve security 2^k; t denotes the correctness parameter to achieve an error rate 2^{-t}; k_{Enc} denotes the ciphertext length of the ElGamal or Paillier encryption schemes. We consider 1 secure multiplication requires 4 ciphertexts/noisy encodings/field elements, depending on the implementation. Modular Exp. over $\mathbb{Z}_Q$ in Sec. 3 is only for a safe prime Q.)

Setting	Protocol	Rounds	Communication
Integer Exp. $x \neq 0$	Sec. 3	5	18 ciphertexts $= 18k_{\mathsf{Enc}}$ bits
	Sec. 4	3	10 noisy encodings $= O(\ell \cdot (\ell + k))$ bits
Integer Exp. arbitrary x	Extending above	Above plus 4	Above plus a zero-test (+$12t$ secure multiplications)
Modular Exp. over $\mathbb{Z}_Q$ $x \neq 0$ and $y \leq Q/2$	Sec. 3	5	18 ciphertexts $= 18k_{\mathsf{Enc}}$ bits
	Sec. 4	3	10 noisy encodings $= O(\ell \cdot (\ell + k))$ bits
Modular Exp. over $\mathbb{Z}_Q$ for general case	Extending above	Above plus 19	Above plus a zero-test and a comparison (+$12t + 279\ell + 5$ secure multiplications)
	[12]	> 114	$> 110\ell \log \ell$ secure multiplications
Modular Exp. over a shared secret	[12]	$O(1)$	$O(\ell \log \ell)$ secure mult. (large constants)
	[16]	$O(\ell)$	$O(\ell^2)$ bits (with a large constant)

3. Alice and Bob jointly compute multiplicative shares of $z' = ((x'_A)^{y'_B} \cdot (x_B)'^{y'_A}) = z'_A \cdot z'_B \mod Q$. This uses the protocol SP ("scalar product"[4]), which computes multiplicative shares of $(x'_A)^{y'_B} \cdot (x'_B)^{y'_A}$. We have

$$x^y = (x'_A)^{(y'_A+y'_B)}(x'_B)^{(y'_A+y'_B)} = (x'^{y'_A}_A) \cdot ((x'_A)^{y'_B} \cdot (x'_B)^{y'_A}) \cdot (x'^{y'_B}_B) \pmod{Q}$$

by the identity $a^{Q-1} = 1 \pmod{Q}$ $\forall a \in \mathbb{Z}_Q^*$. The terms $x'^{y'_A}_A$ and $x'^{y'_B}_B$ can be computed locally by Alice and Bob, respectively. As a result, they have multiplicative shares of $x^y = (z'_A \cdot x'^{y'_A}_A \mod Q) \cdot (z'_B \cdot x'^{y'_B}_B \mod Q)$.

4. Alice and Bob locally compute $(z'_A \cdot x'^{y'_A}_A \mod Q)$ and $(z'_B \cdot x'^{y'_B}_B \mod Q)$, and convert the multiplicative shares back to the additive shares. $z_A + z_B = x^y = (z'_A \cdot x'^{y'_A}_A) \cdot (z'_B \cdot x'^{y'_B}_B) \mod Q$. This step requires the protocol $\mathsf{M2A}$ (multiplicative sharing to additive sharing) that converts shares from the multiplicative form to the additive form.

Our exponentiation protocol is a straightforward composition of the above four protocols. A formal description is given in Figure 2. Note that a secure multiplication over $\mathbb{Z}_Q$ can be composed by two invocations of $\mathsf{M2A}$. Our implementations of $\mathsf{M2A}$ based on homomorphic encryption schemes requires a new idea, which is inspired by the existing integer sharing schemes [16]. In the rest of the paper, we will present implementations of the above steps basing on different assumptions. However, there are two subtleties to be addressed in the next two paragraphs.

[4] It is possible to define an "exponential module" in such a way that the cross term $x_A^{y_B} \cdot x_B^{y_A}$ is the inner product of two module elements. Indeed, this is the reason that we call it the scalar product protocol.

A2M Protocol, denoted as $(z_A, z_B) \leftarrow (A(x_A), B(x_B))_{\mathsf{A2M}}$.

- **Inputs:** Alice holds $x_A \in \mathbb{Z}_Q$, and Bob holds $x_B \in \mathbb{Z}_Q$, where $x = x_A + x_B \in \mathbb{Z}_Q^*$.
- **Outputs:** Alice obtains z_A, and Bob obtains z_B such that $z_A \cdot z_B = x$.

Modular Reduction Protocol, denoted as $(z_A, z_B) \leftarrow (A(x_A), B(x_B); Q)_{\mathsf{ModRed}}$.

- **Inputs:** Alice holds $x_A \in \mathbb{Z}_Q$, and Bob holds $x_B \in \mathbb{Z}_Q$, and $x = x_A + x_B \in \mathbb{Z}_Q$.
- **Outputs:** Alice obtains $z_A \in \mathbb{Z}_{Q-1}$, and Bob obtains $z_B \in \mathbb{Z}_{Q-1}$ such that $z_A + z_B = x \in \mathbb{Z}_{Q-1}$.

Scalar Product Protocol, denoted as $(z_A, z_B) \leftarrow (A(x_A, y_A), B(x_B, y_B))_{\mathsf{SP}}$.

- **Inputs:** Alice holds $x_A \in \mathbb{Z}_Q$, $y_A \in \mathbb{Z}_{Q-1}$ and Bob holds $x_B \in \mathbb{Z}_Q$, $y_B \in \mathbb{Z}_{Q-1}$, where $x_A \cdot x_B \in \mathbb{Z}_Q^*$.
- **Outputs:** Alice obtains z_A, and Bob obtains z_B such that $z_A \cdot z_B = x_A^{y_B} \cdot x_B^{y_A}$.

M2A Protocol, denoted as $(z_A, z_B) \leftarrow (A(x_A), B(x_B))_{\mathsf{M2A}}$.

- **Inputs:** Alice holds $x_A \in \mathbb{Z}_Q^*$, and Bob holds $x_B \in \mathbb{Z}_Q^*$, where $x = x_A \cdot x_B \in \mathbb{Z}_Q^*$.
- **Outputs:** Alice obtains z_A, and Bob obtains z_B such that $z_A + z_B = x$.

Fig. 1. The interfaces for Protocol A2M, ModRed, SP, and M2A

Non-Zero Base in Multiplicative Shares. Our first step does not make sense when $x = 0$ since x has no multiplicative shares. This is also an "exception case" of the protocol of Damgård *et al.* [12]. To handle this, they use a trick to make x always non-zero, which can also be applied to our protocol.[5] However, it costs an additional equality-test protocol, which is relatively cheap when compared with their protocol, but is more expensive than the complexity of our protocol. Hence, we suggest avoiding using that when it can be safely assumed x is non-zero.

Modular Reduction. The second issue is to construct an efficient ModRed protocol. In cases where y is already given as shares in $\mathbb{Z}_{Q-1}$ (e.g., when y is randomly generated by two parties), we do not need to do modular reduction at all. When the input x is guaranteed to be less than $Q/2$, we present a simple implementation in Figure 3 using an M2A protocol. Note that for the setting of integer exponentiation, the condition $y \leq Q/2$ holds for free since we are guaranteed

[5] Define $x' = x + (x \overset{?}{=} 0)$, where $(x \overset{?}{=} 0)$ is 1 if $x = 0$, and is 0 otherwise. With a secure equality-test protocol that computes additive shares of $(x \overset{?}{=} 0)$, we can use the identity $x^y = x'^y - (x \overset{?}{=} 0)$ to compute x^y avoiding the base being zero [12].

Exponentiation Protocol:

- **Inputs:** Alice holds $x_A, y_A \in \mathbb{Z}_Q$, Bob holds $x_B, y_B \in \mathbb{Z}_Q$, where $x = x_A + x_B \in \mathbb{Z}_Q^*$, $y = y_A + y_B \in \mathbb{Z}_Q$.
- **Outputs:** Alice obtains $z_A \in \mathbb{Z}_Q$, and Bob obtains $z_B \in \mathbb{Z}_Q$ such that $z_A + z_B = x^y$.

1. Alice and Bob run $(x'_A, x'_B) \leftarrow (A(x_A), B(x_B))_{\mathsf{A2M}}$.
2. Run $(y'_A, y'_B) \leftarrow (A(y_A), B(y_B); Q)_{\mathsf{ModRed}}$.
3. Run $(z'_A, z'_B) \leftarrow (A(x'_A, y'_A), B(x'_B, y'_B))_{\mathsf{SP}}$.
4. Run $(z_A, z_B) \leftarrow (A(x_A'^{y'_A} \cdot z'_A), B(x_B'^{y'_B} \cdot z'_B))_{\mathsf{M2A}}$.

Fig. 2. The interface of the exponentiation protocol and its implementation

that $y \leq M \ll Q$. The security and efficiency of the ModRed protocol follow from those of M2A, and its correctness is proved in the following lemma.

Lemma 1. *For a prime $Q \in \mathbb{Z}$, $x_A, x_B \in \mathbb{Z}_Q$, and $x = x_A + x_B \in \mathbb{Z}_Q$. Suppose $x \leq Q/2$, then $(z_A, z_B) \leftarrow (A(x_A), B(x_B); Q)_{\mathsf{ModRed}}$ form shares of x in $\mathbb{Z}_{Q-1}$.*

Proof. From the construction, it is clear that $z_A, z_B \in \mathbb{Z}_{Q-1}$. From the property of M2A protocol, we know that $z'_A + z'_B = b_A \cdot b_B \mod (Q-1)$. So we know that $z'_A + z'_B = 1 \mod (Q-1)$ if and only if $x_A \leq Q/2$ and $x_B \leq Q/2$. This implies $x = x_A + x_B \leq Q - 1$. Thus, we know that $x = z_A + z_B \mod (Q-1)$.

On the other hand, if $z'_A + z'_B = 0 \mod (Q-1)$, at least one of x_A, x_B is greater than $Q/2$, and thus $2Q > x_A + x_B > Q/2 > x$. This means $x_A + x_B = x + Q$, thus we can see $z_A + z_B = (x_A + x_B - Q) \mod (Q-1) = x$ (as $x \leq Q/2$).

Note that in Step 3, Alice and Bob require to run M2A protocol over $\mathbb{Z}_{Q-1}$ (corresponding to the inputs and outputs of the invocation). This is valid in our scheme even though $Q-1$ is not a prime. □

For the general case, however, we do not know an efficient protocol with complexity independent of the bit-length $\ell = \log Q$. A natural idea to implement ModRed is to test whether $x_A + x_B \geq Q$ or $x_A + x_B < Q$, and then subtract Q before taking mod $Q-1$ if it is the former case. However, this idea requires a secure comparison protocol (e.g. [19]), which is expensive.

Semi-honest Security. We informally define the semi-honest security in the two party case and refer the readers to the literature (e.g., [5, 8]) for the standard formal definition. For a deterministic functionality $f(\cdot,\cdot)$, a protocol is said to be *semi-honest secure* if for any honest-but-curious adversary $\mathcal{A}$ who corrupts the first party, there exists a probabilistic polynomial time simulator $\mathcal{S}$, who gets the inputs and the randomness of $\mathcal{A}$, can produce a view of $\mathcal{A}$ which is (statistically/computationally) indistinguishable against time 2^k distinguisher

Modular Reduction Protocol for a public prime number $Q \in \mathbb{Z}$ (for the case where $x \leq Q/2$):

- **Inputs:** Alice holds $x_A \in \mathbb{Z}_Q$, and Bob holds $x_B \in \mathbb{Z}_Q$, and $x = x_A + x_B \mod Q$ such that $x \leq Q/2$.
- **Outputs:** Alice obtains z_A, and Bob obtains z_B such that $z_A + z_B = x \mod (Q-1)$.

1. Alice locally computes a number b_A such that $b_A = 1 \mod (Q-1)$ iff $x_A \leq Q/2$ otherwise 0.
2. Bob locally computes a number b_B such that $b_B = 1 \mod (Q-1)$ iff $x_B \leq Q/2$ otherwise 0.
3. Run $(z'_A, z'_B) \leftarrow (A(b_A), B(b_B))_{\mathsf{M2A}}$.
4. Alice outputs $z_A = (x_A + z'_A \cdot Q) \mod (Q-1)$, and Bob outputs $z_B = (x_B + (z'_B - 1) \cdot Q) \mod (Q-1)$

Fig. 3. Protocol ModRed

with advantage 2^{-k} from the real interaction with the second party, where k is a security parameter. Similar case should hold for the corrupted second party.

3 Implementation Using Homomorphic Encryption

Now we present our implementation of the protocols M2A, A2M, SP using homomorphic encryption schemes. Formal description can be found in Figure 4-6.

Homomorphic Encryption. A homomorphic encryption scheme (Gen, Enc, Dec) has both the message space and ciphertext space associated with certain algebraic structure and the encryption function Enc is homomorphic with respect to the corresponding operation in both spaces. There are several efficient public-key homomorphic encryption schemes which possess different homomorphic properties. We will use both the ElGamal encryption scheme [21] and the Paillier encryption scheme [22]. The ElGamal encryption scheme is semantically secure over *the subgroup of quadratic residue* $H \overset{\text{def}}{=} \{x^2 : x \in \mathbb{Z}_Q^*\} \subset \mathbb{Z}_Q^*$, when Q is a safe prime (i.e., $Q = 2P + 1$ where P is also a prime) from a common belief that the decisional Diffie-Hellman (DDH) assumption holds for H, and possesses *multiplicative homomorphism.* On the other hand, the Paillier encryption scheme is semantically secure over $\mathbb{Z}_N$ for a composite number $N = PQ$ under the decisional composite residuosity assumption, and possesses *additive homomorphism.*

We assume that the encryption schemes with security parameter k is semantic secure against time 2^k adversary with advantage 2^{-k}, and our protocol achieve semi-honest security against time $O(2^k/T)$ distinguisher with advantage $O(2^{-k})$, where T is the run-time of the protocol. To achieve this, we also require that the modulo N of the Paillier encryption scheme satisfying $N \geq 20 \cdot 2^k Q^2$. This is a mild requirement[6] and can be satisfied by using a larger security parameter.

Our Implementations. We first observe that if we have additive (resp., multiplicative) homomorphic encryption schemes over $\mathbb{Z}_Q$ (resp., $\mathbb{Z}_Q^*$), then secure $\mathsf{A2M}, \mathsf{M2A}$ (resp., SP) protocols are very easy to achieve — we can let Alice send encryption of her input to Bob, who can then perform computation homomorphically, and send back an encrypted share to Alice. Intuitively, Bob can learn nothing since he only receives encrypted messages from Alice; and Alice also learns nothing, since she only receives a share of the computed value from Bob.

Unfortunately, the Paillier encryption scheme is only semantically secure over $\mathbb{Z}_N$ for a composite number N, and the ElGamal encryption scheme is semantically secure over the subgroup of quadratic residue in $\mathbb{Z}_Q^*$ when Q is a safe prime. These make the implementation of $\mathsf{A2M}, \mathsf{M2A}$ and SP protocols non-trivial. At a high level, we overcome these difficulties with the following ideas.

– To implement $\mathsf{A2M}$ and $\mathsf{M2A}$ using the Paillier encryption scheme over $\mathbb{Z}_N$, we exploit an idea inspired by the integer sharing schemes of Algesheimer, Camenisch, and Shoup [16]. Briefly, instead of using secret sharing to hide the secret x (which can only be done for additively homomorphism over $\mathbb{Z}_Q$), we require $N \gg Q$ and use a random noise to *statistically* hide the secret.
– Implementing SP using the ElGamal encryption scheme over $H \subset \mathbb{Z}_Q^*$ is trickier. Very briefly, note that $\mathbb{Z}_Q^* = B \times H$, where B is the binary subgroup of Legendre symbols, our idea is to handle the H and B parts of $\mathbb{Z}_Q^*$ separately, where the H part is handled by ElGamal, and the B part is handled by two calls to $\mathsf{A2M}$ and $\mathsf{M2A}$ protocols.

Our implementation of the three protocols can be found in Figure 4–6. We proceed to explain the details of our implementation as follows.

– $\mathsf{M2A}$ protocol (Figure 4): Alice sends her encrypted input $\hat{x}_A = \mathsf{Enc}(x_A)$ to Bob, who can homomorphically compute encrypted secret $\hat{x} = \mathsf{Enc}(x_A \cdot x_B) = \mathsf{Enc}(x_A)^{x_B}$ of x using the additively homomorphic property of Enc. Bob then wants to split the secret x into additive shares, so he selects a random $u \in \mathbb{Z}_Q$ and computes encrypted share $\mathsf{Enc}(x+u) = \mathsf{Enc}(x) \cdot \mathsf{Enc}(u)$ and his share $-u$. However, Paillier encryption is additively homomorphic over $\mathbb{Z}_N$ with $N \gg Q$, the resulting $x+u$ is a number between 0 and Q^2+Q and Bob cannot send $\mathsf{Enc}(x+u)$ back to Alice directly (since it leaks partial information). Hence, Bob uses a large random noise w (say, in $[-N/10, N/10]$) to hide the secret x and sends $\mathsf{Enc}(w+u+x)$ to Alice. On the other hand, to help Alice to find out $x+u \mod Q$, Bob also sends $w \mod Q$ to Alice, who can then recover $x+u \mod Q$. Note that the noise hides $x+u$ statistically.
– $\mathsf{A2M}$ protocol (Figure 5): Its idea and structure are similar to those of $\mathsf{M2A}$.

[6] It is satisfied automatically unless $\log Q \gg k$.

```
M2A Protocol $(z_A, z_B) \leftarrow (A(x_A), B(x_B))_{\mathsf{M2A}}$
 1. Alice generates a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$, and sends $\mathsf{pk}$ and $\hat{x}_A = \mathsf{Enc}_{\mathsf{pk}}(x_A)$ to Bob. (Recall that we require $N \geq 20 \cdot 2^k Q^2$.)
 2. Bob samples uniformly random $u \leftarrow \mathbb{Z}_Q$ and $w \leftarrow [-N/10, N/10]$, computes $\hat{v} = \mathsf{Enc}_{\mathsf{pk}}(w + u + x_A \cdot x_B)$ and sends $\hat{v}$ and $v' = (w \mod Q)$ to Alice.
 3. Alice outputs $z_A = (\mathsf{Dec}_{\mathsf{sk}}(\hat{v}) - v' \mod Q) = (u + (x_A \cdot x_B) \mod Q)$.
 4. Bob outputs $z_B = -u$.
```

Fig. 4. Implementation of M2A using (additively) homomorphic encryption

– SP protocol (Figure 6): The SP protocol involves three parts: 1) converting the messages from $\mathbb{Z}_Q^*$ into H $(x_A, x_B \to x'_A, x'_B)$ accompanied parity bits $(z_{A,1}, z_{B,1}, z_{A,2}, z_{B,2})$ in Step 1-2, 2) computing (an encrypted share of) $x_A'^{y_B} x_B'^{y_A}$ in Step 3-8, and 3) recovering the messages from H to $\mathbb{Z}_Q^*$.

1) Alice and Bob convert the messages into H, where the DDH assumption is believed to hold, and so ElGamal encryption is semantically secure. Note that in $\mathbb{Z}_Q^*$, exactly one of x_A and $-x_A$ is a quadratic residue, and we denote such one in H as m. From this we can calculate $x_A^{y_B}$ (in $\mathbb{Z}_Q^*$) by first calculating m^{y_B} in H and then multiplying the outcome with a number $b \in \{1, -1\}$ depending on $m = x_A$ or $m = -x_A$ and the parity of y_B. To formalize this, we use two boolean variables ℓ_A, t_B and set $\ell_A = 1$ in the case where x_A is a quadratic residue and $t_B = 1$ if y_B is odd. When $\ell_A = 1$ and $t_B = 1$, we know $x_A^{y_B} = -1 \cdot m^{y_B}$; and for all the other cases, we have $x_A^{y_B} = m^{y_B}$. This is equivalent to compute $(1 - 2 \cdot \ell_A \cdot t_B) \cdot m^{y_B}$, where additive shares of $(1 - 2 \cdot \ell_A \cdot t_B)$ can be computed using one M2A and A2M. That is, we can think of Alice holding $w_A = 2\ell_A$, and Bob holding $w_B = t_B$, and view them as multiplicative shares of $w = w_A \cdot w_B$. We can then apply M2A to turn them into additive shares $w = w'_A + w'_B$ (which is $2\ell_A \cdot t_B$). Alice and Bob can locally compute $u_A = 1 - w'_A$, and $u_B = -w'_B$, so that $u = u_A + u_B$ is an additive share of $1 - 2\ell_A \cdot t_B$.

2) Both parties send $\hat{x}_A = \mathsf{Enc}_{\mathsf{pk}_A}(x'_A)$ and $\hat{x}_B = \mathsf{Enc}_{\mathsf{pk}_B}(x'_B)$ to each other. Upon receiving $\hat{x}_A$ and $\hat{x}_B$, they can compute $\mathsf{Enc}_{\mathsf{pk}_A}(x_A'^{y_B}) = \hat{x}_A^{y_B}$ and $\mathsf{Enc}_{\mathsf{pk}_B}(x_B'^{y_A}) = \hat{x}_B^{y_A}$ using the multiplicatively homomorphic property of Enc. To protect their own privacy, they split these values into multiplicative shares, and send each other an encrypted share. For example, Bob splits the encrypted $x_A'^{y_B}$ into $(u_B \cdot x_A'^{y_B}) \cdot (u_B^{-1})$ for a random u_B, and sends an encrypted $(u_B \cdot x_A'^{y_B})$ to Alice. Finally, they can locally combine their shares of $x_A'^{y_B}$ and $x_B'^{y_A}$ into shares of $x_A'^{y_B} \cdot x_B'^{y_B}$.

3) Both parties combines shares into the final output. From Step 1-2, they have $x_A^{y_B} = (1 - 2 \cdot \ell_A \cdot t_B) \cdot x_A'^{y_B} = z_{A,1} \cdot z_{B,1} \cdot x_A'^{y_B}$ and $x_B^{y_A} = (1 - 2 \cdot \ell_B \cdot t_A) \cdot x_B'^{y_A} = z_{A,2} \cdot z_{B,2} \cdot x_B'^{y_A}$; and from Step 3-8, $x_A'^{y_B} x_B'^{y_A} = z_{A,3} \cdot z_{B,3}$. These lead to $x_A^{y_B} x_B^{y_A} = (z_{A,1} \cdot z_{A,2} \cdot z_{A,3}) \cdot (z_{B,1} \cdot z_{B,2} \cdot z_{B,3})$.

A2M Protocol $(z_A, z_B) \leftarrow (A(x_A), B(x_B))_{\mathsf{A2M}}$

1. Alice generates a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$, and sends pk and $\hat{x}_A = \mathsf{Enc}_{\mathsf{pk}}(x_A)$ to Bob. (Recall that we require $N \geq 20 \cdot 2^k Q^2$.)
2. Bob samples uniformly at random $u \leftarrow \mathbb{Z}_Q^*$ and $w \leftarrow [-N/10, N/10]$, computes $\hat{v} = \mathsf{Enc}_{\mathsf{pk}}(w + u \cdot (x_A + x_B))$ and sends $(\hat{v}, v')$ to Alice.
3. Alice outputs $z_A = (\mathsf{Dec}_{\mathsf{sk}}(\hat{v}) - v' \mod Q) = (u \cdot (x_A + x_B) \mod Q)$.
4. Bob outputs $z_B = u^{-1}$.

Fig. 5. Implementation of A2M using (additively) homomorphic encryption

SP Protocol $(z_A, z_B) \leftarrow (A(x_A, y_A), B(x_B, y_B))_{\mathsf{SP}}$

Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be ElGamal encryption over H and $\left(\frac{a}{Q}\right)$ be the Legendre symbol.

1. Let $\ell_A = 1$ if $\left(\frac{x_A}{Q}\right) = -1$, and $\ell_A = 0$ if $\left(\frac{x_A}{Q}\right) = 1$. Alice sets $x'_A = x_A \cdot (-1)^{\ell_A}$. Similarly, let $\ell_B = 1$ if $\left(\frac{x_B}{Q}\right) = -1$ and 0 otherwise. Bob sets $x'_B = x_B \cdot (-1)^{\ell_B}$.
2. Let $t_A = y_A \mod 2$, and $t_B = y_B \mod 2$. Alice and Bob run two secure sub-protocols (each with one M2A and one A2M) to obtain $(z_{A,1}, z_{B,1})$ such that $z_{A,1} \cdot z_{B,1} = 1 - 2 \cdot \ell_A \cdot t_B$, and $(z_{A,2}, z_{B,2})$ such that $z_{A,2} \cdot z_{B,2} = 1 - 2 \cdot \ell_B \cdot t_A$.
3. Alice generates $(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{Gen}(1^k)$, and sends pk_A, $\hat{x}_A = \mathsf{Enc}_{\mathsf{pk}_A}(x'_A)$ to Bob.
4. Bob generates $(\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow \mathsf{Gen}(1^k)$, and sends pk_B, $\hat{x}_B = \mathsf{Enc}_{\mathsf{pk}_B}(x'_B)$ to Alice.
5. Alice computes $\hat{u}_A = \mathsf{Enc}_{\mathsf{pk}_B}(u_A)$, and sends $\hat{v}_A = \hat{u}_A \cdot \hat{x}_B^{y_A}$ to Bob, for $u_A \leftarrow Z_Q^*$.
6. Bob computes $\hat{u}_B = \mathsf{Enc}_{\mathsf{pk}_A}(u_B)$, and sends $\hat{v}_B = \hat{u}_B \cdot \hat{x}_A^{y_B}$ to Alice, for $u_B \leftarrow Z_Q^*$.
7. Alice computes $z_{A,3} = u_A^{-1} \cdot \mathsf{Dec}_{\mathsf{sk}_A}(\hat{v}_B) = (u_B \cdot x_A'^{y_B})/u_A$.
8. Bob computes $z_{B,3} = u_B^{-1} \cdot \mathsf{Dec}_{\mathsf{sk}_B}(\hat{v}_A) = (u_A \cdot x_B'^{y_A})/u_B$.
9. Alice outputs $z_A = z_{A,1} \cdot z_{A,2} \cdot z_{A,3}$, and Bob outputs $z_B = z_{B,1} \cdot z_{B,2} \cdot z_{B,3}$.

Fig. 6. Implementation of SP using (multiplicatively) homomorphic encryption

Efficiency. Suppose the key generation has been done in a setup stage, we have:

- M2A protocol: Alice needs to do 1 encryption and 1 decryption. Bob needs to do 1 encryption and 1 exponentiation with exponent $x_B \in Z_Q$. The protocol consists of 1 round and 2 messages, where Alice's message is a ciphertext, and Bob's message is a ciphertext plus an element in $\mathbb{Z}_Q$.
- A2M protocol: Alice needs to do 1 encryption and 1 decryption. Bob needs to do 2 encryptions and 1 exponentiation with exponent $u \in Z_Q$. The protocol consists of 1 round and 2 messages, where Alice's message is a ciphertext, and Bob's message is a ciphertext plus an element in $\mathbb{Z}_Q$.
- SP protocol: To compute $x_A'^{y_B} x_B'^{y_A}$, both parties need to do 2 encryptions, 1 decryption, 1 multiplication, and 1 exponentiation with exponent in Z_Q.

They exchange 4 messages, each consists of a ciphertext. In addition, each of $z_{A,1}z_{B,1}$ and $z_{A,2}z_{B,2}$ uses one invocation of $\mathsf{M2A}$ and one invocation of $\mathsf{A2M}$. Hence, the total communication consists of $4 + (2 + 2) \cdot 2 = 12$ ciphertexts and 4 field elements, and these takes $1 + (1 + 1) = 3$ rounds (since the computation of $z_{A,1}z_{B,1}$, $z_{A,2}z_{B,2}$ and Step 3-4 can be parallelized).
– EXP protocol (for $x \neq 0$ and $y \leq Q/2$): The total communication consists of $2(\mathsf{A2M}) + 2(\mathsf{ModRed}) + 12(\mathsf{SP}) + 2(\mathsf{M2A}) = 18$ ciphertexts and 7 field elements, and these takes 5 rounds (running $\mathsf{A2M}$ and ModRed in parallel.)

For the general case (x might be 0, y might be greater than $Q/2$), we need one more equality test which costs 4 rounds and $12t$ secure multiplications with error probability 2^{-t} [19], and one more comparison protocol for implementing ModRed, which costs 15 rounds and $279\ell + 5$ secure multiplications [19]. We remark that a secure multiplication can be done using 2 invocations of $\mathsf{M2A}$.

For integer exponentiation, we can choose a big enough Q and embed the integers into $\mathbb{Z}_Q$ in which we do all the arithmetic computations. We choose Q to be a big enough *safe prime* and use homomorphic encryption to build efficient secure integer exponentiation as mentioned. However, for realizing modular exponentiation for a general Q, we are not aware of any candidate encryption schemes that can be used in our scalar product protocol. On the other hand, our protocol to be described in the next section works for a general Q.

4 Implementation Using Oblivious Transfer

We use the noisy encoding techniques proposed by Ishai, Prabhakaran and Sahai [11] to implement our new protocols $\mathsf{A2M}$ and SP in Figure 7 and 9. We also describe $\mathsf{M2A}$ from [11] in Figure 8 for completeness.

We use $OT(m_0, m_1; \sigma)_{A \to B}$ to denote the OT protocol jointly run by Alice, who holds two messages $m^{(0)}, m^{(1)} \in \mathbb{Z}_Q$, and Bob, who holds the selection bit $\sigma \in \{0, 1\}$. Informally, security requires that after the protocol, Alice cannot learn σ, while Bob can only obtain $m^{(\sigma)}$, and receives no further information of $m^{(1-\sigma)}$. Similarly, let $\boldsymbol{v}$ be any vector and $\boldsymbol{v}_i$ be its i-th element. We use $OT(\boldsymbol{m}^{(0)}, \boldsymbol{m}^{(1)}; \boldsymbol{\sigma})_{A \to B}$ to denote the "vector version" of an OT protocol jointly run by Alice, who holds two vectors of messages $\boldsymbol{m}^{(0)}, \boldsymbol{m}^{(1)} \in (\mathbb{Z}_Q)^n$, and Bob, who holds the selection vector $\boldsymbol{\sigma} \in \{0, 1\}^n$ and wants to learn $\boldsymbol{m}_i^{(\sigma_i)}$ for $i \in [1, n]$.

Noisy Encoding. We review the noisy encoding scheme of Ishai, Prabhakaran and Sahai [11], which we are going to use as a building block. Encoding of $x \in \mathbb{Z}_Q$, denoted as $\mathsf{NoisyEnc}_n^{\mathbb{Z}_Q}(x)$ where n is a parameter of the encoding and $\mathbb{Z}_Q$ is the underlying field/ring, is computed by the following randomized procedure:

1. Pick a random bit-vector $\boldsymbol{\sigma} \leftarrow \{0, 1\}^n$.
2. Pick a random vector $\boldsymbol{u} \in (\mathbb{Z}_Q)^n$ conditioned on $\sum_{i=1}^n \boldsymbol{u}_i = x$.
3. Pick a random vector pair $(\boldsymbol{v}^0, \boldsymbol{v}^1) \in ((\mathbb{Z}_Q)^n)^2$ conditioned on $\forall i, \boldsymbol{v}_i^{\boldsymbol{\sigma}_i} = \boldsymbol{u}_i$.
4. Output $(\boldsymbol{v}^0, \boldsymbol{v}^1, \boldsymbol{\sigma})$.

The encoding contains two parts: $(\boldsymbol{v}^0, \boldsymbol{v}^1)$ and $\boldsymbol{\sigma}$. It has been proven [11] that with sufficiently large n, the distribution of $(\boldsymbol{v}^0, \boldsymbol{v}^1)$ (without $\boldsymbol{\sigma}$) statistically hides x; while one can decode $(\boldsymbol{v}^0, \boldsymbol{v}^1)$ with $\boldsymbol{\sigma}$ to retrieve x.

Our Implementations. At a high level, in protocols A2M and M2A (Figure 7, 8), Bob computes a noisy encoding $(\boldsymbol{u}^0, \boldsymbol{u}^1, \boldsymbol{\sigma})$ of his input x_B and sends $(\boldsymbol{u}^0, \boldsymbol{u}^1)$ to Alice, which contains the information of x_B but statistically hides it from Alice. Alice can still compute another "re-randomized" encoding of $x_A + x_B$ or $x_A \cdot x_B$ from $(\boldsymbol{u}^0, \boldsymbol{u}^1)$ to protect her privacy, and use OT to let Bob learn the messages according to $\boldsymbol{\sigma}$. This is similar to our solution based on the homomorphic encryption. In protocol SP(Figure 9), both parties need to compute the noisy encoding of their inputs y_A, y_B and send the information to each other since they want the other party to compute $x_B^{y_A}$ and $x_A^{y_B}$ respectively. Similarly, they both do re-randomization and use OT to send the messages back.

A2M Protocol $(z_A, z_B) \leftarrow (A(x_A), B(x_B))_{\mathsf{A2M}}$

1. Bob computes $(\boldsymbol{u}^0, \boldsymbol{u}^1, \boldsymbol{\sigma}) \leftarrow \mathsf{NoisyEnc}_n^{\mathbb{Z}_Q}(x_B)$, and sends $(\boldsymbol{u}^0, \boldsymbol{u}^1)$ to Alice.
2. Alice does the following:
   - Pick a random $p \leftarrow \mathbb{Z}_Q^*$.
   - Pick a random vector $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)$ conditioned on $x_A = \sum_{i=1}^n t_i$.
   - Compute $\boldsymbol{w}^0 = p \cdot (\boldsymbol{u}^0 + \boldsymbol{t})$ and $\boldsymbol{w}^1 = p \cdot (\boldsymbol{u}^1 + \boldsymbol{t})$.
   - Send to Bob with $OT((\boldsymbol{w}^0, \boldsymbol{w}^1); \boldsymbol{\sigma})_{A \to B}$.
   - Output $z_A = p^{-1}$.
3. Bob outputs $z_B = \sum_{i=1}^n \boldsymbol{w}_i^{\boldsymbol{\sigma}_i}$.

Fig. 7. Implementation of A2M using OT

Efficiency. We consider the running time of our protocol and the communication complexity in terms of the number of bits and the number of messages being sent. In protocol A2M and M2A, the operations we perform are: (1) multiplication, (2) addition, (3) sampling a random element and (4) oblivious transfer. Among all these operations, multiplication and oblivious transfer are the most expensive, so when we measure the efficiency, we count the number of these two operations. In protocol SP, we need another operation – exponentiation in $\mathbb{Z}_Q$, which is much more expensive than multiplication. Thus, we count the number of exponentiations and OTs in protocol SP. In the vector version of OT, the sender sends $2n$ elements in $\mathbb{Z}_Q$ and the receiver learns n of them. We consider this as n operations of basic OT in which the sender sends two elements in $\mathbb{Z}_Q$ and the receiver learns one of them. Under this measurement standard:

- M2A and A2M protocols: Alice performs $2n$ multiplications and n OTs. Bob only performs additions and sampling random elements. The protocol exchanges 2 messages: one with $2n$ elements in $\mathbb{Z}_Q$, and the other with n OTs.

M2A Protocol [11] $(z_A, z_B) \leftarrow (A(x_A), B(x_B))_{\mathsf{M2A}}$

1. Bob computes $(\boldsymbol{u}^0, \boldsymbol{u}^1, \boldsymbol{\sigma}) \leftarrow \mathsf{NoisyEnc}_n^{\mathbb{Z}_Q}(x_B)$, and sends $(\boldsymbol{u}^0, \boldsymbol{u}^1)$ to Alice.
2. Alice does the following:
   - Pick a random $\boldsymbol{t} \leftarrow (\mathbb{Z}_Q)^n$.
   - Compute $\boldsymbol{w}^0 = \boldsymbol{u}^0 \cdot x_A + \boldsymbol{t}$ and $\boldsymbol{w}^1 = \boldsymbol{u}^1 \cdot x_A + \boldsymbol{t}$.
   - Send to Bob with $OT((\boldsymbol{w}^0, \boldsymbol{w}^1); \boldsymbol{\sigma})_{A \to B}$.
   - Output $z_A = -\sum_{i=1}^n \boldsymbol{t}_i$.
3. Bob outputs $z_B = \sum_{i=1}^n \boldsymbol{w}_i^{\boldsymbol{\sigma}_i}$.

Fig. 8. Implementation of M2A using OT

SP Protocol $(z_A, z_B) \leftarrow (A(x_A, y_A), B(x_B, y_B))_{\mathsf{SP}}$

1. Alice computes $(\boldsymbol{u}_A^0, \boldsymbol{u}_A^1, \boldsymbol{\sigma}_A) \leftarrow \mathsf{NoisyEnc}_n^{\mathbb{Z}_Q}(y_A)$, and sends $(\boldsymbol{u}_A^0, \boldsymbol{u}_A^1)$ to Bob.
2. Bob computes $(\boldsymbol{u}_B^0, \boldsymbol{u}_B^1, \boldsymbol{\sigma}_B) \leftarrow \mathsf{NoisyEnc}_n^{\mathbb{Z}_Q}(y_B)$, and sends $(\boldsymbol{u}_B^0, \boldsymbol{u}_B^1)$ to Alice.
3. Alice picks a random $\boldsymbol{t}_A \leftarrow (\mathbb{Z}_Q^*)^n$, computes $\boldsymbol{w}_{A,i}^0 = x_A^{\boldsymbol{u}_{B,i}^0} \cdot \boldsymbol{t}_{A,i}$, $\boldsymbol{w}_{A,i}^1 = x_A^{\boldsymbol{u}_{B,i}^1} \cdot \boldsymbol{t}_{A,i}$, and sends to Bob with $OT((\boldsymbol{w}_A^0, \boldsymbol{w}_A^1); \boldsymbol{\sigma}_B)_{A \to B}$.
4. Bob picks a random $\boldsymbol{t}_B \leftarrow (\mathbb{Z}_Q^*)^n$. compute $\boldsymbol{w}_{B,i}^0 = x_B^{\boldsymbol{u}_{A,i}^0} \cdot \boldsymbol{t}_{B,i}$, $\boldsymbol{w}_{B,i}^1 = x_B^{\boldsymbol{u}_{A,i}^1} \cdot \boldsymbol{t}_{B,i}$. and sends to Alice with $OT((\boldsymbol{w}_B^0, \boldsymbol{w}_B^1); \boldsymbol{\sigma}_A)_{B \to A}$.
5. Alice outputs $z_A = \prod_{i=1}^n \boldsymbol{w}_{B,i}^{\boldsymbol{\sigma}_{A,i}} / \prod_{i=1}^n \boldsymbol{t}_{A,i}$.
6. Bob outputs $z_B = \prod_{i=1}^n \boldsymbol{w}_{A,i}^{\boldsymbol{\sigma}_{B,i}} / \prod_{i=1}^n \boldsymbol{t}_{B,i}$.

Fig. 9. Implementation of SP using OT

- SP protocol: Both parties perform $2n$ exponentiations and n OTs, involving 4 message exchanges: two with $2n$ elements in $\mathbb{Z}_Q$, and two with n OTs.
- EXP protocol (for $x \neq 0$ and $y \leq Q/2$): The protocol consists of the four protocols (also protocol ModRed, which requires 1 call of M2A). With parallelization, the round complexity just needs 5 message exchanges. The total communication complexity is $10n$ elements in $\mathbb{Z}_Q$, and $5n$ OTs.

For the parameter setting, we need to set $n = O(k + \log Q)$ to achieve a $2^{-\Omega(k)}$-security, where it is sufficient to take the hidden constant as 3. For online OT, we need to send 2 elements in $\mathbb{Z}_Q$ per OT, and thus the communication complexity is $10n + 2 \cdot 5n = 20n = 60k + 60 \log Q$ elements in $\mathbb{Z}_Q$. Due to the lack of space, security analysis of our protocols are deferred to the full version of this paper.

For the general case (x might be 0, and y might be greater than $Q/2$), as before, we need one more equality test in the beginning, which costs 4 rounds and $12t$ secure multiplications with error probability 2^{-t} [19], and one more comparison protocol for implementing ModRed, which costs 15 rounds and $279\ell +$ 5 secure multiplications [19]. Recall that we can perform a secure multiplication using 2 invocations of the M2A protocol.

5 Conclusion and Future Directions

In this paper, we propose a new framework to efficiently compute exponentiation when both the base and the exponent are shared among different parties. The goal is to build constant-round protocols with communication cost comparable to that of a secure multiplication. Instead of using bit-decomposition, we utilize the inter-conversion of additive and multiplicative sharing and "scalar product" over an "exponential module" (A2M, M2A, and SP).

We implemented A2M, M2A and SP in two ways – homomorphic encryption, and oblivious transfer (OT). We use both an additive homomorphic encryption and a multiplicative homomorphic encryption (but not a fully homomorphic one) to achieve efficiency. Our OT-based solution uses the noisy encoding techniques [11] and provides versatility of the underlying assumption. We remark that these protocols support precomputation for higher online performance.

The extension of the work runs through several directions. First, our framework of two-party exponentiation implies a framework of multiparty exponentiation without honest majority. So a goal is to devise efficient protocols for the underlying A2M and M2A for the multiparty setting. Second, to duel with malicious adversaries without loss of efficiency, instead of general knowledge proofs, a specific arithmetic proof is required. Furthermore, it is worth investigating the multiparty exponentiation in a general adversary setting. Finally, our solution for a general modulus requires one invocation of a secure comparison. Hence, a cheap secure comparison protocol, comparable to the cost of a secure multiplication protocol, would be a key to improve efficiency further.

References

1. Yao, A.C.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 - A Completeness Theorem for Protocols with Honest Majority. In: Proc. 19th STOC, pp. 218–229 (1987)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In: Proc. 20th STOC, pp. 1–10 (1988)
4. Chaum, D., Crepéau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Proc. 20th STOC, pp. 11–19 (1988)
5. Lindell, Y., Pinkas, B.: Privacy-Preserving Data Mining. Journal of the Cryptology 15(3), 177–206 (2002)
6. Bunn, P., Ostrovsky, R.: Secure Two-Party k-Means Clustering. In: Proc. 14th CCS, pp. 486–497 (2007)
7. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T.P., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M.I., Toft, T.: Secure Multiparty Computation Goes Live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
8. Lindell, Y., Pinkas, B.: Secure Multiparty Computation for Privacy-Preserving Data Mining. Journal of the ACM 1(1), 59–98 (2009)

9. Damgård, I., Mikkelsen, G.L.: Efficient Robust and Constant-Round Distributed RSA Key Generation. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 183–200. Springer, Heidelberg (2010)
10. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer - Efficiency. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
11. 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)
12. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006)
13. Cleve, R.: Limits on the Security of Coin Flips when Half the Processors are Faulty. In: Proc. 18th STOC, pp. 364–369 (1986)
14. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: Proc. 41st STOC, pp. 169–178 (2009)
15. 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)
16. Algesheimer, J., Camenisch, J., Shoup, V.: Efficient Computation Modulo a Shared Secret with Application to the Generation of Shared Safe-Prime Products. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 417–432. Springer, Heidelberg (2002)
17. Kiltz, E., Leander, G., Malone-Lee, J.: Secure computation of the mean and related statistics. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 283–302. Springer, Heidelberg (2005)
18. Damgård, I., Thorbek, R.: Linear Integer Secret Sharing and Distributed Exponentiation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 75–90. Springer, Heidelberg (2006)
19. Nishide, T., Ohta, K.: Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 343–360. Springer, Heidelberg (2007)
20. Beaver, D.: Commodity-based Cryptography (Extended Abstract). In: Proc. 29th STOC, pp. 446–455 (1997)
21. El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Goos, G., Hartmanis, J. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
22. 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)

A General, Flexible and Efficient Proof of Inclusion and Exclusion

Kun Peng

Institute for Infocomm Research, Singapore
dr.kun.peng@gmail.com

Abstract. Inclusion proof shows that a secret committed message is in a finite group of messages, while exclusion proof shows that a secret committed message is not in a finite group of messages. A general, flexible and efficient solution to inclusion proof and exclusion proof is proposed in this paper. It overcomes the drawbacks of the existing solutions to inclusion proof and exclusion proof. It achieves all the desired security properties in inclusion proof and exclusion proof. It is the most efficient general solution to inclusion proof and exclusion proof and only costs $O(\sqrt{n})$ for any inclusion proof and exclusion proof regarding any finite group of n messages.

1 Introduction

In cryptographic secure protocols, sometimes a party chooses a message from a finite set $S = \{s_1, s_2, \ldots, s_n\}$ and then commits to it. He keeps the message secret and publishes the commitment. He needs to prove that the message in the commitment is indeed in S, but cannot reveal the secret message. Such a proof is called inclusion proof in this paper. For example, in e-auction [18,20,21,25] and e-voting [19,22,23,24,26], very often a bidder or voter has to prove that his secret bid or vote is chosen from a list of candidates. As explained in [5], inclusion proof is also useful in applications like e-cash systems and anonymous credential systems. In some cryptographic applications, it is needed for a party to prove that a committed secret message m is not in a finite set $S = \{s_1, s_2, \ldots, s_n\}$ without revealing it. For example, as explained in [14], a financial institute may ask a loan applier to prove that he is not in a black list, while the applier does not want to reveal his identity before the application goes to next step. This proof is called nonmembership proof in [14] and called exclusion proof in this paper.

According to [10], any secret knowledge can be proved without revealing it if there is no critical requirement on efficiency. There are some general zero knowldge proof techniques [10,9,13,12], which handles various proofs including inclusion proof and exclusion proof by reducing them to a standard form and then giving an all-purpose proof. We are not very interested in those techniques as we focus on high efficiency. Obviously, proof techniques specially designed for inclusion proof and exclusion proof have an advantage in efficiency improvement of the two proofs over the general all-purposed proof techniques as the former does not need to consider any other proof. So we focus on proof techniques to handle only inclusion proof and exclusion proof in this paper.

Apart from the straightforward solution to inclusion proof through ZK (zero knowledge) proof of partial knowledge [7] and the brute-force solution to exclusion proof by proving that the committed integer is unequal to every integer in the set, there are several more efficient inclusion and exclusion proof schemes [3,14,5]. However, they have their drawbacks as will be detailed in Section 2. Inclusion proof in [3] is strictly limited by a few conditions and so lacks generality and flexibility. Exclusion proof in [14] is a variant of [3], so has the same drawback. Inclusion proof in [5] lacks public verifiability, must be interactive and is inefficient when there are many verifiers.

In this paper, new inclusion proof and new exclusion proof are proposed. They employ the same strategy: reducing a proof regarding a large set to multiple proofs regarding smaller sets and then reducing each proof regarding a smaller set to a proof regarding a single integer. In this way, a complex task is divided into multiple simpler tasks and high efficiency is achieved. After that a calculation-optimising method is designed to further improve efficiency. The new proof technique overcomes the drawbacks in [3,14,5] and are very efficient. It is more efficient than the existing general solutions to inclusion proof and exclusion proof including the straightforward simple solutions and [5], while [3,14] are special solutions strictly limited to special applications. When the size of S is n, it only costs $O(\sqrt{n})$ exponentiations in computation and transfers $O(\sqrt{n})$ integers in communication, no matter what messages are in S and committed.

2 Security Requirements and the Existing Solutions

The following security properties are usually desired in inclusion proof and exclusion proof.

- Completeness: in an inclusion proof protocol, if the committed integer is in the set and the prover strictly follows the inclusion proof protocol, he can pass the verification in the protocol; in an exclusion proof protocol, if the committed integer is not in the set and the prover strictly follows the exclusion proof protocol, he can pass the verification in the protocol.
- Soundness: in an inclusion proof protocol, if the committed integer is not in the set, the probability that the prover passes the verification in the protocol is negligible; in an exclusion proof protocol, if the committed integer is in the set, the probability that the prover passes the verification in the protocol is negligible.
- Zero knowledge: in an inclusion proof protocol, no information about the committed message is revealed except that it is in the set; in an exclusion proof protocol, no information about the committed message is revealed except that it is not in the set. More precisely, in both inclusion proof and exclusion proof, the proof transcript can be simulated without any difference by a party without any knowledge of any secret.

- Public verifiability: validity of all the operations can be publicly verified by any verifier and independent observer, in both inclusion proof and exclusion proof.
- Generality and flexibility: format of the committed integer and the set is not limited in any way. More precisely, in any application of inclusion proof or exclusion proof, just choose a large enough message space for the commitment algorithm to cover any possible committed integer and the set, then inclusion proof and exclusion proof can always work.
- Non-interaction: when necessary, inclusion proof and exclusion proof can be non-interactive.

The simplest solution to inclusion proof is ZK proof of partial knowledge [7], which proves that the committed message may be every message in the set one by one and then link the multiple proofs with OR logic. This solution is called simple inclusion proof in this paper. Similarly, exclusion proof can be implemented by proving that the committed message is unequal to each message in the set one by one and then linking the multiple proofs with AND logic. Inequality of two secret integers can be proved using techniques like ZK proof of inequality of discrete logarithm in [4]. This solution is called simple exclusion proof in this paper. The advantage of these two simple solutions is generality and versatility. They can prove inclusion and exclusion regarding any committed integer and any set. They can achieve all the desired security properties including public verifiability and flexibility. Their drawback is low efficiency. In communication, they have to to transfer $O(n)$ integers. In computation, they cost both the prover and the verifier $O(n)$ exponentiations.

A more efficient inclusion proof is proposed by Camenisch *et al.* [5]. In [5], a verifier signs every message in S using his own private key and sends all the signatures to the prover, who then proves that he knows the signature on the message in the commitment. In this method, the computational cost of a prover becomes constant and thus much more efficient although efficiency improvement in communication and on the verifier's side is not evident. This inclusion proof has several drawbacks. Its main drawback is lack of public verifiability. The signatures sent to the prover are not public. Except for the prover and the verifier generating them, the other parties including other verifiers do not know whether any signature of other messages is sent to the prover. So it is a two-party private proof between a prover and a certain verifier and it has to be separately and repeatedly run between the prover and every verifier. Therefore, when there are many verifiers, the overhead for the prover is very high. Moreover, Fiat-Shamir heuristic cannot be employed to achieve non-interaction and every verifier must interactively run the inclusion proof protocol with the prover. In addition, this proof technique cannot handle exclusion proof.

The most efficient inclusion proof is proposed by Camenisch *et al.* [3]. In [3] to show that a secret message committed in c is in S, knowledge of integers m and ϵ is proved such that m is committed in c and $\epsilon^m = g^{\prod_{i=1}^{n} s_i}$ where g is a generator of a cyclic multiplication group with a composite multiplication modulus difficult to factorize. Obviously, if $m = s_j$, the prover can use $\epsilon = g^{\prod_{i=1}^{j-1} s_i \prod_{i=j+1}^{n} s_i}$ to give the proof and pass the verification. The main drawback of this solution is lack of generality and flexibility. It is strictly limited by a few conditions. Firstly, the messages in the set must be positive prime integers in a certain interval range. Secondly, the committed message must be proved to be in the interval range to guarantee that the prover does not commit to the product of some integers in the set. This limitation implies that additional range proof is needed. Thirdly, a co-called strong RSA assumption is necessary for security of the inclusion proof in [3]. Apart from depending on an unusual computational hard problem, the assumption implies that the set must be chosen independent of the prover so that it appears random to him. Application of [3] to inclusion is so strictly limited that its own author Camenisch only suggests to use it in special applications like anonymous credential. For general purpose inclusion proof, Camenisch *et al.* later propose the inclusion proof technique in [5], which we have discussed.

The inclusion proof technique in [3] is extended to exclusion proof by Li *et al.* [14]. The key technique in [14] is an accumulator-based proof system, which can provide a witness for each integer in a special set but not in S to show its exclusion from S. It is more efficient than the simple exclusion proof, but like the inclusion proof technique in [3] it is strictly limited in application. It is subject to three conditions. Firstly, all the messages in S and the committed message must be prime integers. Secondly, all the messages in S and the committed message must be non-negative integers smaller than 2^{ι} where ι is a security parameter denoted as l in [14]. Thirdly, a necessary condition satisfied in [3] is ignored in [14]: no integer in the set can be larger than the product of any other integers in the set. Moreover, dependence on the strong RSA assumption implies another condition in [14]: the set must be chosen independent of the prover so that it appears random to him.

Although mapping all the the messages in the set and all the messages possible to commit to into the special supported set may improve applicability of [3] and [14], this method does not always work simply and effectively. Instead, its applicability and complexity depend on the application environment as explained in the following.

– Any two different messages in the set and out of the set respectively cannot share the same image in the mapping so that the mapping always distinguishes the messages in the set and the messages out of the set. Moreover, sometimes the committed message will be recovered and used later. So the mapping function needs to be invertible and some simple functions (like mapping an integer to the prime nearest to it) cannot work.
– Some invertible mapping functions need a large memory to store, especially when the message space is large.
– In some applications the committed message must be processed in the form of commitment (e.g. in multi-party secure computation or e-voting where the commitment function is in the form of an encryption algorithm). Such applications usually exploit homomorphism of the commitment algorithm to implement computation of commitments, so the original messages in them cannot be changed in any way.

There are some even more special proof schemes [1,15,11], which prove that a secret committed integer lies in a finite interval range. They are the so called “range proof” schemes and are incomparable to our work. Moroever, as stated in Section 1, unpublished and rejected proposals with problems and limitations are incomparable to our work.

3 New Inclusion Proof and Exclusion Proof

The main idea of the new design is to divide the set S into multiple subsets, so that inclusion of a message in S is reduced to its inclusion in one of the subsets and exclusion of a message from S is reduced to its exclusion from all of the subsets. In this way, an inclusion proof or exclusion proof is reduced to multiple inclusion proofs or multiple exclusion proofs in a smaller scale. Then each smaller-scale inclusion proof is reduced to proof of commitment and each smaller-scale exclusion proof is reduced to proof of uncommitment where the former proves that a message is committed in a commitment and the latter proves that a message is not committed in a commitment. To be consistent with the existing inclusion proof and exclusion proof schemes and make a fair comparison, the following commitment function is employed.

- p and q are large primes such that $q|p-1$ and $q > s_i$ for $i = 1, 2, \ldots, n$. G is the cyclic subgroup with order q of Z_p^*. Integers g and h are generators of G such that $\log_g h$ is unknown.
- From now on in this paper, all the computations involving the integers in any matrix and vector is carried out modulo q.
- A prover randomly chooses r from Z_q and commits to a secret integer m in $c = g^m h^r \bmod p$.

3.1 Reducing Inclusion Proof and Exclusion Proof to Simpler Proofs

The simplifying reduction from inclusion proof and exclusion proof to commitment proof and uncommitment proof is as follows.

1. For simplicity of description, suppose S can be divided into t subsets $S_1, S_2, \ldots, S_t$ and each S_l contains k integers $s_{l,1}, s_{l,2}, \ldots, s_{l,k}$.
2. The prover randomly chooses an integer s in Z_q and calculates for each S_l integers $b_{l,i}$ for $i = 1, 2, \ldots, k$ in Z_q to satisfy

$$\textstyle\sum_{i=1}^{k} b_{l,i} s_{l,\rho}^{i} = s \bmod q \text{ for } \rho = 1, 2, \ldots, k. \quad (1)$$

   More precisely, integers $b_{l,i}$ for $l = 1, 2, \ldots, t$ and $i = 1, 2, \ldots, k$ must satisfy

$$\begin{pmatrix} s_{l,1} & s_{l,1}^2 & \cdots & s_{l,1}^k \\ s_{l,2} & s_{l,2}^2 & \cdots & s_{l,2}^k \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ s_{l,k} & s_{l,k}^2 & \cdots & s_{l,k}^k \end{pmatrix} \begin{pmatrix} b_{l,1} \\ b_{l,2} \\ \cdots \\ \cdots \\ b_{l,k} \end{pmatrix} = \begin{pmatrix} s \\ s \\ \cdots \\ \cdots \\ s \end{pmatrix}$$

for $l = 1, 2, \dots, t$. As $s_{l,i} < q$ for $l = 1, 2, \dots, t$ and $i = 1, 2, \dots, k$ and they are different integers,

$$M_l = \begin{pmatrix} s_{l,1} & s_{l,1}^2 & \dots & s_{l,1}^k \\ s_{l,2} & s_{l,2}^2 & \dots & s_{l,2}^k \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ s_{l,k} & s_{l,k}^2 & \dots & s_{l,k}^k \end{pmatrix}$$

is a non-singular matrix for $l = 1, 2, \dots, t$ and there is a unique solution for $b_{l,1}, b_{l,2}, \dots, b_{l,k}$:

$$\begin{pmatrix} b_{l,1} \\ b_{l,2} \\ \dots \\ \dots \\ b_{l,k} \end{pmatrix} = M_l^{-1} \begin{pmatrix} s \\ s \\ \dots \\ \dots \\ s \end{pmatrix}$$

for $l = 1, 2, \dots, t$. Therefore, functions $F_l(x) = \sum_{i=1}^{k} b_{l,i} x^i \bmod q$ for $l = 1, 2, \dots, t$ are obtained, each to satisfy

$$F_l(s_{l,i}) = s \text{ for } i = 1, 2, \dots, k. \qquad (2)$$

The prover publishes s. Note that $F_l()$ is actually the unique polynomial with degree at most k to satisfy (2) and $F_l(0) = 0$. Readers with basic knowledge in linear algebra should know a few efficient methods, which do not cost any exponentiation, to calculate $F_l()$ from $s_{l,i}$ for $i = 1, 2, \dots, k$. Our presentation of $F_l()$ through matrix calculations is only one of them, which seems formal and straightforward. Also note that if necessary calculation of $F_l()$ can be performed beforehand once S is published such that it is already available when the inclusion proof or exclusion proof starts.

3. The prover calculates $e_i = e_{i-1}^m h^{\gamma_i} \bmod p$ for $i = 1, 2, \dots, k-1$ where $e_0 = c$ and γ_i is randomly chosen from Z_q. The prover proves validity of $e_1, e_2, \dots, e_{k-1}$ using a zero knowledge proof that he knows m, r and γ_i for $i = 1, 2, \dots, k-1$ such that $c = g^m h^r \bmod p$ and $e_i = e_{i-1}^m h^{\gamma_i} \bmod p$ for $i = 1, 2, \dots, k-1$, which can be implemented through a simple combination of ZK proof of knowledge of discrete logarithm [27] and ZK proof of equality of discrete logarithms [6].
4. A verifier
   (a) calculates $b_{l,i}$ for $l = 1, 2, \dots, t$ and $i = 1, 2, \dots, k$ to satisfy (1) like the prover does where s is provided by the prover;
   (b) verifies the prover's proof of validity of $e_1, e_2, \dots, e_{k-1}$.

   He accepts the reduction iff the prover's proof is passed and $e_1, e_2, \dots, e_{k-1}$ are valid.

The operations above have reduced inclusion proof and exclusion proof to commitment proof and uncommitment proof respectively. More precisely,

– Inclusion of m in S is reduced to inclusion of m in S_1 or S_2 or ...... or S_t. As $s = F_l(m)$ if $m \in S_l$, inclusion of m in S_l is reduced to commitment of s in ω_l where

$$\omega_l = C(F_l(m)) = C(\sum_{i=1}^{k} b_{l,i}x^i) = \prod_{i=0}^{k-1} e_i^{b_{l,i+1}} \bmod p.$$

and $C()$ denotes the commitment function to commit a message m' in $C(m') = g^{m'}h^{\eta} \bmod p$ using a random integer η in Z_q.

– Exclusion of m from S is reduced to exclusion of m from S_1 and S_2 and ...... and S_t, while exclusion of m from S_l is reduced to uncommitment of s from ω_l.

3.2 Specification of the Two Simpler Proofs

The reduction work above is the same for inclusion proof and exclusion proof. After that, the left work is different for inclusion proof and exclusion proof. In an inclusion proof, the prover has to prove that s is committed to by him in ω_1 or ω_2 or ...... or ω_t. More precisely, he has to prove that he knows $\log_h \omega_1/g^s$ or $\log_h \omega_2/g^s$ or ...... or $\log_h \omega_t/g^s$ as follows.

1. ω_l can be publicly calculated by any verifier in the form

$$\omega_l = \prod_{i=0}^{k-1} e_i^{b_{l,i+1}} \bmod p.$$

2. If needed the prover himself can secretly calculate ω_l/g^s more efficiently:

$$\omega_l/g^s = \begin{cases} h^{\sum_{i=0}^{k-1} b_{l,i+1}\Gamma_{i+1}} \bmod p & \text{if } m \in S_l \\ g^{(\sum_{i=0}^{k-1} b_{l,i+1}m^{i+1})-s}h^{\sum_{i=0}^{k-1} b_{l,i+1}\Gamma_{i+1}} \bmod p & \text{if } m \notin S_l \end{cases}$$

where $\Gamma_i = m\Gamma_{i-1} + \gamma_{i-1} \bmod q$ for $i = 2, 3, \ldots, k$, $\Gamma_1 = r$ and $m^2, m^3, \ldots, m^k$ can be calculated using $k-1$ multiplications and reused in calculation of $\omega_1, \omega_2, \ldots, \omega_t$.

3. The prover runs ZK proof of partial knowledge [7] to implement the proof that he knows one of t discrete logarithms $\log_h \omega_1/g^s$, $\log_h \omega_2/g^s$, ..., $\log_h \omega_t/g^s$.
4. Any verifier can publicly verify the prover's proof of knowledge of one of t discrete logarithms. He accepts the inclusion proof iff the prover's proof is successfully verified.

In an exclusion proof, the prover has to prove s is not committed in any of ω_1, ω_2, ..., ω_t. Proof that s is not committed in ω_l is as follows where the prover and the verifier can calculate ω_l respectively like in the inclusion proof and the prover knows $M_l = F_l(m) = \sum_{i=1}^{k} b_{l,i}m^i \bmod q$, which is committed in ω_l.

1. The prover randomly chooses a positive integer T in Z_q and publishes $y = g^{T(s-M_l)} \bmod p$.

2. He proves knowledge of secret integer $x = T(s - M_l)$ such that $y = g^x \bmod p$ using zero knowledge proof of knowledge of discrete logarithm [27].
3. He proves knowledge of secret integers T and r' such that $(g^s)^T h^{r'} = \omega_l^T y \bmod p$ where $r' = T \sum_{i=1}^{k} (b_{l,i} \Gamma_i) \bmod q$, $\Gamma_i = m\Gamma_{i-1} + \gamma_{i-1} \bmod q$ for $i = 2, 3, \ldots, k$ and $\Gamma_1 = r$ using zero knowledge proof of knowledge of discrete logarithm [27] and knowledge proof of equality of discrete logarithms [6].
4. Any verifier can verify $y > 1$ and the two zero knowledge proofs. He accepts the uncommitment claim if and only if all the three conditions are satisfied in his check.

This proof is called uncommitment proof. The prover repeats it for each l in $\{1, 2, \ldots, t\}$ and any verifier can verify the prover's proof. The verifier accepts the exclusion proof iff the all the t instances of proof are successfully verified. Note that $m^2, m^3, \ldots, m^k$ can be calculated using $k - 1$ multiplications and reused in calculation of $M_1, M_2, \ldots, M_t$ by the prover.

4 Security Analysis

Completeness of the new inclusion proof and exclusion proof is obvious. Any reader can follow the running of the two proof protocols step by step to verify that an honest prover can strictly follow them to pass their verifications. If the challenges in the employed zero knowledge proof primitives are generated by a pseudo-random function, no interactive verifier is needed and the new inclusion proof and exclusion proof can be non-interactive in the random oracle model. Moreover, public verifiability is achieved in the two proofs as every detail of them can be publicly verified by any one. Other security properties of them are proved in Theorems 1, 2 and 3.

Theorem 1. *Both the new inclusion proof protocol and the new exclusion proof protocol achieve honest-verifier zero knowledge.*

Proof: Both the new inclusion proof protocol and the new exclusion proof protocol only employ three zero knowledge proof primitives: zero knowledge proof of knowledge of discrete logarithm [27], zero knowledge proof of equality of discrete logarithms [6] and zero knowledge proof of partial knowledge [7]. Honest-verifier zero knowledge of these three proof primitives is formally proved when they are proposed. More precisely, the proof transcripts of the three primitives with an honest verifier can be simulated without any difference by a party without any secret knowledge.

Besides the three zero knowledge proof primitives, the two proofs only reveal s, e_1, $e_2, \ldots,$ e_{k-1}. As s is randomly chosen from Z_q, the distribution of s is uniform in Z_q. As $e_i = e_{i-1}^m h^{\gamma_i} \bmod p$ for $i = 1, 2, \ldots, k - 1$ and γ_i is randomly chosen from Z_q, each e_i is uniformly distributed in G. So anybody can simulate s, e_1, $e_2, \ldots,$ e_{k-1} without any difference by randomly choosing s in Z_q and every e_i in G.

Other integers used in the proof like $b_{l,i}$ and ω_l are deterministic public functions of $s_1,\ s_2,\ \dots, s_n,\ s,\ c,\ e_1,\ e_2, \dots,\ e_{k-1}$. So they are not independent variables affecting zero knowledge of the two proof primitives.

Since the whole proof transcripts of the two proof protocols with an honest verifier can be simulated without any difference by a party without any secret knowledge, they achieve honest-verifier zero knowledge. □

Theorem 2. *The new inclusion proof is sound. More precisely, if a polynomial prover can extract an opening* (m, r) *of* c *such that* $m \neq s_i \bmod q$ *for* $i = 1, 2, \dots, n$*, then the probability that the prover can pass the verification in the new inclusion proof is negligible.*

Proof: If the prover extracts m, r and passes the verification in the new inclusion proof with a non-negligible probability while $c = g^m h^r \bmod p$ and $m \neq s_i \bmod q$ for $i = 1, 2, \dots, n$, a contradiction can be found as follows. As he passes the verification in the new inclusion proof with a non-negligible probability, he must have successfully proved validity of $e_1, e_2, \dots, e_{k-1}$ with a non-negligible probability. As proof of validity of $e_1, e_2, \dots, e_{k-1}$ is based on proof of knowledge of discrete logarithm in [27] and proof of equality of discrete logarithms in [6], whose soundness is formally proved when they are proposed, it is guaranteed with a non-negligible probability that the prover can calculate integers m, r and γ_i for $i = 1, 2, \dots, k-1$ in polynomial time such that

$$c = g^m h^r \bmod p \tag{3}$$

$$e_i = e_{i-1}^m h^{\gamma_i} \bmod p \text{ for } i = 1, 2, \dots, k-1 \tag{4}$$

where $e_0 = c$.

As he passes the verification in the new inclusion proof with a non-negligible probability, the prover also must have successfully passed the zero knowledge proof of knowledge of one out of t discrete logarithms [7] with a non-negligible probability. As soundness of zero knowledge proof of partial knowledge [7] is formally proved when it is proposed, it is guaranteed that for some l in $\{1, 2, \dots, t\}$ the prover can calculate integers s and R in polynomial time such that

$$g^s h^R = \prod_{i=0}^{k-1} e_i^{b_{l,i+1}} \bmod p \tag{5}$$

with a non-negligible probability where $e_0 = c$.

(3), (4) and (5) imply that the prover can calculate integers s, R, $\sum_{i=0}^{k-1} b_{l,i+1} m^{i+1}$ and $\sum_{i=0}^{k-1} b_{l,i+1} \Gamma_{i+1}$ in polynomial time with a non-negligible probability such that

$$g^s h^R = \prod_{i=0}^{k-1} g^{b_{l,i+1} m^{i+1}} h^{b_{l,i+1} \Gamma_{i+1}} = g^{\sum_{i=0}^{k-1} b_{l,i+1} m^{i+1}} h^{\sum_{i=0}^{k-1} b_{l,i+1} \Gamma_{i+1}} \bmod p$$

where

$$\Gamma_i = m\Gamma_{i-1} + \gamma_{i-1} \bmod q \text{ for } i = 2, 3, \dots, k$$

and $\Gamma_1 = r$. So

$$s = \sum_{i=0}^{k-1} b_{l,i+1} m^{i+1} = \sum_{i=1}^{k} b_{l,i} m^i \bmod q$$

with a non-negligible probability. Otherwise, with a non-negligible probability the prover can calculate non-zero (modulo q) integers $\alpha = s - \sum_{i=0}^{k-1} b_{l,i+1} m^{i+1}$ and $\beta = R - \sum_{i=0}^{k-1} b_{l,i+1} \Gamma_{i+1}$ in polynomial time to satisfy $g^\alpha h^\beta = 1$ and thus can calculate $\log_g h$ in polynomial time, which is a contradiction.

Note that $b_{l,1}, b_{l,2}, \ldots, b_{l,k}$ are generated through

$$\textstyle\sum_{i=1}^{k} b_{l,i} s_{l,\rho}^{i} = s \bmod q \text{ for } \rho = 1, 2, \ldots, k.$$

So with a non-negligible probability

$$\begin{pmatrix} s_{l,1} & s_{l,1}^2 & \ldots & s_{l,1}^k \\ s_{l,2} & s_{l,2}^2 & \ldots & s_{l,2}^k \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ s_{l,k} & s_{l,k}^2 & \ldots & s_{l,k}^k \\ m & m^2 & \ldots, & m^k \end{pmatrix} \begin{pmatrix} b_{l,1} \\ b_{l,2} \\ \ldots \\ \ldots \\ b_{l,k} \end{pmatrix} = \begin{pmatrix} s \\ s \\ \ldots \\ \ldots \\ s \\ s \end{pmatrix} \tag{6}$$

However, as $m \neq s_i \bmod q$ for $i = 1, 2, \ldots, n$ and all the calculations in the matrix is performed modulo q, $\begin{pmatrix} s_{l,1} & s_{l,1}^2 & \ldots & s_{l,1}^k & s \\ s_{l,2} & s_{l,2}^2 & \ldots & s_{l,2}^k & s \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ s_{l,k} & s_{l,k}^2 & \ldots & s_{l,k}^k & s \\ m & m^2 & \ldots, & m^k & s \end{pmatrix}$ is a non-singular matrix and thus (6) absolutely and always fails. Therefore, a contradiction is found and the probability that a prover can pass the new inclusion proof is negligible if the integer he commits to in c is not in S. □

Theorem 3. *The new exclusion proof is sound and the probability that a prover can pass its verification is negligible if he can extract an opening (m, r) of c such that $m \in S$.*

Before Theorem 3 is proved, a lemma is proved first.

Lemma 1. *The uncommitment proof is sound. More precisely, if the prover passes its verification, then with an overwhelmingly large probability $s \neq M_l$.*

Proof: Note that the uncommitment proof is a simple combination of two instances of proof of knowledge of discrete logarithm [27] and one instance of proof of equality of discrete logarithms [6], whose soundness is formally proved when they are proposed. So it is guaranteed with an overwhelmingly large probability that the prover can calculate secret integers x, T and r' in polynomial time to satisfy

$$y = g^x \bmod p$$
$$(g^s)^T h^{r'} = \omega_l^T y \bmod p.$$

So with an overwhelmingly large probability

$$(g^s)^T h^{r'} = \omega_l^T g^x \bmod p. \tag{7}$$

As M_l is the message the prover commits to in ω_l, the prover can calculate integers M_l and R in polynomial time such that

$$\omega_l = g^{M_l} h^R \bmod p$$

and thus (7) implies that with an overwhelmingly large probability the prover can calculate x, T, r', M_l and R in polynomial time such that

$$(g^s)^T h^{r'} = (g^{M_l} h^R)^T g^x \bmod p.$$

So with an overwhelmingly large probability the prover can calculate $T(s-M_l)-x$ and $r'-TR$ in polynomial time such that

$$g^{T(s-M_l)-x} h^{r'-TR} = 1 \bmod p.$$

So with an overwhelmingly large probability

$$T(s-M_l)-x = 0 \bmod q$$

Otherwise, with an overwhelmingly large probability the prover can calculate $\log_g h = (TR-r')/(T(s-M_l)-x) \bmod q$ in polynomial time, which is a contradiction. As $y > 1$, $x \neq 0 \bmod q$ and so with an overwhelmingly large probability $s - M_l \neq 0 \bmod q$. Therefore, with an overwhelmingly large probability $s \neq M_l \bmod q$. □

Proof of Theorem 3: If the prover passes the verification in the new exclusion proof with a non-negligible probability while $m \in S$ and $c = g^m h^r \bmod p$, a contradiction can be found as follows. As the prover passes the verification in the new exclusion proof with a non-negligible probability, he must have successfully proved validity of $e_1, e_2, \ldots, e_{k-1}$ with a non-negligible probability. As proof of validity of $e_1, e_2, \ldots, e_{k-1}$ is based on proof of knowledge of discrete logarithm in [27] and proof of equality of discrete logarithms in [6], whose soundness is formally proved when they are proposed, it is guaranteed with a non-negligible probability that the prover can calculate integers m, r and γ_i for $i = 1, 2, \ldots, k-1$ in polynomial time such that

$$c = g^m h^r \bmod p \tag{8}$$

$$e_i = e_{i-1}^m h^{\gamma_i} \bmod p \text{ for } i = 1, 2, \ldots, k-1 \tag{9}$$

where $e_0 = c$.

(8) and (9) imply that with a non-negligible probability

$$\begin{aligned}\prod_{i=0}^{k-1} e_i^{b_{l,i+1}} &= \prod_{i=0}^{k-1} g^{b_{l,i+1} m^{i+1}} h^{b_{l,i+1} \Gamma_{i+1}} \\ &= g^{\sum_{i=0}^{k-1} b_{l,i+1} m^{i+1}} h^{\sum_{i=0}^{k-1} b_{l,i+1} \Gamma_{i+1}} \bmod p \end{aligned} \tag{10}$$

As $m \in S$, there must exist $l \in \{1, 2, \ldots, t\}$ such that $m \in S_l$. As

$$\begin{pmatrix} s_{l,1} & s_{l,1}^2 & \ldots & s_{l,1}^k \\ s_{l,2} & s_{l,2}^2 & \ldots & s_{l,2}^k \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ s_{l,k} & s_{l,k}^2 & \ldots & s_{l,k}^k \end{pmatrix} \begin{pmatrix} b_{l,1} \\ b_{l,2} \\ \ldots \\ \ldots \\ b_{l,k} \end{pmatrix} = \begin{pmatrix} s \\ s \\ \ldots \\ \ldots \\ s \end{pmatrix}$$

and $S_l = \{s_{l,1}, s_{l,2}, \ldots, s_{l,k}\}$, m satisfies

$$\sum_{i=1}^{k} b_{l,i} m^i = s \bmod q. \tag{11}$$

As $\omega_l = \prod_{i=0}^{k-1} e_i^{b_{l,i+1}}$, (10) and (11) imply that with a non-negligible probability

$$\omega_l = g^s h^{\sum_{i=0}^{k-1} b_{l,i+1}\Gamma_{i+1}} \bmod p$$

and thus s is committed to by the prover in ω_l with a non-negligible probability.

As the prover passes the verification in the new exclusion proof with a non-negligible probability, he must have successfully passed the t instances of proof of uncommitment with a non-negligible probability, say P_1. So according to Lemma 1, it is guaranteed with a probability P_1P_2 that s is not committed to by the prover in ω_l for any l in $\{1, 2, \ldots, t\}$ where P_2 is an overwhelmingly large probability. As P_1P_2 is non-negligible, it is guaranteed with an non-negligible probability that s is not committed to by the prover in ω_l for any l in $\{1, 2, \ldots, t\}$. So a contradiction is found. Therefore, the probability that a prover can pass the exclusion proof is negligible if $m \in S$. □

5 Efficiency Optimisation

The cost of the new inclusion proof and exclusion proof includes communicational cost and computational cost. In communication, $3k + 3t + 2$ integers are transfered in the new inclusion proof and $3k + 6t + 2$ integers are transfered in the new exclusion proof. Their computational cost is measured in terms of the number of exponentiations. When estimating their computational cost, we have an observation: exponentiations with small (in comparison with q) exponents like s_ρ^i with $1 \le i \le k$ is much less costly than an exponentiation with an exponent chosen from Z_q. Actually, the $k - 1$ exponentiations $s_\rho^2, s_\rho^3, \ldots, s_\rho^k$ can be calculated in a batch using $k - 1$ multiplications. So, in efficiency analysis of cryptographic protocols (e.g. threshold secret sharing [17,28]), an exponentiation used in Lagrange Interpolation is usually not counted like an exponentiation with a full-length exponent as its exponent is usually much smaller. So the number of exponentiations needed in the new inclusion proof is $3k + 4t - 3$ for the prover and $4k + n + 2t$ for a verifier, while the number of exponentiations needed in the new exclusion proof is $3k + 6t - 3$ for the prover and $3k + n + 6t$ for a verifier.

Efficiency of general inclusion proof and exclusion proof has been greatly improved in our work as $O(k) + O(t)$ is actually $O(\sqrt{n})$. For the first time, communicational cost of general inclusion proof and general exclusion proof in a set with cardinality n is reduced to $O(\sqrt{n})$. Computational cost of the prover is $O(\sqrt{n})$ exponentiations as well, the most efficient in publicly verifiable general solutions to inclusion proof and exclusion proof. However, computational cost of a verifier still includes n exponentiations, which are needed to calculate $\omega_l = \prod_{i=0}^{k-1} e_i^{b_{l,i+1}}$ for $l = 1, 2, \ldots, t$. Those n exponentiations is the bottleneck in efficiency of the new inclusion proof and exclusion proof technique.

To overcome this bottleneck, we exploit a special phenomenon in the new inclusion proof and the new exclusion proof, which does not happen in the existing solutions to inclusion proof or exclusion proof. That is in the t instances of calculation of ω_l the same k bases $e_0, e_1, \ldots, e_{k-1}$ are used. Although directly calculating $\omega_1, \omega_2, \ldots, \omega_t$ is costly for a verifier, verification of validity of them can be efficient if someone else knows (e.g. using some other more efficient method) and publishes them. In the the new inclusion proof and the new exclusion proof the prover can calculate each ω_l using no more than 2 exponentiations. So if he publishes $\omega_1, \omega_2, \ldots, \omega_t$ a verifier only needs to verify validity of them. Therefore, calculation of $\omega_1, \omega_2, \ldots, \omega_t$ by a verifier in the new inclusion proof and the new exclusion proof can be optimised as follows.

1. The prover calculates and publishes for $l = 1, 2, \ldots, t$

$$\omega_l = \begin{cases} g^s h^{\sum_{i=1}^{k} b_{l,i}\Gamma_i} \bmod p & \text{if } m \in S_l \\ g^{\sum_{i=1}^{k} b_{l,i} m^i} h^{\sum_{i=1}^{k} b_{l,i}\Gamma_i} \bmod p & \text{if } m \notin S_l \end{cases}$$

2. A verifier randomly chooses integers $\theta_1, \theta_2, \ldots, \theta_t$ from Z_τ where τ is a security parameter smaller than q.
3. The verifier checks

$$\textstyle\prod_{l=1}^{t} \omega_l^{\theta_l} = \prod_{i=0}^{k-1} e_i^{\sum_{l=1}^{t} \theta_l b_{l,i+1}} \bmod p. \tag{12}$$

He accepts validity of $\omega_1, \omega_2, \ldots, \omega_t$ iff (12) holds.

This method only transfers t integers and costs $t + k$ exponentiations, while as illustrated in Theorem 4, ω_l is guaranteed to be $\prod_{i=0}^{k-1} e_i^{b_{l,i+1}}$ for $l = 1, 2, \ldots, t$ if (12) is satisfied with a non-negligible probability.

Theorem 4. *If (12) is satisfied with a probability larger than $1/\tau$, then it is guaranteed that $\omega_l = \prod_{i=0}^{k-1} e_i^{b_{l,i+1}}$ for $l = 1, 2, \ldots, t$.*

Proof: For any integer L in $\{1, 2, \ldots, t\}$ there must exist integers $\theta_1, \theta_2, \ldots, \theta_{L-1}, \theta_{L+1}, \ldots, \theta_t$ in z_τ and two different integers θ_L and $\hat{\theta}_L$ in Z_τ such that

$$\textstyle\prod_{l=1}^{t} \omega_l^{\theta_l} = \prod_{i=0}^{k-1} e_i^{\sum_{l=1}^{t} \theta_l b_{l,i+1}} \bmod p \tag{13}$$

$$\textstyle(\prod_{l=1}^{L-1} \omega_l^{\theta_l}) \omega_L^{\hat{\theta}_L} \prod_{l=L+1}^{t} \omega_l^{\theta_l} \tag{14}$$

$$\textstyle= \prod_{i=0}^{k-1} e_i^{(\sum_{l=1}^{L-1} \theta_l b_{l,i+1}) + \hat{\theta}_L b_{L,i+1} + \sum_{l=L+1}^{t} \theta_l b_{l,i+1}} \bmod p$$

Otherwise, with this L for any combination of $\theta_1, \theta_2, \ldots, \theta_{L-1}, \theta_{L+1}, \ldots, \theta_t$ there is at most one θ_L to satisfy (12) among the τ possible choices of θ_L, which leads to a contradiction: the probability that (12) is satisfied is no larger than $1/\tau$. (13)/(14) yields

$$\omega_L^{\theta_L - \hat{\theta}_L} = \textstyle\prod_{i=0}^{k-1} e_i^{(\theta_L - \hat{\theta}_L) b_{L,i+1}} \bmod p$$

As $\theta_L,\ \hat{\theta}_L < \tau < q$ and q is prime, $(\theta_L - \hat{\theta}_L)^{-1} \bmod q$ exists. So

$$\omega_L = \prod_{i=0}^{k-1} e_i^{b_{L,i+1}} \bmod p$$

Note that L can be any integer in $\{1, 2, \ldots, t\}$. Therefore,

$$\omega_l = \prod_{i=0}^{k-1} e_i^{b_{l,i+1}} \text{ for } l = 1, 2, \ldots, t. \qquad \square$$

6 Comparison and Conclusion

The new inclusion proof protocol and the new exclusion protocol are compared with the existing solutions to inclusion proof and exclusion proof in Table 1, which clearly demonstrates the advantages of the new scheme in both security and efficiency. As stated in Section 1, we focus on proof techniques to especially designed to handle inclusion proof and exclusion proof in this paper. Communicational cost is estimated in terms of the number of transferred integers. Computational cost is estimated in terms of the number of exponentiations with bases in G (or similar large cyclic groups) and exponents in Z_q (or a similar large range as wide as the order of a large cyclic group). The simple exclusion proof is assumed to employ ZK proof of inequality of discrete logarithm in [4]. Our new solution costs $O(\sqrt{n})$ and is more efficient than all the existing general solutions including the simple inclusion proof, the simple exclusion proof and the inclusion proof protocol in [5], while the inclusion proof protocol and exclusion proof protocols in [3,14] are only special solutions working under strict conditions. Moreover, our new technique overcomes the drawbacks of the existing solutions.

Table 1. Comparison of inclusion proof and exclusion proof schemes

scheme	type	public	generality &	non-	communi-	computation	
		verifiability	flexibility	-interaction	-cation	prover	verifier
simple proof	inclusion	achieved	achieved	yes	$3n$	$2n-1$	$2n$
simple proof	exclusion	achieved	achieved	yes	$6n$	$6n$	$6n$
[3]	inclusion	achieved	no and strictly limited	yes	54	46	56
[14]	exclusion	achieved	no and strictly limited	yes	68	56	67
[5]	inclusion	no	achieved	no	$n+6$ for every verifier	7 for every verifier	$n+9$
new proof	inclusion	achieved	achieved	yes	$3k+4t+2$	$3k+4t-3$	$4k+3t$
new proof	exclusion	achieved	achieved	yes	$3k+7t+2$	$3k+6t-3$	$4k+7t$

References

1. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
2. Brickell, E., Gordon, D., McCurley, K., Wilson, D.: Fast exponentiation with precomputation. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 200–207. Springer, Heidelberg (1993)
3. 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)
4. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
5. 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)
6. Chaum, D., Pedersen, T.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
7. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
8. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
9. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In: FOCS 1986, pp. 174–187 (1986)
10. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Computer 18, 186–208 (1985)
11. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
12. Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 192–208. Springer, Heidelberg (2009)
13. Kilian, J., Petrank, E., Tardos, G.: Probabilistically checkable proofs with zero knowledge. In: STOC 1997, pp. 496–505 (1997)
14. Li, J., Li, N., Xue, R.: Universal Accumulators with Efficient Nonmembership Proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
15. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
16. Micali, S., Rabin, M., Kilian, J.: Zero-knowledge sets. In: IEEE FOCS 2003, p. 80 (2003)
17. Pedersen, T.: Distributed provers with applications to undeniable signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 221–242. Springer, Heidelberg (1991)
18. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.: Robust, Privacy Protecting and Publicly Verifiable Sealed-Bid Auction. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 147–159. Springer, Heidelberg (2002)

19. Peng, K., Boyd, C., Dawson, E., Lee, B.: Multiplicative Homomorphic E-Voting. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 61–72. Springer, Heidelberg (2004)
20. Peng, K., Boyd, C., Dawson, E.: Batch Verification of Validity of Bids in Homomorphic E-auction. Computer Communications 29(15), 2798–2805 (2006)
21. Peng, K., Bao, F.: Efficient Bid Validity Check in ElGamal-Based Sealed-Bid E-auction. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 209–224. Springer, Heidelberg (2007)
22. Peng, K., Bao, F.: Efficient Vote Validity Check in Homomorphic Electronic Voting. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 202–217. Springer, Heidelberg (2009)
23. Peng, K., Bao, F.: A Hybrid E-Voting Scheme. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 195–206. Springer, Heidelberg (2009)
24. Peng, K., Bao, F.: A Design of Secure Preferential E-Voting. In: Ryan, P.Y.A., Schoenmakers, B. (eds.) VOTE-ID 2009. LNCS, vol. 5767, pp. 141–156. Springer, Heidelberg (2009)
25. Peng, K., Bao, F.: Efficiency Improvement of Homomorphic E-Auction. In: Katsikas, S., Lopez, J., Soriano, M. (eds.) TrustBus 2010. LNCS, vol. 6264, pp. 238–249. Springer, Heidelberg (2010)
26. Peng, K., Bao, F.: Efficient Proof Of Validity of Votes In Homomorphic E-Voting. In: NSS 2010, pp. 17–23 (2010)
27. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology 4, 161–174 (1991)
28. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 149–164. Springer, Heidelberg (1999)

Non-interactive Confirmer Signatures

Sherman S.M. Chow[1,⋆] and Kristiyan Haralambiev[2]

[1] Department of Combinatorics and Optimization
University of Waterloo, Ontario, Canada N2L 3G1
smchow@math.uwaterloo.ca

[2] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
kkh@cs.nyu.edu

Abstract. The study of non-transferability of digital signatures, such as confirmer signatures, has enjoyed much interest over the last twenty years. In PKC '08, Liskov and Micali noted that all previous constructions of confirmer signatures consider only offline untransferability – non-transferability is not preserved if the recipient interacts concurrently with the signer/confirmer and an unexpected verifier. We view this as a result of all these schemes being interactive in the confirmation step. In this paper, we introduce the concept of non-interactive confirmer signatures (which can also be interpreted as extractable universal designated-verifier signatures). Non-interactive confirmer signatures give a neat way to ensure the online untransferability of signatures. We realize our notion under the "encryption of a signature" paradigm using pairings and provide a security proof for our construction without random oracles.

Keywords: non-interactive confirmer signature, extractable universal designated verifier signature, online untransferability.

1 Introduction

Non-transferability of digital signatures is an interesting research problem that has been investigated in various works over the last twenty years. A canonical application considers the scenario in which Alice wants to make an offer to Bob, but does not want Bob to show it to anybody else, so that Bob can not use Alice's offer as leverage to negotiate better terms or to gain any advantage. This covers the scenarios of job offers, contracts, receipt-free elections, and selling of malware-free software.

1.1 Undeniable Signatures and Confirmer Signatures

To address this problem, Chaum and van Antwerpen [1] introduced the notion of *undeniable signatures* which requires the signer's presence and cooperation for the recipient to verify the validity of a signature. In this way, the signer

⋆ A major part of the work was done while at New York University.

controls when the validity of the signature is being confirmed, and the validity is unknown without the participation of the signer. With this extra power of the signer, a basic security requirement is that the signer cannot cheat about the (in)validity of an undeniable signature when participating in the confirmation/disavowal protocols. However, there is no cryptographic means which can prevent a signer from refusing to cooperate. If the signer becomes unavailable or decides to "repudiate" the signature by ignoring any confirmation requests, the recipient is left with no cryptographic evidence of the signature's validity.

To overcome this disadvantage and to better ensure *non-repudiation*, Chaum [2] introduced a confirmer in this setting, which is a party other than the signer who can confirm/deny a signature. Now the trust (of willingness to participate in the protocol) is moved from the signer to the confirmer. Furthermore, this confirmer can extract an ordinary digital signature that is publicly and non-interactively verifiable (say when Bob has accepted the offer but Alice denies making one). This notion is known as designated confirmer signatures. In this paper, we follow the naming of some recent work and call it confirmer signatures.

Recently, Liskov and Micali [3] pointed out that all constructions of confirmer signatures provide only *offline untransferability*, and possibly Bob can "transfer" the validity of the signature by interacting with Alice and a verifier concurrently. They propose the notion of online-untransferable signatures to address this problem. However, their construction is inefficient due to the use of "cut-and-choose" proofs, i.e., the number of cryptographic operations like encryption and signing is linear in the security parameter. Also, the confirmer needs to either setup a public key for *each* signer, or to use an identity-based encryption (IBE) for the extraction of a publicly-verifiable signature. Both could be viewed as shortcomings of their construction, or the complexity one needs to pay to achieve online untransferability in an interactive setting. Lastly, their definitions deviate from the standard ones due to the absence of the confirmation protocol, which might be needed if the confirmer has to convince a verifier different from the recipient of a signature (e.g., in cases of checking integrity-critical content as part of a subscription service [4]). It is fair to say previous constructions either are inefficient or provide only offline untransferability.

1.2 (Universal) Designated-Verifier Signatures

Shortly after Chaum's work, Jakobsson *et. al.* [5] observed that undeniable signatures allow the signer to choose only whether to engage in the confirm/disavowal protocol but not with whom, i.e., it is possible that the recipient acts as a man-in-the-middle and executes the confirmation protocol with the signer so as to convince a third party. Moreover, this puts the signer at risk of being coerced and forced to participate in the confirmation protocol. To address these problems, they suggested the idea of *designated verifier proofs* which allows the prover (signer) to designate who will be convinced by the proof. If Alice wanted to convince Bob of the validity of a signature, or generally a statement θ, then Alice would prove to Bob that "*either θ is true, or I am Bob*". That would definitely convince Bob, but if he tried to transfer the proof to others, it would not imply anything about the validity of θ simply because the proof came from Bob.

Steinfeld *et. al.* [6] generalized this notion to *universal designated-verifier signatures* (UDVS). Unlike the original definition, one does not have to be the signer in order to convince a designated verifier. Anyone who is in possession of a regular signature can perform such a proof to a designated verifier $\mathcal{V}$. Their construction requires the verifiers to register their public/private key pairs with a key registration authority using the same system parameters as the signature scheme of the signer. This key registration model is a fairly common requirement when using public key infrastructure.

The original application of UDVS is motivated by privacy concerns associated with dissemination of signed digital certificates. An universal designator, which means any receiver of a certificate, can transfer the validity of the signature to any designated verifier. There is no mean to extract an ordinary (publicly-verifiable) signature from a UDVS. One way to do this is to ask the universal designator to hold the original signature. Alice still needs to trust that the signature will be kept confidential by this designator, i.e., active collusion with Bob or attack by Bob would not happen. Bob must ask Alice or the confirmer for the signature, which means Bob need to place trust on this middle-man too. In other words, we still need to make trust assumptions which may be implicitly made when using confirmer signatures. Simply put, the UDVS may provide online-untransferability, but not non-repudiation, as also pointed out in [3].

We remark that the universal designated verifier signature proof proposed by Baek *et. al.* [7] (which is later renamed to credential ownership proof [8]) does not has any requirement about whether the verifier cannot convince a third party that the message has been actually signed by a signer. In their proof systems, interactive protocols between the signature holder and the designated verifier are required to avoid the key requirement for the verifiers.

1.3 Related Work

Boyar *et. al.* [9] introduced the concept of convertible undeniable signatures, which allows the possibility of converting either a single undeniable signature or all undeniable signatures ever produced by a signer into an ordinary one. Similar to traditional undeniable/confirmer signature, confirmation is interactive.

ElAimani revisited in a series of work [10,11] the construction of undeniable or confirmer signature following the "encryption of a signature" paradigm firstly studied by Okamoto [12]. These studies identified the minimal security required for the encryption scheme in the generic constructions of strongly-unforgeable schemes which can be instantiated by a rather wide class of signature schemes. Since our scheme is also constructed using the "encryption of a signature" approach, all these constructions share similarity (except some technical details such as an ordinary signature[1] can be extracted from our scheme since we do

[1] By an ordinary signature, we mean that the signature just signs on the message of interest but includes nothing else. For example, there should be no other auxiliary information that is only useful for (the security of) the confirmer signature.

not aim at getting strong unforgebility). We view the merit of our work as a variation of the traditional paradigm which gives a conceptually simple solution to the online transferability problem.

1.4 Our Contribution

The research focus of confirmer signatures has been on defining the right model and constructing efficient schemes. We follow these directions here by introducing the concept of *non-interactive* confirmer signatures (NICS) and the first efficient (non-interactive) confirmer signature scheme with *online untransferability.* Our construction is secure in the key registration model without random oracles. This result can also be interpreted as *extractable* universal designated-verifier signatures (xUDVS), i.e., one could extract the underlying regular signature if in possession of the private extraction key. This extraction key pair is a single key pair for normal public key encryption, but not an extra key pair generated by a confirmer for each signer, nor a master public/private key pair for an identity-based encryption (c.f. [3]). Surprisingly, the works studying confirmer signatures and designated verifier proofs/signatures have been almost entirely independent despite both are originating from the same problem.

The correspondences between an NICS and an xUDVS are as follow. Signing and extraction are the same. Confirmation is done by taking an ordinary signature as an input and creating a new NICS/xUDVS with respect to *any* verifier who asked for a confirmation (in contrast to [3]). This process can be possibly done by the signer herself, or any holder of an ordinary signature (i.e., a universal confirmation) At the same time, it is possible to extract an ordinary signature out of the NICS/xUDVS by using the private key of a third-party.

The benefits of non-interactive confirmation are numerous. It provides non-transferability of signatures in a natural way and simplifies the traditional way of "sending an unverifiable signature to the verifier first, then the verifier asks for the signer/confirmer to confirm/disavow later". In particular, the disavowal protocol is not necessary any more. Most importantly, a non-interactive confirmer signature scheme avoids the problem of the recipient interacting concurrently with the signer and another verifier. So, the online untransferability is easily satisfied. We will see shortly that the security definition also becomes neater.

2 Non-interactive Model for Confirmer Signatures

2.1 Notations

Let $\mathsf{negl}(\kappa)$ denote a negligible function in κ where a function $\epsilon : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for all $c > 0$ and sufficiently large κ, $\epsilon(\kappa) \leq \kappa^{-c}$. For a finite set S, we denote $x \in_R S$ the sampling of an element x from S uniformly at random. If $\mathcal{A}$ is a PPT algorithm, $\mathcal{A}(x)$ denotes the output distribution of $\mathcal{A}$ on input x. We write $y \leftarrow \mathcal{A}(x)$ to denote the experiment of running $\mathcal{A}$ on input x and assigning the output to the variable y. Also, let

$$\Pr[x_1 \leftarrow X_1;\ x_2 \leftarrow X_2(x_1);\ \ldots;\ x_n \leftarrow X_n(x_1, \ldots, x_{n-1}) : \rho(x_1, \ldots, x_n)]$$

be the probability that the predicate $\rho(x_1, \ldots, x_n)$ is true when x_1 is sampled from the distribution X_1; x_2 is sampled from the distribution $X_2(x_1)$ which possibly depends on x_1; $x_3, \ldots, x_{n-1}$ are defined similarly; and finally x_n is sampled from distribution $X_n(x_1, \ldots, x_{n-1})$ which possibly depends on $x_1, \ldots, x_{n-1}$. The predicate might include execution of probabilistic algorithms.

2.2 Framework

Before describing different algorithms required in a non-interactive confirmer signature scheme, we first introduce the four kinds of participants involved, namely, signers, verifiers, adjudicator, and universal confirmer. The role of the former two are obvious. An adjudicator is mostly a passive entity who is assigned by a signer (possibly with the consent of the verifier given outside of our protocols) in the creation of a confirmer signature. A verifier may turn to an adjudicator when the signer refused to give an ordinary signature afterwards.

The role of a universal confirmer is not the same as a traditional one. For traditional confirmer/undeniable signatures, the signatures generated by the signer is ambiguous, i.e., it does not bind to the signer by itself. The job of a confirmer is to convince the verifier about the validity of a signature, which also means that the confirmer must maintain some secret information other than the signature itself, may it be a private key or some random values used by the signer during signature generation; otherwise anyone can confirm the validity of a signature.

In our setting, the signing process started by generating a regular signature that is binding to the signer. Consequently, there must be a step which converts an ordinary signature to an ambiguous one. We include this step as part of the confirmation. While the signature produced is ambiguous, the confirmation can still convince the verifier that the signer has signed on a particular message. Moreover, this can also confirm the fact that an ordinary signature by the signer on this message can also be extracted by an adjudicator. The confirmer in our notion is universal, which means that anyone who holds an ordinary signature can do this job. The confirmer does not need to maintain additional secret state information. Of course, the signer may perform the confirmation herself as well.

Now we are ready to define a non-interactive confirmer signature scheme with a global setup. Specifically, this setup decides the security parameter κ, the cryptographic groups to be used, and possibly a common reference string. All these will be included in the system parameters param, implicitly required by all algorithms. Like existing constructions, we require both signers and verifiers to register a public (verification) key with the key registration authority. One way to do that is to prove knowledge of the secret key during key registration.

Definition 1 (Non-Interactive Confirmer Signatures). *A non-interactive confirmer signature scheme is a signature scheme* $(\mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ *augmented with two suites of algorithms: First, for convincing a designated verifier, we have:*

– $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma, m)$*: takes as inputs an unordered pair of verification keys (one of the signer and one of the verifier), an adjudicator public key* pk*, and a signature/message pair which is valid for any key among* $\{\mathsf{vk}_S, \mathsf{vk}_V\}$*; outputs a confirmer signature* $\hat{\sigma}$*.*

– $\mathsf{DVer}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \hat{\sigma}, m)$*: takes as inputs an unordered pair of verification keys, a confirmer signature* $\hat{\sigma}$ *and a message* m*; outputs* 1 *if* $\hat{\sigma}$ *is an output of* $\mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma, m)$*, where* σ *is a signature of* m *verifiable under* vk_0 *or* vk_1*; otherwise, outputs 0.*

For extracting an ordinary signature for a verifier, we have:

– $\mathsf{EKGen}(1^\kappa)$*: outputs a private/public key* $(\mathsf{xk}, \mathsf{pk})$ *for the adjudicator.*
– $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma})$*: takes as inputs the private extraction key and a valid (publicly verifiable) confirmer signature outputted by* $\mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma, m)$*, outputs an ordinary signature* σ *and a bit* b *indicating that* σ *is given under the signing key corresponding to* vk_b*.*

In the traditional notion of confirmer signatures, the job of our adjudicator about extracting an ordinary signature is also performed by the confirmer. Here we distill this task out. As argued before, a confirmer may be absent, as a designated verifier can "confirm" the signature $\hat{\sigma}$ by himself. On the other hand, any holder of an ordinary signature σ can designate the proof to any verifier, similar to the functionality of the universal designated verifier signatures.

2.3 Security Requirements

Our model borrows ideas from both confirmer signatures and universal designated verifier signatures. Compared with traditional confirmer signatures, our definition is considerably neater since the confirmation and disavowal protocols are essentially replaced by a single Des algorithm. Most importantly, all previous constructions are interactive and their security requirements are defined with respect to that. We see that as a reason why all these works can only achieve offline untransferability and a more complex definition is required to ensure online untransferability. We believe that our definition satisfy all fundamental properties of confirmer signatures. The following requirements should be satisfied for all system parameters param generated by the global setup.

Definition 2 (Correctness). *A non-interactive confirmer signature scheme (of security parameter* κ*) is correct if the four conditions below are satisfied (with overwhelming probability in* κ*).*

– ***Key-Correctness***
We assume it is efficient to check for the validity of all kinds of key pairs in our system. We denote this check by the predicate $\mathsf{Valid}(\cdot)$ *which takes as implicit input the key pair type. Specifically, we require that* $\mathsf{Valid}(\mathsf{sk}, \mathsf{vk}) = 1$ *and* $\mathsf{Valid}(\mathsf{xk}, \mathsf{pk}) = 1$ *for all* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(1^\kappa)$ *and* $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa)$*.*
– ***Sign-Correctness***
For all messages m*, and all* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(1^\kappa)$*,* $\mathsf{Ver}(\mathsf{vk}, m, \mathsf{Sig}(\mathsf{sk}, m)) = 1$*.*
– ***Designate-Correctness***
For all messages m*, all* $(\mathsf{sk}_S, \mathsf{vk}_S), (\mathsf{sk}_V, \mathsf{vk}_V) \leftarrow \mathsf{SKGen}(1^\kappa)$*, and all* $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa)$*, and all* $\hat{\sigma} \leftarrow \mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \mathsf{Sig}(\mathsf{sk}_S, m), m)$*, we expect that* $\mathsf{DVer}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \hat{\sigma}, m) = 1$*, where* $\{\mathsf{vk}_S, \mathsf{vk}_V\}$ *is ordered lexicographically.*

- ***Extract-Correctness***
  For all messages m, all $(\mathsf{sk}_0, \mathsf{vk}_0), (\mathsf{sk}_1, \mathsf{vk}_1) \leftarrow \mathsf{SKGen}(1^\kappa)$*, all* $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa)$*, all* $b \in \{0, 1\}$*, all* $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_b, m)$*, and finally for all* $\hat{\sigma} \leftarrow \mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma, m)$*, it is true that* $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma}) = (\sigma, b)$*.*

Privacy. Regarding the privacy requirement of the confirmer signatures, some existing works consider invisibility, which informally means that an adversary $\mathcal{A}$ with the signing keys and accesses to extraction oracles cannot distinguish whether a confirmer signature is on which of the two chosen messages m_0 or m_1. However, this is more about the messages invisibility but may not protect the signer if the adversary is interested in knowing if the purported signer has participated or not instead (say Eve will agree to give Bob a job offer if Alice does, and how much salary Alice is going to offer Bob does not really matter to Eve). So here we consider the privacy requirement of the real signer, which ensures that $\mathcal{A}$ cannot distinguish whether a confirmer signature is signed using a signing key sk_0 or sk_1.

Definition 3 (Source Privacy)

$$\begin{aligned}
&|\Pr[\ (\mathsf{xk}, \mathsf{pk}) \leftarrow \mathsf{EKGen}(1^\kappa); (m, (\mathsf{sk}_0, \mathsf{vk}_0), (\mathsf{sk}_1, \mathsf{vk}_1), state) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{E}}_{\mathsf{xk}}(\cdot)}(\mathsf{param}, \mathsf{pk});\\
&\qquad b \leftarrow \{0, 1\}; \sigma^* \leftarrow \mathsf{Sig}(\mathsf{sk}_b, m); \hat{\sigma}^* \leftarrow \mathsf{Des}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \sigma^*, m);\\
&\qquad b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{E}}_{\mathsf{xk}}(\cdot)}(state, \hat{\sigma}^*) :\\
&\qquad b = b' \mid \mathsf{Valid}(\mathsf{sk}_0, \mathsf{vk}_0)\ \wedge\ \mathsf{Valid}(\mathsf{sk}_1, \mathsf{vk}_1)] - \tfrac{1}{2} \mid <\ \mathsf{negl}(\kappa)
\end{aligned}$$

where $\mathcal{O}^{\mathsf{E}}_{\mathsf{xk}}(\cdot)$ *is an extraction oracle, taking a valid confirmer signature* $\hat{\sigma} \neq \hat{\sigma}^*$ *as an input and returning the output of* $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma})$*.*

One might think at first that the non-interactive confirmer signatures reveal too much information about the possible signer. However, while the two possible signers are known, they are equally probable as a real signer.

Note on Indistinguishability: Another way to model the privacy guarantee of the real signer is based on an indistinguishability notion. Specifically, there is a PPT algorithm Des' taking as inputs a message, the verification key of the purported signer, and possibly the signing and the verification key of a designated verifier, and outputting a fake/simulated signature such that it looks indistinguishable as a real signature generated by Des to any adversary $\mathcal{A}$. If the capability of extracting ordinary signature is present, $\mathcal{A}$ is disallowed to win in a trivial way, such as having the extraction key xk or asking the oracle to use xk to extract the signature out of the confirmer/designated-verifier signature in question.

We claim that our privacy notion implies indistinguishability. Our privacy notion guarantees that it is difficult to tell which key among $\{\mathsf{vk}_S, \mathsf{vk}_V\}$ is the actual verification key for the σ in $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma, m)$. A faking/simulation algorithm Des' under our framework is readily available. We can do that by using the designated-verifier's signing key sk_V to create a signature $\sigma = \mathsf{Sig}(\mathsf{sk}_V, m)$ and then create a confirmer signature $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma, m)$. This is exactly how a signature is created when the roles of the signer and the verifier are interchanged.

A similar argument is used in [6] to argue for *unconditional* indistinguishability in their case. While in our case, we added an encryption of the "real" verification key, so we can only achieve computational indistinguishability.

Stronger Privacy to Protect All Possible Signers' Identities: If one insists on hiding the verification keys in the designated-verifier signature, recall that it was shown in [12] that the notion of confirmer signatures is equivalent to public key encryption, so one may always add an extra layer of encryption and encrypt the whole signature under a public key of the designated verifier to achieve stronger privacy guarantee, without relying on any additional assumption.

Soundness. While privacy protects the signer, the verifier is protected by the soundness guarantee. Intuitively, the verifier does not want to get a confirmer signature $\hat{\sigma}$ which is valid according to the DVer algorithm, but the extracted ordinary signature of which cannot pass the Ver algorithm.

Definition 4 (Extraction Soundness)

$$\begin{aligned}\Pr[\ &(m, (\mathsf{sk}_0, \mathsf{vk}_0), (\mathsf{sk}_1, \mathsf{vk}_1), (\mathsf{xk}, \mathsf{pk}), \hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{param}); (\sigma, b) = \mathsf{Ext}(\mathsf{xk}, \hat{\sigma}) : \\ &\mathsf{DVer}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \hat{\sigma}, m) = 1 \ \wedge\ \mathsf{Ver}(\mathsf{vk}_b, \sigma, m) = 0 \\ &\wedge \mathsf{Valid}(\mathsf{sk}_0, \mathsf{vk}_0) \ \wedge\ \mathsf{Valid}(\mathsf{sk}_1, \mathsf{vk}_1) \ \wedge\ \mathsf{Valid}(\mathsf{xk}, \mathsf{pk})\ ] \ <\ \mathsf{negl}(\kappa)\end{aligned}$$

The extraction soundness only guarantees that a valid confirmer signature $\hat{\sigma}$ can be extracted to a valid ordinary signature created by either one of the secret keys. Since the creation of confirmer signature is universal in our scheme, it is possible that a designated verifier who expects to get someone else's signature eventually may end up with getting his own signature! An easy solution to this problem is to register a key pair which is only for the purpose of designation but not for signing any message. If a single key pair is used, a user should be cautious in what signature to accept and what to sign. For the offer scenario, it means that Bob should protect his private key secretly and refuse to sign any message similar to "xxx is willing to offer to Bob yyy", and should never get convinced by a confirmer signature on a message without mentioning who is agreed to make the offer, which should be a common practice to follow in checking an offer.

Unforgeability. There are two types of unforgeability to be considered. The first is the standard notion of unforgeability under chosen message attack. The second one similarly requires that it is infeasible to compute a confirmer signature $\hat{\sigma}$ on a new message which could be verified using DVer for a pair of verification keys, the secret keys of which are unknown. Note that Des is a public algorithm which requires no secret knowledge, so like in the regular unforgeability game the adversary needs access only to a signing oracle for the unknown signing keys.

Definition 5 (Unforgeability)

$$\begin{aligned}\Pr[\ &(\mathsf{sk}_0, \mathsf{vk}_0), (\mathsf{sk}_1, \mathsf{vk}_1) \leftarrow \mathsf{SKGen}(1^\kappa); \\ &(m^*, \sigma^*, \hat{\sigma}^*, (\mathsf{xk}, \mathsf{pk})) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_0}(\cdot), \mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_1}(\cdot)}(\mathsf{param}); \\ &\rho = \mathsf{Ver}(\mathsf{vk}_0, \sigma^*, m^*); \hat{\rho} = \mathsf{DVer}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \mathsf{pk}, \hat{\sigma}^*, m^*) : \\ &m^* \notin \{m_i\} \ \wedge\ \mathsf{Valid}(\mathsf{xk}, \mathsf{pk}) \ \wedge\ (\rho = 1 \ \vee\ \hat{\rho} = 1)\ ] \ <\ \mathsf{negl}(\kappa)\end{aligned}$$

where $\{m_i\}$ is the set of messages supplied to $\mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_0}$ or $\mathcal{O}^{\mathsf{S}}_{\mathsf{sk}_1}$ which take a message and output a digital signature signed with the secret key sk_0 and sk_1, respectively.

Note that extraction soundness "converts" that any forgery of the second type to a forgery of the first type as the extracted signature has to be verifiable under one of the verification keys. One may also try to define unforgeability by allowing the adversary to generate $(\mathsf{sk}_1, \mathsf{vk}_1)$ and only considering $\hat{\sigma}^*$ is a valid forgery if one can extract a valid signature under vk_0 from $\hat{\sigma}^*$. However, this definition turns out to be equivalent to our unforgeability definition above.

3 Preliminaries

3.1 Number Theoretic Assumptions

Definition 6 (Bilinear Map). *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of prime order p. A bilinear map $\hat{e}(\cdot,\cdot) : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies:*

- non-degeneracy: *$\hat{e}(g, g)$ is a generator of $\mathbb{G}_T$ when g is a generator of $\mathbb{G}$;*
- bilinearity: *for all $x, y \in \mathbb{Z}_p$, $\hat{e}(g^x, g^y) = \hat{e}(g, g)^{xy}$.*

Definition 7 (Decisional Linear (DLIN) Assumption [13]). *The DLIN assumption holds if for all PPT adversaries, on input a sextuple $(u, v, g, u^a, v^b, g^c) \in \mathbb{G}^6$, where $c = a + b$ or c is a random element in $\mathbb{Z}_p$ with equal probability, the probability of guessing which is the case correctly over $\frac{1}{2}$ is negligible.*

Definition 8 (q-Strong Diffie-Hellman (q-SDH) Assumption [14]). *The q-SDH assumption holds if for all PPT adversaries, it is of negligible probability to find a pair $(m, g^{\frac{1}{m+x}}) \in \mathbb{Z}_p \times \mathbb{G}$ when given $(g, g^x, g^{x^2}, \ldots, g^{x^q}) \in \mathbb{G}^{q+1}$.*

3.2 Cryptographic Building Blocks

Definition 9 (Collision-Resistant Hash Function (CRHF)). *A family of hash functions $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{l(\kappa)}$ is said to be collision resistant if for all PPT adversaries $\mathcal{A}$, we have:*

$$\Pr[H \in_R \mathcal{H};\ x, y \leftarrow \mathcal{A}(H) : H(x) = H(y)] < \mathsf{negl}(\kappa).$$

We use a CRHF to compress messages of any length to messages of length $l(\kappa)$, where $2^{l(\kappa)} < p$ and p is the order of the groups we are working with.

Definition 10 (Signature Schemes). *A signature scheme Σ is a triple of PPT algorithms $(\mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ with the following properties:*

- SKGen*: takes as an input a security parameter 1^κ; outputs a signing key sk and a corresponding verification key vk.*
- Sig*: takes as inputs a signing key sk and a message m and outputs a signature $\sigma = \mathsf{Sig}(\mathsf{sk}, m)$.*

– Ver*: takes as inputs a verification key* vk*, a message* m *and a purported signature* σ*; outputs either* 1 *or* 0 *denoting "accept" or "reject".*
– *Correctness:* $\forall \kappa \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(1^\kappa)$, $\mathsf{Ver}(\mathsf{vk}, \mathsf{Sig}(\mathsf{sk}, m), m) = 1$.

Definition 11 (Strong One-Time Signature Scheme). *A signature scheme* Σ *is said to be strongly secure one-time signature, if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr[\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}(); (m, state) \leftarrow \mathcal{A}(\mathsf{vk}); \sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, m);$$
$$(\sigma^*, m^*) \leftarrow \mathcal{A}(state, \sigma) : \mathsf{Ver}(\mathsf{vk}, \sigma^*, m^*) = 1 \wedge (\sigma^*, m^*) \neq (\sigma, m)] < \mathsf{negl}(\kappa).$$

In our construction, we could use any strong one-time signature scheme, preferably with security follows from some of the aforementioned assumptions. One such scheme is described in [15] which is based on the DLIN assumption. This scheme is also used in [16].

Definition 12 (Weakly-Secure Signatures). *A signature scheme* Σ *is defined to be secure against weak chosen message attack (wEUF-secure) if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr[\ (m_1, m_2, \ldots, m_q) \leftarrow \mathcal{A}(1^\kappa); (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SKGen}();$$
$$\sigma_i = \mathsf{Sig}(\mathsf{sk}, m_i); (m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \sigma_1, \ldots, \sigma_q) :$$
$$\mathsf{Ver}(\mathsf{vk}, \sigma^*, m^*) = 1 \wedge m^* \notin \{m_1, \ldots, m_q\}\ ] < \mathsf{negl}(\kappa).$$

We use the signature scheme of Boneh and Boyen [14] which is wEUF-secure under the q-SDH assumption. The system parameters are $(p, \mathbb{G}, \mathbb{G}_T, g, \hat{e}(\cdot,\cdot), H(\cdot))$, where H is a collision-resistant hash function with range in $\mathbb{Z}_p$.

– $\mathcal{BB}.\mathsf{SKGen}()$: Pick $\mathsf{sk} \in_R \mathbb{Z}_p^*$ and compute $\mathsf{vk} = g^{\mathsf{sk}}$; output the pair of private signing key and the public verification key as $(\mathsf{sk}, \mathsf{vk})$.
– $\mathcal{BB}.\mathsf{Sig}(\mathsf{sk}, m)$: Output the signature $\sigma = g^{\frac{1}{\mathsf{sk}+H(m)}}$. (It fails to sign on m if $H(m) = -\mathsf{sk}$.)
– $\mathcal{BB}.\mathsf{Ver}(\mathsf{vk}, \sigma, m)$: Accept if and only if $\hat{e}(\sigma, \mathsf{vk} \cdot g^{H(m)}) = \hat{e}(g, g)$.

Tag-Based Encryption

Kiltz [17] extended the linear encryption [13] to a tag-based encryption which is secure against selective-tag weak chosen-ciphertext attacks (CCA), under the decision linear assumption.

– $\mathcal{TBE}.\mathsf{EKGen}(1^\kappa)$: The encryption key is $(u, v, g_0, U, V) \in \mathbb{G}^5$ where $u^a = v^b = g_0$, and the decryption key is (a, b).
– $\mathcal{TBE}.\mathsf{Enc}((u, v, g_0, U, V), m, \ell)$: To encrypt a message $m \in \mathbb{G}$ under a tag (or a label) $\ell \in \mathbb{Z}_p^*$, picks $\varphi, \psi \in_R \mathbb{Z}_p^*$ and returns $(T_1, T_2, T_3, T_4, T_5) = (u^\varphi, v^\psi, m g_0^{\varphi+\psi}, (g_0^\ell U)^\varphi, (g_0^\ell V)^\psi)$,
– $\mathcal{TBE}.\mathsf{Dec}((a, b), (T_1, T_2, T_3, T_4, T_5), \ell)$: To decrypt $(T_1, T_2, T_3, T_4, T_5)$, return $T_3/(T_1^a \cdot T_2^b)$ if $\hat{e}(u, T_4) = \hat{e}(T_1, g_0^\ell U)$ and $\hat{e}(v, T_5) = \hat{e}(T_2, g_0^\ell V)$ hold. The latter check can also be done without pairing if the discrete logarithm of U, V with respect to u, v respectively are kept as part of the private key.

The message space of this encryption scheme is $\mathbb{G}$, which matches with the signature space as well as the verification key space of the signature scheme $\mathcal{BB}$.

Non-interactive Proofs for Bilinear Groups

Groth and Sahai [18] developed techniques for proving statements expressed as equations of certain types. Their proofs are non-interactive and in the common references string (CRS) model. The proofs have perfect completeness and, depending on the CRS, perfect soundness or witness indistinguishability / zero-knowledge. The two types of CRS are computationally indistinguishable. Groth and Sahai showed how to construct such proofs under various assumptions, one of which is the decisional linear assumption.

The first type of equations were interested in is linear equations over $\mathbb{G}$ (described as multi-exponentiation of constants in the one sided case in [18]), of the form $\prod_{j=1}^{L} a_j^{\chi_j} = a_0$, where $\chi_1, \dots \chi_L$ are variables and $a_0, a_1, \dots, a_L \in \mathbb{G}$ are constants. Such equations allow to prove equality of committed and encrypted values, with the randomness used to commit and encrypt being the witness (the assignment of the variables) which satisfies the corresponding equations. The proofs for this type of equations are zero-knowledge, i.e. valid proofs could be produced without a witness using the trapdoor of the simulated CRS.

Pairing product equations allow to prove validity of $\mathcal{BB}$ signatures without revealing the signature and/or the verification key, i.e., $\hat{e}(\sigma, v \cdot g^m) = \hat{e}(g, g)$ for variables σ and v. The Groth-Sahai proofs for this type of equations are only witness indistinguishable, which is sufficient for our purposes, though could be transformed into zero-knowledge proofs if certain requirements are met like in the case of the above equation.

The last type of equations we need is to assert the plaintext of an encryption $\mathfrak{C}$ is one of two publicly known messages m_1 and m_2. The is the key step for the designated verification. Rather than using OR-proofs which do not mesh well with Groth-Sahai proofs, we introduce two additional variables α, β to be used in the exponent for which we prove $\alpha + \beta = 1$, both α and $\beta \in \{0, 1\}$, and the ciphertext $\mathfrak{C}$ being an encryption of $m_1^{\alpha} m_2^{\beta}$. The first proof is done using the linear equation $g^{\alpha} g^{\beta} = g$; $\alpha, \beta \in \{0, 1\}$ is proven using the technique of Groth *et. al.* [19] which constructs a witness-indistinguishable proof for a commitment of $\chi \in \mathbb{Z}_p$ being a commitment of 0 or 1; and that $\mathfrak{C}$ contains the proper plaintext is shown using linear equations.

4 Our Construction

4.1 High Level Idea

We present an efficient non-interactive confirmer signature scheme using the tools we just described. The key property is that our confirmer signature is publicly verifiable, but its holder cannot tell whether it was produced by the signer or the designated verifier (unless, of course, in possession of the extraction key). One elegant way to make this possible is that we create the designated-verifier signature by proving the statement "*θ is a valid signature signed by either Alice or Bob*". While it is essentially the concept of ring signature [20], it is the confirmer who performs the designation in our case, but not the signer. We stress that, in existing applications of using a 2-user ring signature as a designated verifier signature, *the signer* is the one who performs the designation. In other words, the universal confirmer has the ability to "turn" *any* ordinary signature into a 2-user ring signature by involving any designated verifier in the "ring".

To make this proof efficient and non-interactive, we employ the NIZK/NIWI proof system proposed by Groth and Sahai [18]. This is a powerful primitive but the target language is limited. Hence, building blocks for the construction are chosen so as to be within this limit. For the signature scheme, we combine a strong one-time signature and a weak $\mathcal{BB}$ signature [14] we reviewed earlier, the latter of which is convenient when constructing efficient non-interactive proofs. The $\mathcal{BB}$ signature is used to sign random messages, i.e., verification keys for the one-time signature scheme, so security against weak chosen message attack suffices, we then use the one-time signature scheme to sign on the actual message.

To realize our universal designation, the designation algorithm makes commitments for the verification key and the $\mathcal{BB}$ signature, proves those committed values satisfy the verification equation with the message being the one-time verification key, and proves that the verification key commitment is either of the signer's or of the designated verifier's verification key.

To achieve extractability of ordinary signature and satisfy our privacy requirement, we require a CCA2-secure public-key encryption. The encryption scheme should be chosen so we could prove equality of committed and encrypted values using the Groth-Sahai proofs. Two appropriate schemes are the tag-based encryption scheme of Kiltz [17], and Shacham's Linear Cramer-Shoup [21]. Proving that the plaintext of an encryption is one of two possible messages require some extra work as we described in Section 3.2.

To prevent the adversary from massaging the confirmer signature and learn (partial) information of the real signer or the encrypted signature, we employ a public key encryption scheme with label and another one-time signature to ensure the non-malleability of the confirmer signature.

The verification and designated verification are straightforward – simply checking the validity of the signatures/proofs. The extraction algorithm uses the secret key of the encryption scheme to decrypt the $\mathcal{BB}$ signature and the verification key from the corresponding ciphertexts.

4.2 Instantiation

– $\mathsf{Setup}(1^\kappa)$: This setup algorithm takes up a bit string 1^κ, picks groups $\mathbb{G}, \mathbb{G}_T$ of prime order p with a bilinear map $\hat{e}(\cdot,\cdot)$, where $2^\kappa < p < 2^{\kappa+1}$. This bilinear map context determines the common reference string (CRS) for the Groth-Sahai proof system, which in turn determines the Groth-Sahai commitment function $\mathsf{Com}(m; \boldsymbol{r})$ which commits to $m \in \mathbb{G}$ or $m \in \mathbb{Z}_p$ using appropriately sampled randomness vector $\boldsymbol{r}$. This algorithm also chooses a collision resistant hash function $H(\cdot) : \{0,1\}^* \to \mathbb{Z}_p$.

  All these parameters are concatenated into a single string param. For brevity, we omit the inclusion of param in the interface of our algorithms, which makes some of the algorithms like $\mathsf{EKGen}()$ has no explicit input.

– $\mathsf{SKGen}()$: $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathcal{BB}.\mathsf{SKGen}()$.
– $\mathsf{Sig}(\mathsf{sk}_S, m)$:
  1. $(\mathsf{osk}, \mathsf{ovk}) \leftarrow \mathcal{OTS}.\mathsf{SKGen}()$.
  2. If $H(\mathsf{ovk}) = -\mathsf{sk}_S$, repeat step 1.
  3. Output $\sigma = (\mathsf{ovk}, \sigma_{bb} = \mathcal{BB}.\mathsf{Sig}(\mathsf{sk}_S, H(\mathsf{ovk})), \sigma_{ots} = \mathcal{OTS}.\mathsf{Sig}(\mathsf{osk}, m))$.
– $\mathsf{Ver}(\mathsf{vk}, \sigma = (\mathsf{ovk}, \sigma_{bb}, \sigma_{ots}), m)$: Output 1 if and only if $\mathcal{BB}.\mathsf{Ver}(\mathsf{vk}, \sigma_{bb}, H(\mathsf{ovk}))$ and $\mathcal{OTS}.\mathsf{Ver}(\mathsf{ovk}, \sigma_{ots}, m)$ both output 1; otherwise, output 0.
– $\mathsf{EKGen}()$: $(\mathsf{xk}, \mathsf{pk}) \leftarrow \mathcal{TBE}.\mathsf{EKGen}()$.
– $\mathsf{Des}(\{\mathsf{vk}_S, \mathsf{vk}_V\}, \mathsf{pk}, \sigma = (\mathsf{ovk}, \sigma_{bb}, \sigma_{ots}), m)$:
  - Initialization:
    1. $(\mathsf{osk}', \mathsf{ovk}') \leftarrow \mathcal{OTS}.\mathsf{SKGen}()$.
    2. Order $\{\mathsf{vk}_S, \mathsf{vk}_V\}$ lexicographically into $(\mathsf{vk}_0, \mathsf{vk}_1)$.
  - Commit and encrypt the verification key and prove their well-formedness:
    1. Encrypt vk_S labelled with ovk' in $\mathfrak{C}_{\mathsf{vk}} \leftarrow \mathcal{TBE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{vk}_S, \boldsymbol{r'}, H(\mathsf{ovk}'))$.
    2. Create π_{enc} which is a proof that $\mathfrak{C}_{\mathsf{vk}}$ is an encryption of $\mathsf{vk}_S = \mathsf{vk}_0^{\alpha}\mathsf{vk}_1^{\beta}$ using NIZK proofs for satisfiable linear equations with variables $\boldsymbol{r'}, \alpha, \beta$, with proofs for $\alpha + \beta = 1$ and $\alpha, \beta \in \{0, 1\}$.
    3. Create a commitment of the verification key by $C_{\mathsf{vk}} = \mathsf{Com}(\mathsf{vk}_S;\ \boldsymbol{r})$.
    4. Create π_{vk} which is a proof of equality of the committed/encrypted values in C_{vk} and $\mathfrak{C}_{\mathsf{vk}}$ using NIZK proofs for satisfiable linear equations with variables $\boldsymbol{r'}, \boldsymbol{r}$.
  - Commit and encrypt the key-certifying signature and prove their well-formness:
    1. Encrypt σ_{bb} labelled with ovk' in $\mathfrak{C}_{\sigma} \leftarrow \mathcal{TBE}.\mathsf{Enc}(\mathsf{pk}, \sigma_{bb};\ \boldsymbol{s'}, H(\mathsf{ovk}'))$.
    2. Create a commitment of the signature by $C_{\sigma} = \mathsf{Com}(\sigma_{bb};\ \boldsymbol{s})$.
    3. Create π_{σ} which is a proof of equality of the committed/encrypted values in C_{σ} and $\mathfrak{C}_{\sigma}$ using NIZK proofs for satisfiable linear equations with variables $\boldsymbol{s'}, \boldsymbol{s}$.
  - Linking all pieces together:
    1. Create π_{sgn} which is an NIWI proof of validity of the $\mathcal{BB}$ signature for the committed values of C_{vk} and C_{σ}; C_{vk} and C_{σ} are commitments produced by the proof system to create π_{sgn} but are given explicitly in the construction as we require equality of committed values (used for the proofs) and encrypted ones (used for extraction).
    2. During the creation of the above proofs, commitments of the variables $\boldsymbol{r'}, \boldsymbol{r}, \boldsymbol{s'}, \boldsymbol{s}$ are also created. Let π be this set of the proofs $\pi_{enc}, \pi_{vk}, \pi_{\sigma}, \pi_{sgn}$ and the associated commitments. Also, let $m' = (\mathsf{ovk}', \mathsf{vk}_0, \mathsf{vk}_1, \mathsf{ovk},\ \sigma_{ots},\ \mathfrak{C}_{\mathsf{vk}},\ \mathfrak{C}_{\sigma},\ \pi)$.
    3. Sign on the string m' by $\hat{\sigma}' \leftarrow \mathcal{OTS}.\mathsf{Sig}(\mathsf{osk}', m')$.
    4. Output $\hat{\sigma} = (\hat{\sigma}', m')$.
– $\mathsf{DVer}(\{\mathsf{vk}_0, \mathsf{vk}_1\}, \hat{\sigma} = (\hat{\sigma}', m'), m)$: Verify the one-time signatures $\hat{\sigma}'$ on m' under ovk' and all the NIZK/NIWI proofs; also check that $\{\mathsf{vk}_0, \mathsf{vk}_1\}$ are the same verification keys (after ordering them lexicographically) as those in m'. Finally, verify the one-time signature σ_{ots} on m under ovk. If any of the verification is not successful, return 0; otherwise, return 1.

– $\mathsf{Ext}(\mathsf{xk}, \hat{\sigma} = (\hat{\sigma}', m'))$:
1. Parse m' as $(\mathsf{ovk}', \mathsf{vk}_0, \mathsf{vk}_1, \mathsf{ovk}, \sigma_{ots}, \mathfrak{C}_{\mathsf{vk}}, \mathfrak{C}_\sigma, \pi)$.
2. Decrypt $\mathfrak{C}_\sigma$ to get $\sigma_{bb} = \mathcal{TBE}.\mathsf{Dec}(\mathsf{xk}, \mathfrak{C}_\sigma, H(\mathsf{ovk}'))$.
3. Decrypt $\mathfrak{C}_{\mathsf{vk}}$ to get $\mathsf{vk} = \mathcal{TBE}.\mathsf{Dec}(\mathsf{xk}, \mathfrak{C}_{\mathsf{vk}}, H(\mathsf{ovk}'))$.
4. If any decryption was rejected or $\mathcal{OTS}.\mathsf{Ver}(\mathsf{ovk}', m') = 0$, output $(\perp, \perp)$.
5. Output $((\mathsf{ovk}, \sigma_{bb}, \sigma_{ots}), b)$ where $\mathsf{vk}_b = \mathsf{vk}$.

In the the full version, we prove the following theorem:

Theorem 13. *The described non-interactive confirmer signature scheme is secure under the Decisional Linear assumption and the q-SDH assumption, assuming the hash function is collision resistant. That is, the scheme satisfies the correctness, privacy, soundness, and unforgeability requirements.*

4.3 Discussion

All the proofs used in our scheme can be done by variants of the proofs in some existing cryptosystems involving Groth-Sahai proofs as mentioned in Section 3.2. Basically, our confirmer signature is proving that an encrypted signature is a valid one under an encrypted public key, where the public key comes from one of two possibilities. The two possibilities part involves an OR proof as discussed in Section 3.2. The encryption part involves proving that the plaintext of the encryption and the committed value in the corresponding commitment are the same. The proofs of the latter kind for the encryption scheme $\mathcal{TBE}$ has appeared in existing group signature schemes (e.g. [22, Section 7]). With these proofs, the rest is about proving the signature in the commitment verifies, and the corresponding proof for the signature scheme $\mathcal{BB}$ involves a simple pairing product equation which can be found in many "privacy-oriented" Groth-Sahai-proof-based cryptosystems such as anonymous credential (e.g. [23, Section 5]) and group signatures (e.g. [22, Section 7]). Due to the space limitation, we defer the details on the language or the equation required to the full version.

There is an important difference regarding the usage of the signature scheme in the aforementioned systems [23,22] and in our scheme. For the latter, the signature scheme is often used in certifying a user public key by the private key of the "authority". Obviously, the verification of a valid credential/signature would not require the knowledge of the user private key, and hence the signature is essentially signing on a user private key which is "hidden" in the output of a certain one-way function. On the other hand, we just use the signature scheme to sign on a public one-time signature verification key. This is the reason why we do not consider a possible stronger notion such as F-unforgeability [23].

Regarding efficiency, each signature consists of roughly 100 group elements, while the scheme of Liskov-Micali [3] produces signatures with $O(\kappa)$ ciphertexts.

5 Concluding Remarks

We unify the concept of confirmer signatures and designated-verifier signatures. Specifically, we introduce the notion of *non-interactive* confirmer signatures, which can also be interpreted as *extractable* universal designated-verifier signatures. Besides saving in valuable rounds of interaction, we believe a non-interactive construction of confirmer signatures represents a more natural instantiation of the primitive. Most importantly, it resolves the problem of online transferability [3] when the recipient is acting as a man-in-the-middle, in a simple and computationally-efficient way. Our proposed construction is a proof-of-concept scheme. There are many possibilities for optimization. For examples, one may improve our construction by picking better underlying primitives, or try to get rid of using encryption by leveraging the strong unforgeability [24]. Finally, for practical application, one may consider resorting to random oracle model and propose a possibly more efficient implementation.

References

1. Chaum, D., Antwerpen, H.V.: Undeniable Signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
2. Chaum, D.: Designated Confirmer Signatures. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 86–91. Springer, Heidelberg (1995)
3. Liskov, M., Micali, S.: Online-untransferable signatures. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 248–267. Springer, Heidelberg (2008)
4. Gentry, C., Molnar, D., Ramzan, Z.: Efficient Designated Confirmer Signatures Without Random Oracles or General Zero-Knowledge Proofs. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 662–681. Springer, Heidelberg (2005)
5. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
6. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal Designated-Verifier Signatures. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (2003)
7. Baek, J., Safavi-Naini, R., Susilo, W.: Universal Designated Verifier Signature Proof (or How to Efficiently Prove Knowledge of a Signature). In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 644–661. Springer, Heidelberg (2005)
8. Shahandashti, S.F., Safavi-Naini, R., Baek, J.: Concurrently-Secure Credential Ownership Proofs. In: ASIACCS, pp. 161–172 (2007)
9. Boyar, J., Chaum, D., Damgård, I., Pedersen, T.P.: Convertible Undeniable Signatures. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 189–205. Springer, Heidelberg (1991)
10. Aimani, L.E.: Toward a Generic Construction of Universally Convertible Undeniable Signatures from Pairing-Based Signatures. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 145–157. Springer, Heidelberg (2008)
11. Aimani, L.E.: On Generic Constructions of Designated Confirmer Signatures. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 343–362. Springer, Heidelberg (2009)
12. Okamoto, T.: Designated Confirmer Signatures and Public-Key Encryption are Equivalent. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 61–74. Springer, Heidelberg (1994)
13. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

14. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
15. 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)
16. 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)
17. 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)
18. 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)
19. Groth, J., Ostrovsky, R., Sahai, A.: Perfect Non-interactive Zero Knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
20. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret: Theory and Applications of Ring Signatures. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science. LNCS, vol. 3895, pp. 164–186. Springer, Heidelberg (2006)
21. Shacham, H.: A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants. Cryptology ePrint Archive, Report 2007/074 (2007), http://eprint.iacr.org/
22. Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
23. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and Noninteractive Anonymous Credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
24. Schuldt, J.C.N., Matsuura, K.: An Efficient Convertible Undeniable Signature Scheme with Delegatable Verification. In: Kwak, J., Deng, R.H., Won, Y., Wang, G. (eds.) ISPEC 2010. LNCS, vol. 6047, pp. 276–293. Springer, Heidelberg (2010)

Communication-Efficient 2-Round Group Key Establishment from Pairings

Kashi Neupane and Rainer Steinwandt

Department of Mathematical Sciences, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA
{kneupane,rsteinwa}@fau.edu

Abstract. In a recent preprint, Vivek et al. propose a compiler to transform a passively secure 3-party key establishment to a *passively* secure group key establishment. To achieve *active* security, they apply this compiler to Joux's protocol and apply a construction by Katz and Yung, resulting in a 3-round group key establishment.

In this paper we show how Joux's protocol can be extended to an *actively* secure group key establishment with two rounds. The resulting solution is in the standard model, builds on a bilinear Diffie-Hellman assumption and offers forward security as well as strong entity authentication. If strong entity authentication is not required, then one half of the participants does not have to send any message in the second round, which may be of interest for scenarios where communication efficiency is a main concern.

Keywords: group key establishment, pairing, standard model.

1 Introduction

Group key establishment protocols enable a set of $n \geq 2$ users to establish a common key over a public communication network. To obtain a constant round solution, i. e., a protocol where the number of rounds is independent of the number of participants n, a common technique is to impose a ring topology on the set of participants and to establish pairwise Diffie-Hellman keys among neighbors first. In view of Joux's one-round 3-party key establishment [Jou04], it is natural to ask if this primitive could be used to devise an alternative construction for group key establishment. Indeed, in [VSDSR09] a compiler is presented using a three-party key establishment instead of a two-party solution as fundamental building block. To achieve active security,[1] Vivek et al. suggest to combine a construction in [VSDSR09] with a well-known compiler by Katz and Yung [KY03], which relies on the availability of a strongly unforgeable signature scheme. Overall, a 3-round solution for group key establishment is obtained from Joux's protocol.

[1] To avoid a possible ambiguity with strong entity authentication, we avoid the term *authenticated group key establishment* when discussing security in the presence of an active adversary.

With 2-round solutions for group key establishment being available, one may ask for the existence of an alternative construction building on Joux's protocol. Below we provide such a construction in the standard model using a bilinear Diffie-Hellman assumption. The protocol is inspired by the 2-round construction of Bohli et al. [BVS07], but unlike the latter rests on Joux's 3-party key establishment as basic primitive—rather than a 2-party Diffie-Hellman key establishment. An interesting feature is that by restricting to the "standard" security guarantee of key secrecy, which does not imply strong entity authentication, half of the participants have to send only one protocol message. This seems to render our protocol an interesting candidate for settings where communication is costly. As a minor technical point, our protocol relies on an existentially unforgeable signature scheme; strong unforgeability is not needed.

Further related work. After submission of our manuscript we became aware of further related work. In particular, Desmedt and Lange's independent earlier work in [DL08] presents a passively secure two-round solution similar to our construction. To achieve active security, they mention the application of a compiler from [KY03, DLB07], resulting in a 3-round solution. In [CHL04, DWGW03, ZSM06] pairing-based 2-round solutions in an *identity-based setting* are suggested, assuming a suitable trusted authority to be available.

2 Technical Preliminaries

In this section we quickly revisit the relevant terminology and definitions from the literature. No originality is claimed for this section.

2.1 Security Model and Security Goals

The security model we use for our analysis is taken from [BVS07] and includes strong entity authentication as security goal. This "oracle based" model builds on work by Bresson et al. [BCP01] and by Katz and Yung [KY03].

Protocol participants. The set of protocol participants is denoted by $\mathcal{U}$ and of size polynomial in the security parameter k. We model each *user* $U \in \mathcal{U}$ as probabilistic polynomial time (ppt) algorithm and each $U \in \mathcal{U}$ can execute a polynomial number of protocol instances Π_U^s concurrently ($s \in \mathbb{N}$). Further, we assume that all user identities are encoded as bitstrings of identical length and for the ease of notation will subsequently not distinguish between (the algorithm) U and the bitstring describing its identity. With each protocol instance Π_U^s, the following seven variables are associated:

acc_U^s**:** is set to TRUE if the session key stored in sk_U^s has been accepted;
pid_U^s**:** stores the identities of those users in $\mathcal{U}$ with which a key is to be established, including U;
sid_U^s**:** stores a session identifier, i. e., a non-secret identifier for the session key stored in sk_U^s;

sk_U^s**:** is initialized with a distinguished NULL value and after a successful protocol execution stores the session key;
state_U^s**:** stores state information needed for executing the protocol;
term_U^s**:** is set to TRUE if this protocol execution has terminated;
used_U^s**:** indicates if this instance is used, i. e., involved in a protocol run.

Initialization. Before actual protocol executions take place, there is a trusted initialization phase *without adversarial interference.* In this phase, for each $U \in \mathcal{U}$ a (verification key, signing key)-pair $(pk_U, sk_U^{\mathsf{sig}})$ for an existentially unforgeable (EUF-CMA secure) signature scheme is generated, sk_U^{sig} is given to U only, and pk_U is handed to all users in $\mathcal{U}$ and to the adversary.

Adversarial capabilities and communication network. The network is fully asynchronous, allows arbitrary point-to-point connections among users, and it is non-private. The adversary $\mathcal{A}$ is represented as ppt algorithm with full control over the communication network. More specifically, $\mathcal{A}$'s capabilities are expressed through the following *oracles*:

$\mathsf{Send}(U, s, M)$: This oracle serves two purposes.
- The Send oracle enables $\mathcal{A}$ to initialize a protocol execution; sending the special message $M = \{U_{i_1}, \dots, U_{i_r}\}$ with $U \in M$ to an unused instance $\prod_U^s$ initializes a protocol run among $U_{i_1}, \dots, U_{i_r} \in \mathcal{U}$. After such a query, $\prod_U^s$ sets $\mathsf{pid}_U^s := \{U_{i_1}, \dots, U_{i_r}\}$, $\mathsf{used}_U^s :=$ TRUE, and processes the first step of the protocol.
- The message M is sent to instance Π_U^s and the protocol message output by Π_U^s after receiving M is returned.

$\mathsf{Reveal}(U, s)$: returns the session key sk_U^s if $\mathsf{acc}_U^s =$ TRUE and a NULL value otherwise.

$\mathsf{Corrupt}(U)$: for a user $U \in \mathcal{U}$ this query returns U's long term signing key sk_U^{sig}.

Unlike Reveal, $\mathsf{Corrupt}$ addresses a *user*, rather than an individual *protocol instance* of a user. An adversay with access to all of the above oracles is considered *active.* To capture a *passive* adversary, access to Send can be replaced with access to a dedicated $\mathsf{Execute}$ oracle, returning a protocol transcript. An active adversary can simulate such an $\mathsf{Execute}$ oracle by means of Send in the obvious manner, and we therefore do not include $\mathsf{Execute}$.

In addition to the above, there is a Test oracle, and $\mathcal{A}$ must submit exactly one query $\mathsf{Test}(U, s)$ with an instance Π_U^s that has accepted a session key, i. e., $\mathsf{acc}_U^s =$ TRUE has to hold. In response to such a query, a bit $b \leftarrow \{0, 1\}$ is chosen uniformly at random and for $b = 0$ the value of sk_U^s, i. e., the established session key, is returned. For $b = 1$ the output is a uniformly at random chosen element from the space of session keys. The idea is that for a secure protocol, no efficient adversary can distinguish between $b = 0$ and $b = 1$. To make this precise we need some more terminology.

First, to exclude trivialities, we impose key establishment protocols to be *correct*: in the absence of active attacks a common session key is established, along with common session identifier and matching partner identifier. Next, we rely on the following notion of *partnered* instances.

Definition 1 (Partnering). *Two instances* $\prod_{U_i}^{s_i}$ *and* $\prod_{U_j}^{s_j}$ *are* partnered *if* $\mathsf{sid}_{U_i}^{s_i} = \mathsf{sid}_{U_j}^{s_j}$, $\mathsf{pid}_{U_i}^{s_i} = \mathsf{pid}_{U_j}^{s_j}$ *and* $\mathsf{acc}_{U_i}^{s_i} = \mathsf{acc}_{U_j}^{s_j} = \text{TRUE}$.

Based on this notion of partnering, we can specify what we mean by a *fresh* instance, i. e., an instance where the adversary should not know the session key:

Definition 2 (Freshness). *An instance* $\prod_{U_i}^{s_i}$ *is said to be* fresh *if the adversary queried neither* $\mathsf{Corrupt}(U_j)$ *for some* $U_j \in \mathsf{pid}_{U_i}^{s_i}$ *before a query of the form* $\mathsf{Send}(U_k, s_k, *)$ *with* $U_k \in \mathsf{pid}_{U_i}^{s_i}$ *has taken place, nor* $\mathsf{Reveal}(U_j, s_j)$ *for an instance* $\prod_{U_j}^{s_j}$ *that is partnered with* $\prod_{U_i}^{s_i}$.

It is worth noting that the above definition allows an adversary $\mathcal{A}$ to reveal *all* secret signing keys without violating freshness, provided $\mathcal{A}$ does not send any messages after having received the signing keys. As a consequence security in the sense of Definition 3 below implies forward secrecy: We write $\mathsf{Succ}_{\mathcal{A}}$ for the event $\mathcal{A}$ queries Test with a fresh instance and outputs a correct guess for the Test oracle's bit b. By

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{ke}} = \mathrm{Adv}_{\mathcal{A}}^{\mathsf{ke}}(k) := \left|\Pr[\mathsf{Succ}] - \frac{1}{2}\right|$$

we denote the *advantage* of $\mathcal{A}$.

Definition 3 (Semantic security). *A key establishment protocol is said to be* (semantically) secure*, if* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{ke}} = \mathrm{Adv}_{\mathcal{A}}^{\mathsf{ke}}(k)$ *is negligible for all ppt algorithms* $\mathcal{A}$.

In addition to the above standard security goal, we are also interested in *integrity* (which may be interpreted a form of "worst case correctness") and *strong entity authentication*:

Definition 4 (Integrity). *A key establishment protocol fulfills* integrity *if with overwhelming probability for all instances* $\prod_{U_i}^{s_i}$, $\prod_{U_j}^{s_j}$ *of uncorrupted users the following holds: if* $\mathsf{acc}_{U_i}^{s_i} = \mathsf{acc}_{U_j}^{s_j} = \text{TRUE}$ *and* $\mathsf{sid}_{U_i}^{s_i} = \mathsf{sid}_{U_j}^{s_j}$, *then* $\mathsf{sk}_{U_i}^{s_i} = \mathsf{sk}_{U_j}^{s_j}$ *and* $\mathsf{pid}_{U_i}^{s_i} = \mathsf{pid}_{U_j}^{s_j}$.

Definition 5 (Strong entity authentication). *We say that* strong entity authentication *to an instance* $\Pi_{U_i}^{s_i}$ *is provided if* $\mathsf{acc}_{U_i}^{s_i} = \text{TRUE}$ *implies that for all uncorrupted* $U_j \in \mathsf{pid}_{U_i}^{s_i}$ *there exists with overwhelming probability an instance* $\Pi_{U_j}^{s_j}$ *with* $\mathsf{sid}_{U_j}^{s_j} = \mathsf{sid}_{U_i}^{s_i}$ *and* $U_i \in \mathsf{pid}_{U_j}^{s_j}$.

2.2 Common Reference String

In the key derivation of our protocol, we follow the technique used in [ABVS07] (which in turn goes back to [KS05]) and we refer to [KS05] for a construction assuming the existence of a one-way permutation. For this we fix a collision-resistant pseudorandom function family $\mathcal{F} = \{F^k\}_{k\in\mathbb{N}}$ and assume $F^k = \{F^k_\ell\}_{\ell\in\{0,1\}^l}$ to be indexed by a (superpolynomial) size set $\{0,1\}^l$. In the common reference string (accessible to the adversary and all users) we encode two values $v_i = v_i(k)$ such that no ppt algorithm can compute $\lambda \neq \lambda'$ satisfying $F^k_\lambda(v_i) = F^k_{\lambda'}(v_i)$ $(i = 0, 1)$.

Moreover, we encode in the common reference string also an index into a family of universal hash functions $\mathcal{UH}$, specifying one UH $\in \mathcal{UH}$. This function UH will be used to translate common key material to an index into the mentioned collision-resistant pseudorandom function family.

2.3 Bilinear Diffie-Hellman Assumption

On the mathematical side, our main tool is a suitable pairing: Let $(G, +)$, $(G', \cdot)$ be two groups of prime order q, and denote by P a generator of G. We assume q to be superpolynomial in the security parameter k and that all group operations in G and G' can be computed by appropriate ppt algorithms.

Definition 6 (Admissible bilinear map). *We say that $e : G \times G \longrightarrow G'$ is an* admissible bilinear map *if all of the following hold:*

- *There is a ppt algorithm computing $e(P, Q)$ for all $P, Q \in G$.*
- *For all $P, Q \in G$ and all integers a, b we have $e(aP, bQ) = e(P, Q)^{ab}$.*
- *We have $e(P, P) \neq 1$, i. e., $e(P, P)$ generates G'.*

Now consider the following experiment for a ppt algorithm $\mathcal{A}$ outputting 0 or 1: The challenger chooses $a, b, c, d \in \{0, \ldots, q-1\}$ independently and uniformly at random and in addition flips a random coin $\delta \in \{0, 1\}$ uniformly at random. If $\delta = 0$, the tuple $(P, aP, bP, cP, e(P, P)^d)$ is handed to $\mathcal{A}$, whereas for $\delta = 1$ the tuple $(P, aP, bP, cP, e(P, P)^{abc})$ is handed to $\mathcal{A}$. Then $\mathcal{A}$ wins the game whenever the guess δ' it outputs for δ is correct; the advantage of $\mathcal{A}$ is denoted by

$$\mathrm{Adv}^{\mathsf{bdh}}_{\mathcal{A}} := \left|\Pr[\delta = \delta'] - \frac{1}{2}\right| .$$

The hardness assumption underlying the security analysis of the protocol discussed in the next section is that no efficient algorithm can win this game with non-negligible probability:

Definition 7 (Decisional Bilinear Diffie-Hellman assumption)
The Decisional Bilinear Diffie-Hellman assumption ($\mathsf{D\text{-}BDH}$) *for (G, G', e) holds, if the advantage $\mathrm{Adv}^{\mathsf{bdh}}_{\mathcal{A}}$ in the above experiment is negligible for all ppt algorithms $\mathcal{A}$.*

With this, we are in the position to describe our group key establishment.

3 A Pairing-Based Group Key Establishment

To describe our protocol, we write $U_0, \ldots, U_{n-1}$ for the protocol participants who want to establish a common session key. We assume the number n of these participants to be greater than three and even—if $2 \nmid n$, then U_{n-1} can simulate an additional (virtual) user U_n.

3.1 Description of the Protocol

We arrange the participants $U_0, \ldots, U_{n-1}$ in a circle such that $U_{(i-j) \bmod n}$ respectively $U_{(i+j) \bmod n}$ is the participant j positions away from U_i in counterclockwise (left) respectively clockwise (right) direction. Figure 1 describes both rounds of the proposed construction; to simplify notation, we do not explicitly refer to protocol instances $\Pi_i^{s_i}$.

Round 1:

Computation Each U_i chooses $u_i \in \{0, \ldots, q-1\}$ uniformly at random and computes $u_i P$. Users U_i with $2 \mid i$ in addition compute a signature σ_i^{I} on $\mathsf{pid}_i \| u_i P$.

Broadcast Each U_i broadcasts $u_i P$ (if $2 \nmid i$) respectively $(u_i P, \sigma_i^{\mathrm{I}})$ (if $2 \mid i$).

Check: Each U_i verifies $\sigma_0^{\mathrm{I}}, \sigma_2^{\mathrm{I}}, \ldots, \sigma_{n-2}^{\mathrm{I}}$ (using pid_i for the partner identifier). If any check fails, U_i aborts, otherwise U_i computes

$$\begin{cases} t_i^L := e(P,P)^{u_{(i-2) \bmod n} u_{(i-1) \bmod n} u_i} \text{ and} & \\ t_i^R := e(P,P)^{u_i u_{(i+1) \bmod n} u_{(i+2) \bmod n}} & \text{, if } i \text{ is odd} \\ t_i^M := e(P,P)^{u_{(i-1) \bmod n} u_i u_{(i+1) \bmod n}} & \text{, if } i \text{ is even} \end{cases}.$$

Round 2:

Computation Each U_i computes $\mathsf{conf}_i := (\mathsf{pid}_i \| u_0 P \| u_1 P \| \ldots \| u_{n-1} P)$ and

$$\begin{cases} \text{a signature } \sigma_i^{\mathrm{II}} \text{ on } \mathsf{conf}_i \| T_i \text{ where } T_i := t_i^L / t_i^R & \text{, if } i \text{ is odd} \\ \text{a signature } \sigma_i^{\mathrm{II}} \text{ on } \mathsf{conf}_i & \text{, if } i \text{ is even} \end{cases}$$

Broadcast Each U_i broadcasts $(\sigma_i^{\mathrm{II}}, T_i)$ (if $2 \nmid i$) respectively σ_i^{II} (if $2 \mid i$).

Check Each U_i verifies $\sigma_0^{\mathrm{II}}, \ldots, \sigma_{n-1}^{\mathrm{II}}$ (using the $u_j P$ received in Round 1 and pid_i for the partner identifier) and checks if $T_1 \cdot T_3 \cdot T_5 \ldots T_{n-1} = 1$ holds. If any of these checks fails, U_i aborts.

Key derivation: Each U_i recovers the values t_j^R for $j = 1, 3, \ldots, n-1$ as follows:

– U_i with $2 \nmid i$ finds $t_j^R = t_i^L \cdot \prod_{\substack{s=2 \\ 2 \mid s}}^{(i-j-2) \bmod n} T_{(j+s) \bmod n}$

– U_i with $2 \mid i$ finds $t_j^R = t_i^M \cdot \prod_{\substack{s=2 \\ 2 \mid s}}^{(i-j-1) \bmod n} T_{(j+s) \bmod n}$

Finally, each U_i computes the master key $K := (t_1^R, t_3^R, \ldots, t_{n-1}^R, \mathsf{pid}_i)$, sets $\mathsf{sk}_i := F_{\mathrm{UH}(K)}(v_0)$ and $\mathsf{sid}_i := F_{\mathrm{UH}(K)}(v_1)$.

Fig. 1. A 2-round group key establishment derived from Joux's protocol

Thus, in Round 1 users U_i with odd index i perform two executions of Joux's protocol, and users with an even index i perform only one such 3-party key establishment. For the actual key derivation, the messages sent by users with even index in Round 2 are not really needed, and as shown in Proposition 1, omitting those messages does not affect the semantic security of the protocol. Strong entity authentication is no longer guaranteed then, however, as an adversary could simply replay an old message of, say, U_0 in Round 1.

3.2 Security Analysis

The following proposition shows that the protocol in Figure 1 is secure in the sense of Definition 3 and—if we insist on all Round 2 messages being sent—also offers strong entity authentication.

Proposition 1. *Suppose that the* D-BDH *assumption holds for* (G, G', e) *and the underlying signature scheme is secure in the sense of* EUF-CMA*. Then the following hold:*

- *The protocol in Figure 1 is semantically secure, fulfills integrity, and strong entity authentication holds to all involved instances.*
- *If users* U_i *with* i *even do not send their Round 2 messages, the protocol in Figure 1 is semantically secure and fulfills integrity.*

Proof. Let q_{send} be a polynomial upper bound for the number of queries to the Send oracle by $\mathcal{A}$ and denote by Forge the event that $\mathcal{A}$ succeeds in forging a signature σ_i in the protocol without having queried Corrupt(U_i). Moreover, denote by $\text{Adv}^{\mathsf{uf}} = \text{Adv}^{\mathsf{uf}}(k)$ an upper bound for the probability that a ppt adversary creates a successful forgery for the underlying signature scheme. During the protocol's initialization phase, we can assign a challenge verification key to a user $U \in \mathcal{U}$ uniformly at random, and with probability at least $1/|\mathcal{U}|$ the event Forge results in a successful forgery for the challenge verification key. Thus

$$\Pr[\mathsf{Forge}] \leq |\mathcal{U}| \cdot \text{Adv}^{\mathsf{uf}},$$

and the event Forge can occur with negligible probability only.

Security. We prove the security of the protocol by "game hopping", letting adversary $\mathcal{A}$ interact with a simulator. The advantage of $\mathcal{A}$ in Game i will be denoted by $\text{Adv}_{\mathcal{A}}^{\text{Game } i}$:

Game 0: This game is identical to the original attack game, with all oracles of the adversary being simulated faithfully. Consequently,

$$\text{Adv}_{\mathcal{A}} = \text{Adv}_{\mathcal{A}}^{\text{Game } 0}.$$

Game 1: If the event Forge occurs, we abort the game and count this as a successful attack. Otherwise the game is identical with Game 0:

$$|\text{Adv}_{\mathcal{A}}^{\text{Game } 1} - \text{Adv}_{\mathcal{A}}^{\text{Game } 0}| \leq \Pr(\mathsf{Forge}).$$

Game 2: In this game we modify the adversary in such a way that at the beginning she guesses (randomly) which instance $\Pi_{i_0}^{s_{i_0}}$ will be queried to the Test oracle as well as two instances of $\Pi_{i_0}^{s_{i_0}}$ with which $\Pi_{i_0}^{s_{i_0}}$ will in Round 1 establish a 3-party key $t_{i_0}^R$. Whenever at least one of these guesses turns out to be wrong, we abort the simulation and consider the adversary to be at loss. Otherwise the game is identical with Game 1. Consequently,

$$\frac{1}{q_{\text{send}}^3} \cdot \text{Adv}_{\mathcal{A}}^{\text{Game 1}} \leq \text{Adv}_{\mathcal{A}}^{\text{Game 2}},$$

and as q_{send} is polynomial in k it will suffice to recognize $\text{Adv}_{\mathcal{A}}^{\text{Game 2}}$ as negligible.

Game 3: This game differs from Game 2 in the simulator's response in Round 2. Instead of computing $t_{i_0}^R$ resp. $t_{i_0}^M$ as specified in Round 1, the simulator replaces $t_{i_0}^R$ resp. $t_{i_0}^M$ with a uniformly at random chosen element in G'. For consistency, the corresponding key of the other 2 participants in this 3-party key establishment is replaced with the same random value.
We have $|\text{Adv}_{\mathcal{A}}^{\text{Game 3}} - \text{Adv}_{\mathcal{A}}^{\text{Game 2}}| \leq |\Pr(\mathsf{Succ}_{\mathcal{A}}^{\text{Game 3}}) - \Pr(\mathsf{Succ}_{\mathcal{A}}^{\text{Game 2}})|$, and to recognize the latter as negligible consider the following algorithm $\mathcal{B}$ to solve the D-BDH problem: $\mathcal{B}$ faithfully simulates all parties and oracles as faced by $\mathcal{A}$ in Game 2 with two exceptions. Namely, let $(P, aP, bP, cP, e(P,P)^x)$ be the D-BDH challenge received by $\mathcal{B}$. Then

– in Round 1, $\mathcal{B}$ replaces the random value $u_{i_0}P$ with aP, and analogously bP and cP are used as Round 1 message for the other two participants in the 3-party key establishment with U_{i_0};
– the 3-party key $t_{i_0}^R$ resp. $t_{i_0}^M$ of these three parties is not computed as specified in the protocol (in fact, $\mathcal{B}$ does not know a, b, c) but replaced with the value $e(P,P)^x$ in the D-BDH challenge.

Whenever $\mathcal{A}$ correctly identifies the secret bit of the (simulated) Test oracle, $\mathcal{B}$ outputs a 1, i. e., claims $x = abc$. By construction we have

$$\begin{aligned}\text{Adv}_{\mathcal{B}}^{\mathsf{bdh}} &= \left|\frac{1}{2}\cdot\Pr[\mathsf{Succ}_{\mathcal{A}}^{\text{Game 2}}] + \frac{1}{2}\cdot(1-\Pr[\mathsf{Succ}_{\mathcal{A}}^{\text{Game 3}}])] - \frac{1}{2}\right|\\ &= \frac{1}{2}\cdot\left|\Pr[\mathsf{Succ}_{\mathcal{A}}^{\text{Game 3}}] - \Pr[\mathsf{Succ}_{\mathcal{A}}^{\text{Game 2}}]\right|,\end{aligned}$$

and with the D-BDH assumption we recognize $|\text{Adv}_{\mathcal{A}}^{\text{Game 3}} - \text{Adv}_{\mathcal{A}}^{\text{Game 2}}|$ as negligible.

Game 4: In this game the simulation of the Test oracle is modified: instead of returning with probability 1/2 the correctly computed session key sk_{i_0}, always a uniformly at random chosen bitstring is returned. As after the replacement in the previous game one of the entries of the master key K is a uniformly chosen random value, K has sufficient entropy so that $F_{\text{UH(K)}}(v_0)$ is computationally indistinguishable from a random bitstring, i. e., $\left|\text{Adv}_{\mathcal{A}}^{\text{Game 4}} - \text{Adv}_{\mathcal{A}}^{\text{Game 3}}\right|$ is negligible. With $\text{Adv}_{\mathcal{A}}^{\text{Game 4}} = 0$, the semantic security of the protocol in Figure 1 follows. Moreover, we observe that the proof nowhere relied on the Round 2 messages of parties U_i with $2 \mid i$ being sent.

Integrity. Successful signature verification in Round 1 for U_i with $2 \mid i$ and for U_i with $2 \nmid i$ in Round 2 implies that the pid_i-values of all involved parties are identical, and integrity follows from the collision-resistance of the underlying pseudorandom function family.

Entity authentication. Successful verification of the signatures on the Round 2 messages ensures the existence of a used instance for each intended communication partner and that the respective conf_i-values are identical. The latter implies equality of both the pid_i- and the sid_i-values. □

3.3 Making Use of a Random Oracle

If one is willing to make a random oracle assumption, the usage of a universal hash function and a pseudorandom function family in the above protocol is of course no longer required, and we can compute the session key and session identifier as $\mathsf{sk}_i = H(K \parallel 0)$ and $\mathsf{sid}_i = H(K \parallel 1)$, respectively. From an engineering point of view, working in such an idealized model and using a standard cryptographic hash function to implement the random oracle H could be attractive. Going further, with a random oracle $H : \{0,1\}^* \longrightarrow \{0,1\}^k$ we can also replace the values t_i^L, t_i^R, t_i^M from Round 1 with their images under H, so that for computing T_i we only compute $H(t_i^L) \oplus H(t_i^R)$, i. e., instead of arithmetic in G' we can use XOR. Figure 2 shows the relevant changes to the protocol in Figure 1 if a random oracle is used in this way.

Adapting the above security analysis to this random oracle-based variant is straightforward—owing to the random oracle the D-BDH assumption can be replaced with a computational assumption in the usual manner. In the random oracle formulation, the similarity of our proposal with the 2-round group key establishment suggested in [BVS07] is quite apparent, and it is worth highlighting some differences:

– With the main building block in our protocol being Joux's 3-party key establishment, we rely on a (computational) bilinear Diffie-Hellman assumption rather than an ordinary (computational) Diffie-Hellman assumption.
– All protocol participants now have to perform one or two pairing computations, followed by one or two exponentiations, to compute a 3-party key—the number of exponentiations depending on the position in the circle being odd or even. In [BVS07] two exponentiations per participant yield common (Diffie-Hellman) keys with the clockwise and counter-clockwise neighbor in the circle.
– In the protocol proposed above, the session key is derived directly from the t_i^R-values, whereas the approach in [BVS07] relies on separate key contributions for this, one of them being masked by the t_i^R-values. These key contributions (or a hash value of such) are sent in a signed Round 1 message, resulting in a total of two signature computations per participant; in our construction at least those parties U_i with $2 \nmid i$ sign only one message.
– Bohli et al. compute the session identifier based on the Round 1 messages, whereas our construction relies on the availability of the Round 2 messages to do so.

Round 1:

Computation Each U_i chooses $u_i \in \{0, \ldots, q-1\}$ uniformly at random and computes u_iP. Users U_i with $2 \mid i$ in addition compute a signature σ_i^{I} on $\mathsf{pid}_i \| u_iP$.

Broadcast Each U_i broadcasts u_iP (if $2 \nmid i$) respectively $(u_iP, \sigma_i^{\mathrm{I}})$ (if $2 \mid i$).

Check: Each U_i verifies $\sigma_0^{\mathrm{I}}, \sigma_2^{\mathrm{I}}, \ldots, \sigma_{n-2}^{\mathrm{I}}$ (using pid_i for the partner identifier). If any check fails, U_i aborts, otherwise U_i computes

$$\begin{cases} t_i^L := H(e(P,P)^{u_{(i-2) \bmod n} u_{(i-1) \bmod n} u_i}) \text{ and} & \\ t_i^R := H(e(P,P)^{u_i u_{(i+1) \bmod n} u_{(i+2) \bmod n}}) & \text{, if } i \text{ is odd} \\ t_i^M := H(e(P,P)^{u_{(i-1) \bmod n} u_i u_{(i+1) \bmod n}}) & \text{, if } i \text{ is even} \end{cases} .$$

Round 2:

Computation Each U_i computes $\mathsf{conf}_i := (\mathsf{pid}_i \| u_0P \| u_1P \| \ldots \| u_{n-1}P)$ and

$$\begin{cases} \text{a signature } \sigma_i^{\mathrm{II}} \text{ on } \mathsf{conf}_i \| T_i \text{ where } T_i := t_i^L \oplus t_i^R & \text{, if } i \text{ is odd} \\ \text{a signature } \sigma_i^{\mathrm{II}} \text{ on } \mathsf{conf}_i & \text{, if } i \text{ is even} \end{cases}$$

Broadcast Each U_i broadcasts $(\sigma_i^{\mathrm{II}}, T_i)$ (if $2 \nmid i$) respectively σ_i^{II} (if $2 \mid i$).

Check Each U_i verifies $\sigma_0^{\mathrm{II}}, \ldots, \sigma_{n-1}^{\mathrm{II}}$ (using the u_jP received in Round 1 and pid_i for the partner identifier) and checks if $T_1 \oplus T_3 \oplus T_5 \oplus \cdots \oplus T_{n-1} = 0$ holds. If any of these checks fails, U_i aborts.

Key derivation: Each U_i recovers the values t_j^R for $j = 1, 3, \ldots, n-1$ as follows:

– U_i with $2 \nmid i$ finds $t_j^R = t_i^L \oplus \bigoplus_{\substack{s=2 \\ 2|s}}^{(i-j-2) \bmod n} T_{(j+s) \bmod n}$

– U_i with $2 \mid i$ finds $t_j^R = t_i^M \oplus \bigoplus_{\substack{s=2 \\ 2|s}}^{(i-j-1) \bmod n} T_{(j+s) \bmod n}$

Finally, each U_i computes the master key $K := (t_1^R, t_3^R, \ldots, t_{n-1}^R, \mathsf{pid}_i)$, sets $\mathsf{sk}_i := H(K\|0)$ and $\mathsf{sid}_i := H(K\|1)$.

Fig. 2. Introducing a random oracle $H : \{0,1\}^* \longrightarrow \{0,1\}^k$ in the proposed protocol

– In terms of communication cost the two protocols seem pretty comparable, if in our approach we send all Round 2 messages:
  - Participants in the construction from [BVS07] send in Round 1 a group element and a short bitstring (either a key contribution or a hash value) along with a signature on these values; in Round 2 a session identifier and a bitstring (respectively two for one dedicated participant) are sent along with a signature.
  - Half of the participants in our construction send in Round 1 a group element, the other half a group element and a signature; in Round 2 all participants in "odd positions" send a signature along with a bit-string/group element, and if strong entity authentication is desired, all participants at "even positions" send a signature.

4 Conclusion

The 2-round group key establishment we presented uses Joux's protocol as fundamental building block—instead of a 2-party Diffie-Hellman key establishment. In scenarios where semantic security of the session key and forward security are sufficient, the protocol has the attractive feature that every other participant has to broadcast only one message. For applications where communication cost is high, this seems an attractive feature. Even when strong entity authentication is needed, however, the efficiency of the suggested protocol seems to be quite acceptable, in particular when allowing the use of a random oracle.

References

[ABVS07] Abdalla, M., Bohli, J.-M., González Vasco, M.I., Steinwandt, R.: (Password) Authenticated Key Establishment: From 2-Party to Group. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 499–514. Springer, Heidelberg (2007)

[BCP01] Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange-the dynamic case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290–309. Springer, Heidelberg (2001)

[BVS07] Bohli, J.-M., Gonzalez Vasco, M.I., Steinwandt, R.: Secure group key establishment revisited. International Journal of Information Security 6(4), 243–254 (2007)

[CHL04] Choi, K.Y., Hwang, J.Y., Lee, D.H.: Efficient ID-based Group Key Agreement with Bilinear Maps. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 130–144. Springer, Heidelberg (2004)

[DL08] Desmedt, Y., Lange, T.: Revisiting Pairing Based Group Key Exchange. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 53–68. Springer, Heidelberg (2008)

[DLB07] Desmedt, Y., Lange, T., Burmester, M.: Scalable Authenticated Tree Based Group Key Exchange for Ad-Hoc Groups. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 104–118. Springer, Heidelberg (2007)

[DWGW03] Du, X., Wang, Y., Ge, J., Wang, Y.: An Improved ID-based Authenticated Group Key Agreement Scheme. Cryptology ePrint Archive: Report 2003/260 (December 2003), http://eprint.iacr.org/2003/260/

[Jou04] Joux, A.: A One Round Protocol for Tripartite Diffie Hellman. Journal of Cryptology 17(4), 263–276 (2004)

[KS05] Katz, J., Shin, J.S.: Modeling Insider Attacks on Group Key-Exchange Protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 180–189. ACM, New York (2005)

[KY03] Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)

[VSDSR09] Sree Vivek, S., Shukla, D., Sharmila Deva Selvi, S., Pandu Rangan, C.: Scalable Compilers for Group Key Establishment: Two/Three Party to Group. Cryptology ePrint Archive: Report 2009/115 (2009), http://eprint.iacr.org/2009/115

[ZSM06] Zhou, L., Susilo, W., Mu, Y.: Efficient ID-Based Authenticated Group Key Agreement from Bilinear Pairings. In: Cao, J., Stojmenovic, I., Jia, X., Das, S.K. (eds.) MSN 2006. LNCS, vol. 4325, pp. 521–532. Springer, Heidelberg (2006)

Defeating RSA Multiply-Always and Message Blinding Countermeasures

Marc F. Witteman, Jasper G.J. van Woudenberg, and Federico Menarini

Riscure BV, 2628 XJ Delft, The Netherlands
{witteman,vanwoudenberg,menarini}@riscure.com

Abstract. We introduce a new correlation power attack on RSA's modular exponentiation implementations, defeating both message blinding and multiply-always countermeasures. We analyze the correlation between power measurements of two consecutive modular operations, and use this to efficiently recover individual key bits. Based upon simulation and practical application on a state-of-the-art smart card we show the validity of the attack. Further we demonstrate that cross correlation analysis is efficient on hardware RSA implementations, even in the presence of message blinding and strong hiding countermeasures.

Keywords: side channel analysis, multiply-always, message blinding, RSA, correlation.

1 Introduction

Devices can unintentionally leak data via side channels. For example, timing can reveal parts of a secret, as the time taken by the chip to perform an operation may depend upon branches in its programming controlled by secret data. Further, the device's power consumption and electromagnetic field can be measured and subsequently related to the operations performed to extract secrets.

In practice, power analysis is one of the most fruitful side channels. Generally, the power consumed by a chip in any time interval depends on the total number of bits changed during that time interval. A detailed trace of the power consumed during an execution can be made using a fast digital oscilloscope in combination with special interface equipment. This trace can then be examined for patterns corresponding to the operation performed.

Occasionally, a few traces will already contain sufficient information to extract the secrets from a chip – this is called Simple Power Analysis (SPA). Otherwise, a technique known as Differential Power Analysis (DPA) [2] can be applied: several thousands of traces are made, each one stored along with the data that was input to the chip for its transaction. Statistical analysis is then performed, essentially focusing on intermediate cryptographic data depending on hypothetical values for small parts of the key. By careful application of this technique, the power consumption can be analyzed to reveal the secret keys.

RSA[1] is a public key cryptographic algorithm which is widely applied in various systems. The performance of RSA can be a challenge in devices with little computing power (e.g. smart cards). For this reason several implementation variants were developed that compete in efficiency. Modern smart cards include dedicated cryptographic processors to speed up processing time and can often perform 2048 bit RSA operations in less than a second. Typical applications are in the payment and identification area, where public key cryptography can provide strong authentication combined with flexible key management.

RSA was subject to one of the first side channel timing attacks [12], and also has been subject to SPA attacks [5,6]. DPA attacks target different implementations such as binary exponentiation [7], or RSA in CRT mode [7,11].

RSA implementations use various countermeasures to defend against side channel attacks. DPA and SPA attacks can be prevented with message blinding and the multiply-always exponentiation scheme. Message blinding works by the multiplying the input message with a random, and after exponentiation removing the effect of the random value. Exponent blinding is a similar technique, but applied to the secret exponent. Finally, multiply-always is a countermeasure that aims to prevent SPA by always having a square and a multiply operation for each key bit.

In this paper we propose to use a correlation technique that correlates measured samples only, without considering any intermediate data. In this way the result becomes independent of any data masking or padding scheme, and we can focus solely on the comparison of key bits. It targets the modular operations, whether these are used in a straight RSA or multiply-always implementation.

To demonstrate the strength of this technique we apply it to RSA simulations and an actual RSA hardware implementation that resists side channel analysis using multiple countermeasures.

The remainder of this paper is organized as follows. In the next section we explain some concepts of RSA and review relevant side channel attacks and countermeasures for RSA. Then we introduce the foundation of our new attack and simulate its application. In the final part of the article we report experimental results where we demonstrate that a real-life implementation is vulnerable to this attack.

2 RSA

RSA is based upon modular exponentiation. A signature over a message is computed by raising the message m to a power d modulo n, where the private exponent d and public modulus n together form the private key. The basic RSA implementation variant is called 'binary exponentiation', a scheme where each key bit is processed sequentially, either left-to-right, or right-to-left. With binary exponentiation each key bit is processed with a modular square operation followed by a conditional modular multiplication. The multiplication is only executed when the associated key bit is equal to one. Algorithm 1 shows this algorithm for computing a signature s over a message m with private exponent d: $s = m^d \bmod n$.

Algorithm 1. Binary exponentiation algorithm

```
s := 1 // set signature to initial value
for i from |d|-1 down to 0 do: // left-to-right
    s := s * s mod n // square
    if ($d_i$ = 1), then s := s * m mod n // multiply
return s
```

More sophisticated variants that offer improved performance include Montgomery exponentiation, fixed or sliding window exponentiation, CRT, or combinations thereof. All variants have in common that the key is processed sequentially, and in small quantities.

2.1 Side Channel Attacks on RSA

Many side channel analysis attacks on RSA focus on distinguishing square and multiply operations. With binary exponentiation this may lead to direct retrieval of the key. With the CRT variant also attacks are possible on the reduction and recombination phase, but we will not consider those here.

Exponentiation in smart cards is generally supported by a hardware accelerator, a dedicated area on the chip that efficiently performs modular operations. During the exponentiation the CPU and accelerator alternate activity: in between two modular operations the CPU moves data from and to the cryptographic accelerator. Since the accelerator works on many bits in parallel (e.g. 2048), its power consumption may be higher than that of the CPU that typically works on much smaller data sizes (e.g. 8 or 32 bits). For this reason it may be possible to visually recognize modular operations (SPA). If these operations can be related to respective square or multiply operations, for instance by their interval time, it is possible to directly read out a key.

Differential power analysis attacks on RSA attempt to establish a significant correlation between the samples and hypothetical intermediate data. The attacker would repeatedly investigate correlation in between modular operations.

The attacker may for instance try to establish correlation between the input x and the power traces, just before a modular operation. Any operation that performs a multiplication with x may result in observable correlation, which ultimately also yields the key for a straight binary exponentiation scheme [8].

2.2 Countermeasures

Chip and smart card manufacturers generally implement a mix of countermeasures to prevent, or complicate, side channel attacks. In the context of RSA we discuss three popular countermeasures: hiding, masking or blinding, and multiply-always.

The hiding countermeasure aims at reducing the correlation between the trace samples and the expected activity or intermediate data. This is typically done by adding noise, reducing signal leakage, using a variable clock frequency, and

Algorithm 2. Multiply-always binary exponentiation algorithm

```
s := 1 // set signature to initial value
for i from |d|-1 down to 0 do: // left-to-right
    s := s * s mod n // square
    if ($d_i$ = 1), then
        s := s * m mod n // multiply
    else
        t := s * m mod n // multiply, discard result
return s
```

introducing random process interrupts (random delays). With RSA this implies that individual modular operations can no longer be visually identified.

The masking countermeasure aims at breaking the relation between the power leakage and the intermediate data. The data is masked with random data before the encryption, and unmasked after the encryption. With RSA this is done by message blinding. Message blinding works by generating a secret random number r before each encryption. Then the numbers $m_1 = r^e \bmod n$ and $m_2 = r^{-1} \bmod n$ are computed. Next the input of the encryption is multiplied by m_1 and the output is multiplied by m_2. The first mask makes the input of the encryption unpredictable, while second mask corrects the output to conform to the expected encryption result. Due to this masking, a first order DPA is no longer possible [3].

The multiply-always countermeasure aims at preventing SPA and some DPA attacks by transforming the key dependent operation order into a uniform sequence of operation pairs consisting of a square and a multiply operation. The result of a multiplication is discarded when the pair relates to a key bit set to ‘0’, and propagated when the pair relates to a key bit set to ‘1’; see Algorithm 2. Even when an attacker could distinguish the square and multiply operations, this would not lead to key compromise [4].

3 Multiply-Always Cross Correlation

The multiply-always algorithm always executes squares and multiplication in turn, and is therefore resistant to timing or SPA attacks attempting to recover the key by identifying the different modular operations. However, there is a subtle relation between consecutive squares and multiplies that relate to the key.

There are four types of consecutive modular operations: square|multiply (SM), square|multiply|discard (SMd), multiply|square (MS), and multiply|discard| square (MdS). From the algorithm, it can be seen that SM, SMd, and MS do not share multiplicands. MdS, however, calculates $s \times m$ but does not update s, and then calculates $s \times s$. These two operations therefore share one operand.

We use the observation that MdS and MS differ by their shared operand to perform a correlation attack. Because the operations in MdS share an operand, their power leakage should correlate stronger than MS. We can use this as a distinguisher for individual key bits.

3.1 Operation Correlation

Each modular operation multiplies two operands. If two operations have no operands in common the operations would be trivially independent. However, if two operations share one operand there is a relation between the two. Data dependent leakage of multipliers has been shown in [10], a fact which we exploit below.

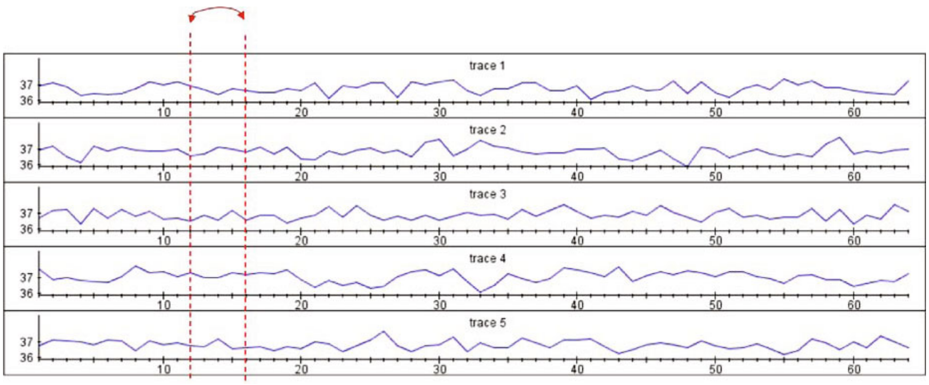


Fig. 1. Cross correlation between vectors of samples

Let us define a matrix of n power traces of length m, which can be plotted as e.g. Figure 1. We also define two vectors $\overline{u}$ and $\overline{v}$, which are columns in the power matrix representing an MdS sequence.

In our model we assume that the values in the matrix are completely represented by the product of the hamming weights of the operands. Elements of the vector $\overline{u}$ are expressed as $u_i = \text{hw}(x_i)\text{hw}(y_i)$, where x_i and y_i are the two operands of the operation expressed by $\overline{u}$. Likewise, elements of the vector $\overline{v}$ are expressed as $v_i = \text{hw}(x_i)\text{hw}(z_i)$, where x_i and z_i are the two operands of the operation expressed by $\overline{v}$.

The Pearson correlation coefficient between $\overline{u}$ and $\overline{v}$ can be computed as:

$$\rho(\overline{u},\overline{v}) = \frac{\sum(u_i v_i) - \frac{\sum u_i \sum v_i}{n}}{\sqrt{\left(\sum\left(u_i^2\right) - \frac{\left(\sum u_i\right)^2}{n}\right)\left(\sum\left(v_i^2\right) - \frac{\left(\sum v_i\right)^2}{n}\right)}}$$

$$= \frac{\sum\left(\text{hw}(x_i)\text{hw}(y_i)\text{hw}(x_i)\text{hw}(z_i)\right) - \frac{\left(\sum \text{hw}(x_i)\text{hw}(y_i)\right)\left(\sum \text{hw}(x_i)\text{hw}(z_i)\right)}{n}}{\sqrt{\left(\sum\left(\text{hw}(x_i)^2\text{hw}(y_i)^2\right) - \frac{\left(\sum \text{hw}(x_i)\text{hw}(y_i)\right)^2}{n}\right)\left(\sum\left(\text{hw}(x_i)^2\text{hw}(z_i)^2\right) - \frac{\left(\sum \text{hw}(x_i)\text{hw}(z_i)\right)^2}{n}\right)}}$$

A hardware multiplication is typically implemented by a shift-and-add mechanism. For a multiplication of x and y, every bit set to 1 will lead to an addition of y. This leaks $\text{hw}(y)$. for the number of '1' bits of x; i.e. $\text{hw}(x)\text{hw}(y)$.

For a random number $0 \leq r \leq 2^k$, the average hamming weight is $\frac{k}{2}$, so the sum of hamming weights for a series of n binary numbers is approximately equal to $\frac{nk}{2}$. For a multiplication with two independent operands, the sum of n multiplied hamming weights will relate to the square of the summed hamming weight, and approximately to $n \times \frac{k}{2} \times \frac{k}{2} = \frac{nk^2}{4}$.

On the other hand, for a square operation – where the operands are fully dependent – the sum of the multiplied hamming weights is different. The sum of the squared hamming weights of all numbers smaller than 2^k is $\sum_{x=0}^{2^k}(\mathrm{hw}(x))^2 = \sum_{i=0}^{k}\binom{k}{i}i^2 = 2^{k-2}k(k+1)$, as demonstrated in [9]. The sum of squared hamming weights for n numbers of k bits therefore will be $\frac{n \times 2^{k-2}k(k+1)}{2^k} = \frac{n(k^2+k)}{4}$.

The correlation coefficient can therefore be approximated as:

$$\frac{n\left(\frac{k}{2}\right)^2\left(\left(\frac{k}{2}\right)^2+\frac{k}{4}\right)\frac{n\left(\frac{k}{2}\right)^2 n\left(\frac{k}{2}\right)^2}{n}}{\sqrt{\left(n\left(\left(\frac{k}{2}\right)^2+\frac{k}{4}\right)\left(\left(\frac{k}{2}\right)^2+\frac{k}{4}\right)\frac{n\left(\frac{k}{2}\right)^2 n\left(\frac{k}{2}\right)^2}{n}\right)\left(n\left(\left(\frac{k}{2}\right)^2+\frac{k}{4}\right)\left(\left(\frac{k}{2}\right)^2+\frac{k}{4}\right)\frac{n\left(\frac{k}{2}\right)^2 n\left(\frac{k}{2}\right)^2}{n}\right)}} =$$

$$\frac{n\left(\frac{k^3}{16}\right)}{n\left(\left(\frac{k}{2}\right)^2+\frac{k}{4}\right)\left(\left(\frac{k}{2}\right)^2+\frac{k}{4}\right)\frac{n\left(\frac{k}{2}\right)^2 n\left(\frac{k}{2}\right)^2}{n}} =$$

$$\frac{n\left(\frac{k^3}{16}\right)}{\frac{n}{16}\left(2k^3+k^2\right)} = \frac{k}{2k+1}$$

For typical large numbers used in RSA computations the correlation coefficient $\frac{k}{2k+1}$ can be approximated by the value $\frac{1}{2}$.

When correlating two statistically independent vectors the correlation converges to zero. So, for sufficiently large vectors we observe a correlation value 0 for modular operations not sharing operands, and a correlation value $\frac{1}{2}$ for operands that share one operand. As such we can distinguish between MdS and MS and therefore attack the individual key bits.

3.2 Application to RSA Simulation

To verify our theoretical correlation predictions, we apply the correlation technique on a simulation of a plain binary exponentiation algorithm, and later show the multiply-always algorithm is not resistant to this type of analysis.

The traces are generated by simulating Hamming weight leakage of the complete operands for an entire run of RSA, for 1000 traces with random input messages.

First, we calculate the Pearson correlation estimate $C = \rho(T(i,x), T(j,x))$ for each pair of columns i and j in our n by m measurement matrix T. In C we can determine whether two modular operations have a particular correlation. We represent C by a graphic where the intensity of each cell represents the strength of correlation. Over the diagonal the correlation is perfect, as these cells represent columns correlated with themselves. All other cells represent correlation between different columns.

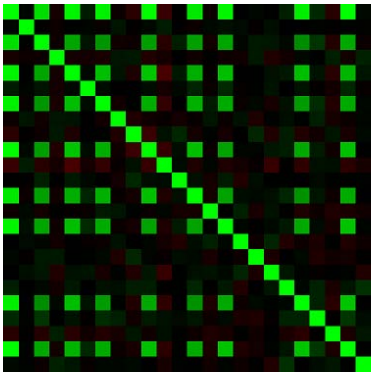


Fig. 2. Cross correlation matrix for simulated plain binary exponentiation

Figure 2 shows C for a simulation of plain RSA binary exponentiation. As all multiplications share the message m as operand, they all show significant correlation. From C we can therefore read out for each operation whether it is a multiplication, and can therefore directly obtain the key.

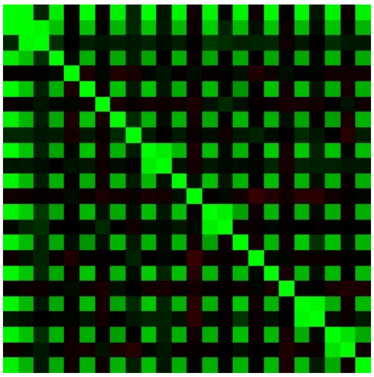


Fig. 3. Cross correlation matrix for a simulated multiply always exponentiation

Next, we simulate a multiply-always RSA exponentiation, and compute the cross correlation matrix. Now each pair of columns represents a square and a subsequent multiplication. In Figure 3 we can still observe the correlation between multiplication operations, but since they are executed always this does not provide any useful information.

However, now we reveal redundant multiplications. A square operation directly following a discarded multiply will work with one of the operands of the discarded multiply, and reveal the previous hidden zero bit by their correlation: we can distinguish between MdS and MS and attack the individual key bits this way.

The simulation shows that the multiply always countermeasure can be defeated. Additionally, since the attack is independent from the processed data it is clear that data blinding does not prevent this attack.

4 Experimental Results

Our correlation attack requires all samples in one vector to relate to one fixed modular operation in an exponentiation scheme, and therefore traces should be compressed such that each modular operation is represented by a single sample. For performing the correlation analysis on actual implementations of RSA we focus on the energy consumed by individual modular operations. First, we acquire side channel traces and compress them such that each modular operation is represented by one sample. When we acquire n traces with m modular operations, we build a matrix of n rows and m columns, where each sample represents the energy consumed by one modular operation.

The energy consumed by the modular operations can be measured by recording either the power consumption, or the electro-magnetic emission of the chip. The first method is often easier to perform, with relatively simple circuitry, while EM measurements require more complex electronics and tuning the optimal measurement position. On the other hand, some chips implement power analysis countermeasures like current flattening, which do not work against EM analysis. In that case EM measurement would be the more attractive acquisition method.

For the attack to work it is important to recognize and distinguish modular operations. In the next two subsections we show how the compression works for simple and complex situations.

4.1 Compressing Visually Identifiable Operations

Modular operations in smart card chips are executed by a cryptographic coprocessor that can perform big number multiplications at high speed. Depending on the key length and the chip technology an individual modular operation may cost 20 − 500 μs. Typically, a cryptographic processor consumes more power than the normal CPU as it switches much more bits in parallel.

Figure 4 shows a power trace taken from a smart card where the modular operations performed by the cryptographic processor can easily be distinguished

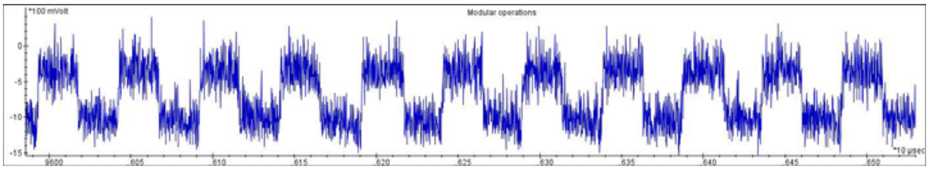


Fig. 4. Modular operations interleaved with CPU activity

from the normal CPU activity. For the analysis we want to retain individual samples representing the energy consumed by each modular operation. To that end, we first apply a low pass filter. Then we set a threshold and compute the surface between the top of the graph and the threshold.

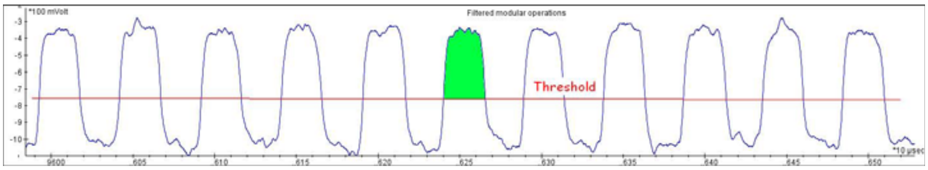


Fig. 5. Simple compression process for modular operations

Figure 5 shows the compression process which is fast and simple. The result of this process is a trace set with one sample per modular operation, which is fit for cross correlation analysis.

4.2 Compressing Hidden Operations

Unfortunately the simple compression process cannot so easily be applied to more advanced smart card chips.

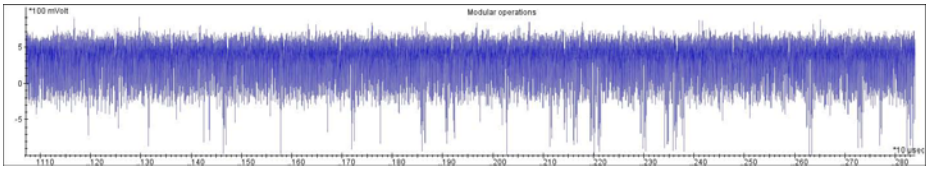


Fig. 6. Modular operations hidden by noise

Figure 6 shows a power trace taken from a chip that uses a strong noise source to hide the energy consumption difference between CPU and cryptographic processor. Even when using a low pass filter it is not possible to identify the modular operations.

In this case another approach can be used to identify and compress the modular operations. First the attacker acquires a set of traces and aligns them at the beginning. Then, an average trace is computed over the trace set. The modular operations close to the alignment point (left side) will become recognizable as a distinct pattern. Operations further away from the alignment point get increasingly blurred because of the jitter of the free running internal clock (see Figure 7).

Next a pattern p representing one modular operation (highlighted in Figure 7) is used to correlate against a trace t to find starting positions for each modular

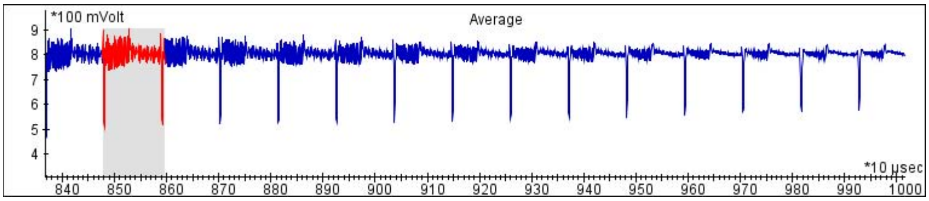


Fig. 7. Average of left aligned modular operations

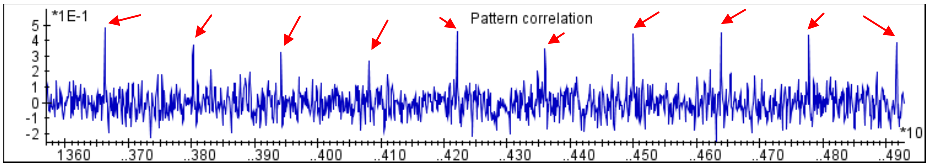


Fig. 8. Pattern correlation with noisy traces

operation. The result is a trace s that indicates for each position s[i] the calculated correlation between vectors p and $t[i] \dots t[i + |p|]$.

The peaks in the correlation graph shown in Figure 8 identify where a selected pattern has the optimal match. This matching mechanism is essential to find the occurrence of patterns that are not on equal distances due to clock jitter or random process interrupts.

Once the modular operations are identified through pattern correlation, they can be compressed by repeatedly averaging $|p|$ samples starting from each detected peak.

4.3 Bounded Pattern Matching

In order to correctly compute the correlation between adjacent modular operations it is essential that all operations are recognized and properly represented by a compressed sample. If some traces would contain errors and incidental modular operations were skipped this would result in a partly shifted trace set, where the correlation values would become distorted.

The pattern matching mechanism introduced in the previous section works well if the distance between the start of two consecutive modular x_0 and x_1 operations is always less than twice the minimal duration (Δx) of the modular operation. This would guarantee that each search period contains exactly one operation.

Figure 9 shows that it is relatively easy to identify the starting point of x_1 by always finding the highest pattern correlation peak within a period between $x_0 + \Delta x$ and $x_0 + 2\Delta x$.

In order to recognize the end of the exponentiation sequence, a minimal correlation threshold value can be used.

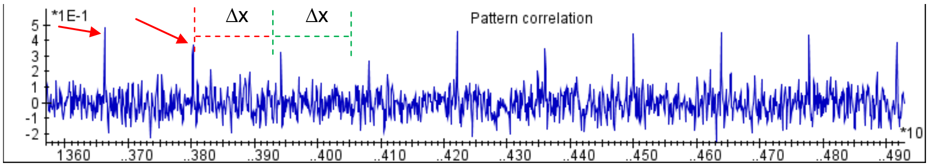


Fig. 9. Bounded pattern matching

4.4 Key Retrieval

We apply the presented attack techniques on a state-of-the-art smart card, for which SPA and DPA attempts on RSA were unsuccessful. We acquire 5000 traces of an RSA 2048 bit signing with a private key. The bounded pattern matching mechanism revealed that 4096 operations were executed, which can be explained by the multiply always countermeasure being used.

The resulting trace set yields 4096 samples per trace. Rather than computing the full cross correlation matrix as in section 3.2, we compute correlation between adjacent columns in the compressed trace set, and plot the correlation between samples and their direct neighbors.

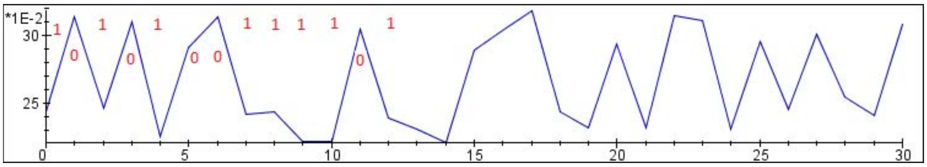


Fig. 10. Cross correlation for adjacent modular operations in a hardware RSA implementation using multiply-always

The resulting graph in Figure 10 allows immediate read out of the entire private key by associating low correlation values with bit value ‘1’, and high correlation values (due to re-use of discarded operand) to value bit ‘0’.

5 Conclusion

We conclude that cross correlation analysis enables a new attack on RSA that attacks the multiply always countermeasure. No intermediates states are calculated, and as such it is oblivious to message blinding. Further, we have shown that the attack works on a modern smart card that produces very noisy power consumption signals.

Although this attack can defeat both message blinding and multiply-always countermeasures, there are known countermeasures that make it substantially harder (noise, floating clocks) or to virtually impossible (exponent blinding) to perform this attack.

References

1. 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)
2. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 388. Springer, Heidelberg (1999)
3. Chaum, D.: Blind Signatures for Untraceable Payments. In: Advances in Cryptology: Proceedings of Crypto 1982, pp. 199–203. Plenum Press, New York (1983)
4. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
5. Novak, R.: SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 252–262. Springer, Heidelberg (2002)
6. Fouque, P.-A., Martinet, G., Poupard, G.: Attacking Unbalanced RSA-CRT Using SPA. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 254–268. Springer, Heidelberg (2003)
7. den Boer, B., Lemke, K., Wicke, G.: A DPA Attack Against the Modular Reduction within a CRT Implementation of RSA. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 228–243. Springer, Heidelberg (2003)
8. Amiel, F., Feix, B., Villegas, K.: Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 110–125. Springer, Heidelberg (2007)
9. Boros, G., Moll, V.: Irresistible Integrals: Symbolics, Analysis and Experiments in the Evaluation of Integrals. Cambridge University Press, Cambridge (2004)
10. Walter, C., Samyde, D.: Data Dependent Power Use in Multipliers. In: Proc. 17th IEEE Symposium on Computer Arithmetic. IEEE Press, Los Alamitos (2005)
11. Witteman, M.: A DPA attack on RSA in CRT mode. Riscure Technical Report, http://www.riscure.com/fileadmin/images/Docs/DPA_attack_on_RSA_in_CRT_mode.pdf
12. 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)

Cryptanalysis of CLEFIA Using Differential Methods with Cache Trace Patterns

Chester Rebeiro and Debdeep Mukhopadhyay

Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India
{chester,debdeep}@cse.iitkgp.ernet.in

Abstract. In this paper we use a combination of differential techniques and cache traces to attack the block cipher CLEFIA in less than 2^{14} encryptions on an embedded processor with a cache line size of 32 bytes. The attack is evaluated on an implementation of CLEFIA on the PowerPC processor present in the SASEBO side channel attack evaluation board. The paper shows that although obtaining cache access patterns from the power consumption of the device may be difficult due to the non-blocking cache architectures of modern processors, still the cache trace has a distinct signature on the power profiles. Experimental results have been presented to show that the power consumption of the device reveal the cache access patterns, which are then used to obtain the CLEFIA key. Further, a simple low overhead countermeasure is implemented that is guaranteed to prevent cache attacks.

1 Introduction

On microprocessors with cache memory, a cache miss takes more power and time than a cache hit. A class of cache attacks, known as *cache-trace attacks*[1,2,4,8,21] monitor these differences to gain secret information about a cryptographic algorithm. This form of cache-attacks is the most powerful in terms of the number of encryptions required. Yet, a naive cache-trace attack on Sony's block cipher CLEFIA[16] is estimated to have a complexity of more than 2^{40} encryptions[11]. In this paper we demonstrate a cache-trace attack which uses the differential properties of CLEFIA to reduce the attack complexity to 2^{14} encryptions on the PowerPC-405 processor.

Most published cache-trace attacks target AES. Bertoni et. al. showed that cache traces are manifested in the power profiles and reveal secret information about the cryptographic algorithm being executed [4]. A first round cache-trace attack on AES was done in [8] and was extended to a two round attack in [2]. A final round attack on AES was also described in [2].

All cache-trace attacks target structures in the cipher such as in Figure 1. The figure shows two accesses to table S with indices $(in_0 \oplus k_0)$ and $(in_1 \oplus k_1)$. In an ideal cache, a cache hit occurs when $(in_0 \oplus k_0) = (in_1 \oplus k_1)$. This reveals information about the ex-or of the key bits: $(k_0 \oplus k_1) = (in_0 \oplus in_1)$. In a real cache however, a cache miss results in a block of data being loaded from memory.

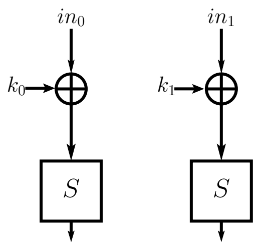


Fig. 1. Look-up Structure in AES

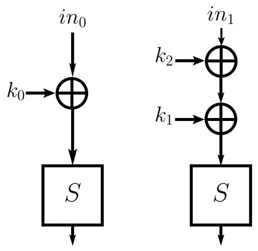


Fig. 2. Look-up Structure in CLEFIA

Hence cache attacks cannot distinguish between locations in the same block. If the cache's line size is l bytes and the size of each entry in the table is b bytes, then $log_2(l/b)$ lower bits of the ex-or cannot be determined. In reality therefore, $\langle k_0 \oplus k_1 \rangle = \langle in_0 \oplus in_1 \rangle$, where $\langle \cdot \rangle$ is the most significant bits.

In CLEFIA, some of the table accesses have a form depicted in Figure 2. A straight forward adaptation of existing cache-trace attacks is capable of revealing only the value of $\langle k_0 \oplus k_1 \oplus k_2 \rangle$, thus has more ambiguity about the key value. This makes cache-trace attacks on CLEFIA more complex than attacking AES. Zhao and Wang have shown a cache-trace attack on CLEFIA[21] under a strong assumption of misaligned tables. Misalignment of data generally does not happen unless it is forced by the programmer or the program is optimized for space. Our cache-trace attack on CLEFIA considers the standard structure of a program where the tables are aligned to cache line boundaries.

Except for [4], none of the other publications provide experimental evidence of the results. In [4] too, results provided were from simulations. A simulated experiment cannot replicate all physical parameters of a real environment. Our attack on the other hand is demonstrated on actual hardware. We use the SASEBO side channel attack evaluation board[1] as the test platform. We show that advanced features in the cache architecture such as non-blocking accesses, make interpretation of power profiles difficult. We then present a method by which the cache access pattern can still be extracted from the profile.

A countermeasure is proposed that would guarantee protection against all forms of cache attacks at a lower performance penalty. The countermeasure is based on the fact that ciphers using tables that fit in a single cache line is protected from cache attacks. The proposed countermeasure is implemented for CLEFIA and its performance analyzed.

The outline of the paper is as follows: Section 2 has a brief description of the block cipher CLEFIA. Section 3 discusses cache attacks based on differential properties of the cipher and presents the principle of the attack. Section 4 presents the attack on CLEFIA while Section 5 provides experimental results. A low-overhead countermeasure for protecting CLEFIA against cache attacks is presented in Section 6. The work is concluded in Section 7.

[1] `http://www.rcis.aist.go.jp/special/SASEBO/index-en.html`

2 The CLEFIA Block Cipher

CLEFIA is a 128 bit block cipher with a generalized Feistel structure. The specification [16] defines three key lengths of 128, 192, and 256 bits. For brevity, this paper considers 128 bit keys though the results are valid for the other key sizes also. The structure of CLEFIA is shown in Figure 3. The input has 16 bytes, P_0 to P_{15}, grouped into four 4 byte words. There are 18 rounds, and in each round, the first and third words are fed into nonlinear functions $F0$ and $F1$ respectively. The output of $F0$ and $F1$ are ex-ored with the second and fourth words. Additionally, the second and fourth words are also whitened at the beginning and end of the encryption. The F functions take 4 input bytes and 4 round keys. The non-linearity in the F functions are due to two 256 element sboxes $S0$ and $S1$. Matrices $M0$ and $M1$ diffuse the outputs of the sboxes. They are defined as follows:

$$M0 = \begin{pmatrix} 1 & 2 & 4 & 6 \\ 2 & 1 & 6 & 4 \\ 4 & 6 & 1 & 2 \\ 6 & 4 & 2 & 1 \end{pmatrix} \qquad M1 = \begin{pmatrix} 1 & 8 & 2 & A \\ 8 & 1 & A & 2 \\ 2 & A & 1 & 8 \\ A & 2 & 8 & 1 \end{pmatrix} \tag{1}$$

The design of the sboxes $S0$ and $S1$ differ. $S0$ is composed of four sboxes $SS0$, $SS1$, $SS2$, and $SS3$; each of 16 bytes. The output of $S0$ is given by :

$$\begin{aligned} \beta_l &= SS2[SS0[\alpha_l] \oplus 2 \cdot SS1[\alpha_h]] \\ \beta_h &= SS3[SS1[\alpha_h] \oplus 2 \cdot SS0[\alpha_l]], \end{aligned} \tag{2}$$

where $\beta = (\beta_h|\beta_l)$, $\alpha = (\alpha_h|\alpha_l)$, and $\beta = S0[\alpha]$. The output of $S1$ for the input byte α is given by $g((f(\alpha))^{-1})$, where g and f are affine transforms and the inverse is found in the field $GF(2^8)$.

The CLEFIA encryption has 4 whitening keys WK_0, WK_1, WK_2, and WK_3; and 36 round keys $RK_0, \cdots, RK_{35}$. Key expansion is a two step process. First a 128 bit intermediate key L is generated from the secret key K using a GFN function [16]. From this the round keys and whitening keys are generated as shown below:

```
Step 1: $WK_0|WK_1|WK_2|WK_3 \leftarrow K$
Step 2: For $i \leftarrow 0$ to 8
            $T \leftarrow L \oplus (CON_{24+4i}|CON_{24+4i+1}|CON_{24+4i+2}|CON_{24+4i+3})$
            $L \leftarrow \Sigma(L)$
            if $i$ is odd: $T \leftarrow T \oplus K$
            $RK4i|RK4i+1|RK4i+2|RK4i+3 \leftarrow T$
```

The function Σ, known as the *double swap* function, rearranges the bits of L.

$$\Sigma(L) \leftarrow L_{(7\cdots63)}|L_{(121\cdots127)}|L_{(0\cdots6)}|L_{(64\cdots120)} \tag{3}$$

From the structure of CLEFIA it is obvious that the knowledge of any set of 4 round keys $(RK4i, RK4i+1, RK4i+2, RK4i+3)$, where $i \bmod 2 = 0$, is sufficient to revert the key expansion process to obtain the secret key. In the attack on CLEFIA described in this paper, round keys $RK0$, $RK1$, $RK2$, and $RK3$ are determined from which K is reversed.

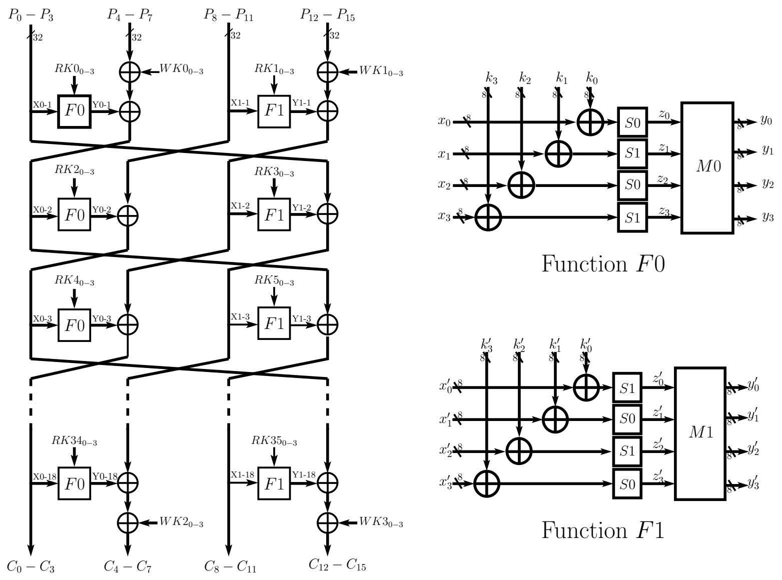


Fig. 3. CLEFIA Block Diagram

3 Differential Cache-Trace Attacks

Combining conventional cryptanalytic techniques with side channel attacks has been shown to be a powerful cryptanalytic tool. Most published works use the algebraic properties of the cipher to reduce the attack's complexity [6,12,13]. In this work we use the differential properties of the cipher instead. This section presents the general principle of what we term as a *differential cache-trace attack*.

Consider the Feistel structure in Figure 4 with two inputs (in_0 and in_1) and two keys k_0 and k_1 of size n bits. Suppose that in_1 is chosen in such a way that the second sbox access collides with the first, then,

$$\langle in_0 \oplus k_0 \rangle = \langle S[in_0 \oplus k_0] \oplus in_1 \oplus k_1 \rangle \tag{4}$$

From this the uncertainty of the keys k_0 and k_1 reduces from 2^{2n} to $2^{n+\delta}$, where 2^{δ} is the number of sbox elements that share a cache line. To further reduce this uncertainty another collision is considered with a different set of inputs in'_0 and in'_1. Due to this,

$$\langle in'_0 \oplus k_0 \rangle = \langle S[in'_0 \oplus k_0] \oplus in'_1 \oplus k_1 \rangle \tag{5}$$

Combining Equations (4) and (5) we obtain,

$$\langle in_0 \oplus in_1 \oplus in'_0 \oplus in'_1 \rangle = \langle S[in_0 \oplus k_0] \oplus S[in'_0 \oplus k_0] \rangle \tag{6}$$

The uncertainty of the key now depends on the differential properties of the sbox. Let f_{avg} be the average number of keys that would satisfy a given input

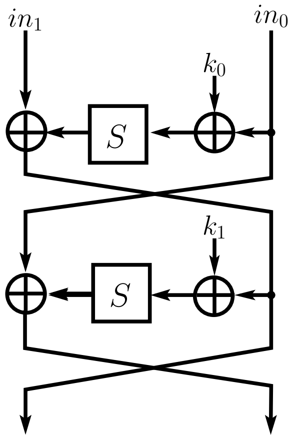


Fig. 4. Two Round Feistel Structure

difference-output difference pair for the sbox S. Then, the expected number of candidate keys is given by

$$N = 2^{\delta} \cdot f_{avg}$$

This set of candidate keys can be reduced by repeating the experiment with different input pairs and obtaining the intersection between all the sets. If the number of times the experiment is repeated is r then,

$$\text{Expected number of candidate keys after } r \text{ repetitions} = \frac{N^r}{2^{n(r-1)}}$$

Generally f_{avg} is small. For $S0$ and $S1$ of CLEFIA, f_{avg} was found to be 1.28 and 1.007 respectively. So, even with $r = 1$, the uncertainty of the key is much lesser than the naive cache-trace attack.

4 Adapting the Differential Cache-Trace Attack to CLEFIA

Our attack on CLEFIA comprises of three steps. First $RK0$ and $RK1$ are determined, then $WK0 \oplus RK2$ and $WK1 \oplus RK3$, and finally $RK4$ and $RK5$. With these round keys, CLEFIA's key expansion algorithm is used to obtain 57 bits of $(RK2|RK3)$. In all, obtaining the 121 bits of the round keys $RK0$, $RK1$, $RK2$, and $RK3$ requires 2^{14} encryptions.

In the attack, we have assumed that there are 8 elements that share a cache line. Therefore, while accessing the look-up table of CLEFIA consisting of 256 elements, the cache traces do not distinguish between the lowest 3 bits of the index. Due to this, while searching for a cache hit, it is sufficient to keep the lowest 3 bits of the plaintext fixed. Thus varying a plaintext byte to find a cache hit would require $2^{8-3} = 2^5$ encryptions. In the remaining part of the section the symbol $\langle \rangle$ signifies the 5 most significant bits of a byte.

4.1 Differential Properties of CLEFIA's F Functions

Our cache-trace attack uses the following observations on the F functions:

– Matrices $M0$ and $M1$ in the F functions do not attain complete diffusion in all bits of the output. If the 5 most significant bits ($MSBs$) of the input of each byte of the matrices $M0$ and $M1$ are known then few bits of the output can be computed (see Figure 3). In particular three $MSBs$ of each byte in $M0$'s output and two $MSBs$ of each byte in $M1$'s output are computable. Since $M0$ and $M1$ are self inverting matrices, the inverse of the above statement also holds. That is, given 5 $MSBs$ of each byte of the output, 3 $MSBs$ of the input in $M0$ and 2 $MSBs$ of the input in $M1$ is computable.
– For a pair of inputs, the non-linearity in the sboxes causes several (60% in $S0$ and 50% in $S1$) of the possible input difference-output difference combinations to be invalid. Additionally, for a valid combination, $S0$ has 1.28 choices on average for the inputs to the sbox, while $S1$ has 1.007.

If the inputs to an sbox is $(p_i \oplus k)$ and $(p_{ii} \oplus k)$, then the ex-or difference is $(p_i \oplus p_{ii})$. This is known. Additionally, the trace attack reveals three bit differences per byte of the output of each sbox of $F0$. For the remaining 5 bits of each output, there are 32 possible input difference-output differences for each sbox resulting in an average of 32 possible key (k) candidates for each byte. Similarly there are about 64 possible choices for each key byte in $F1$. We now show how these differential properties of CLEFIA are used in the recovery of the round keys.

4.2 Determining $RK0$ and $RK1$

The sbox accesses in the first round of CLEFIA have a structure (Figure 1) favorable for cache attacks. The equations for the indices to the tables in the first round is given by:

$$\begin{aligned} I1_{s0}^{0} &= P_0 \oplus RK0_0 & I1_{s0}^{1} &= P_2 \oplus RK0_2 \\ I1_{s0}^{2} &= P_9 \oplus RK1_1 & I1_{s0}^{3} &= P_{11} \oplus RK1_3 \\ \\ I1_{s1}^{0} &= P_1 \oplus RK0_1 & I1_{s1}^{1} &= P_3 \oplus RK0_3 \\ I1_{s1}^{2} &= P_8 \oplus RK1_0 & I1_{s1}^{3} &= P_{10} \oplus RK1_2, \end{aligned} \tag{7}$$

where $I\alpha_{s\beta}^{i}$ denotes the index to the $(i+1)^{th}$ access to table $s\beta$ in round α.

If we make the assumption that no part of the table is present in cache before the start of encryption, then the first access to each table, ie. $I1_{s0}^{0}$ and $I1_{s1}^{0}$, results in cache misses. Keeping P_0 and P_1 fixed and by varying P_2 first and then P_3, 2 cache hits in $F0$ of round 1 can be obtained for some values of P_2 and P_3. Keeping these values, and varying P_8, P_9, P_{10}, and P_{11} independently, it is possible to obtain a maximum of 6 cache hits in the first encryption round. Such a state of the cipher is called a 1-*round colliding state.*

In the second round, the indices to the tables $S0$ and $S1$ in $F0$ are given by equations in (8), where $P_{(0\cdots3)}$ indicates the concatenation of P_0, P_1, P_2, and P_3.

$$\begin{aligned}
I2^0_{s0} &= P_4 \oplus WK0_0 \oplus F0(RK0, P_{(0\cdots3)})_0 \oplus RK2_0 \\
I2^0_{s1} &= P_5 \oplus WK0_1 \oplus F0(RK0, P_{(0\cdots3)})_1 \oplus RK2_1 \\
I2^1_{s0} &= P_6 \oplus WK0_2 \oplus F0(RK0, P_{(0\cdots3)})_2 \oplus RK2_2 \\
I2^1_{s1} &= P_7 \oplus WK0_3 \oplus F0(RK0, P_{(0\cdots3)})_3 \oplus RK2_3
\end{aligned} \tag{8}$$

Starting from the 1-round colliding state, four cache hits can be forced in $F0$ of round two by varying, independently, the $MSBs$ of P_4, P_5, P_6, and P_7 in an order such that P_4 is varied before P_6, and P_5 is varied before P_7. This results in a total of 5 cache hits in table $S0$ (3 in the first round and 2 in the second). The $MSBs$ of the indices to the table are all the same, ie. $\langle I1^0_{s0}\rangle = \langle I1^1_{s0}\rangle = \langle I2^0_{s0}\rangle = \langle I2^1_{s0}\rangle$. We therefore get the following equalities:

$$\begin{aligned}
\langle P_0 \oplus P_4\rangle &= \langle F0(RK0, P_{(0\cdots3)})_0 \oplus RK0_0 \oplus WK0_0 \oplus RK2_0\rangle \\
\langle P_2 \oplus P_6\rangle &= \langle F0(RK0, P_{(0\cdots3)})_2 \oplus RK0_2 \oplus WK0_2 \oplus RK2_2\rangle
\end{aligned} \tag{9}$$

Similarly the 5 cache hits in table $S1$ result in the following equalities:

$$\begin{aligned}
\langle P_1 \oplus P_5\rangle &= \langle F0(RK0, P_{(0\cdots3)})_1 \oplus RK0_1 \oplus WK0_1 \oplus RK2_1\rangle \\
\langle P_3 \oplus P_7\rangle &= \langle F0(RK0, P_{(0\cdots3)})_3 \oplus RK0_3 \oplus WK0_3 \oplus RK2_3\rangle
\end{aligned} \tag{10}$$

For another plaintext Q, with $Q_0 \neq P_0$ and $Q_1 \neq P_1$, equations similar to (9) and (10) can be obtained by tracing cache collisions in the first and second rounds. These are shown in (11), where $0 \leq i < 4$.

$$\langle Q_i \oplus Q_{4+i}\rangle = \langle F0(RK0, Q_{(0\cdots3)})_i \oplus RK0_i \oplus WK0_i \oplus RK2_i\rangle \tag{11}$$

From (9),(10), and (11), and the fact that $\langle P_0 \oplus P_2 \oplus P_4 \oplus P_6\rangle = \langle Q_0 \oplus Q_2 \oplus Q_4 \oplus Q_6\rangle$, and $\langle P_1 \oplus P_3 \oplus P_5 \oplus P_7\rangle = \langle Q_1 \oplus Q_3 \oplus Q_5 \oplus Q_7\rangle$ the following equations are generated:

$$\begin{aligned}
\langle P_0 \oplus P_4 \oplus Q_0 \oplus Q_4\rangle &= \langle F0(RK0, P_{(0\cdots3)})_0 \oplus F0(RK0, Q_{(0\cdots3)})_0\rangle \\
\langle P_1 \oplus P_5 \oplus Q_1 \oplus Q_5\rangle &= \langle F0(RK0, P_{(0\cdots3)})_1 \oplus F0(RK0, Q_{(0\cdots3)})_1\rangle \\
\langle P_2 \oplus P_6 \oplus Q_2 \oplus Q_6\rangle &= \langle F0(RK0, P_{(0\cdots3)})_2 \oplus F0(RK0, Q_{(0\cdots3)})_2\rangle \\
\langle P_3 \oplus P_7 \oplus Q_3 \oplus Q_7\rangle &= \langle F0(RK0, P_{(0\cdots3)})_3 \oplus F0(RK0, Q_{(0\cdots3)})_3\rangle
\end{aligned} \tag{12}$$

It is now possible to apply the differential properties of the F functions to derive possible key candidates. Considering just two blocks of plaintexts, P and Q, would result in 32 candidate key values (on average) for each byte of $RK0$. In order to identify a single key with probability greater than 1/2, cache hits in 4 plaintexts must be considered, and the intersection between all possible candidate key sets must be found.

In a similar way round key $RK1$ can be determined by analyzing cache hits in $F1$. The set of equations that should satisfy $RK1$ is shown below, where $0 \leq i < 4$.

$$\langle P_{8+i} \oplus P_{12+i} \oplus Q_{8+i} \oplus Q_{12+i}\rangle = \langle F1(RK0, P_{(8\cdots11)})_i \oplus F1(RK0, Q_{(8\cdots11)})_i\rangle$$

Due to the matrix $M1$, which only reveals two bits of the difference of outputs in each sbox, 6 plaintext blocks are required instead of 4.

Analysis: To determine $RK0$ requires 4 plaintext blocks to be found which would result in 14 cache hits in the first two rounds. Obtaining each block requires 7 iterations, with each iteration requiring 2^5 encryptions. Thus $4 \cdot 7 \cdot 2^5$ encryptions are required. By a similar argument, determining $RK1$ requires $6 \cdot 7 \cdot 2^5$ encryptions.

4.3 Determining $WK0 \oplus RK2$ and $WK1 \oplus RK3$

A cache hit in the first table accesses in the third round can be found by varying byte P_8 for $S0$ (and P_9 for $S1$) (Figure 3). The cause of this cache hit could be collisions with any of the 8 previous accesses to that table. To reduce the number of 'causes' that result in cache hits, the plaintext bytes are chosen in a way such that the first two rounds have only one cache miss in each table (ie. the first accesses). Such a state of the cipher is called the 2*-round colliding state.* The 2-round colliding state has 14 cache hits in the first two round. Such a state is obtained by first obtaining the 1-round colliding state and then varying bytes P_4 to P_7 and P_{12} to P_{15} independently until 8 cache hits in the second round are also obtained.

The third round first access cache hit caused by changing P_8 (or P_9) starting from the 2-round colliding state has 3 causes. The first two causes are due to collisions with $S0$ table accesses in $F1$ in round two. The third cause is due to collisions with $S0$ accesses in $F0$; this is of interest and is estimated to occur once every 3 collisions. The uninteresting cache hits due to the first two reasons are caused by the changing P_8, which in turn changes $Y1$-1 (Figure 3). On obtaining a cache hit in the first table access in the third round, it is required to identify whether the hit is interesting. This is done by changing the value of P_{12} (or P_{13}) and re-doing the encryption. If a cache hit still occurs in round 3, then with significant probability it is of interest.

Similar cache hits for the other $F0$ table accesses in round 3 can be obtained. With these collisions the following equalities are satisfied for a pair of plaintexts P and Q.

$$\begin{aligned}\langle P_i \oplus Q_i \oplus P_{8+i} \oplus Q_{8+i}\rangle =& \langle F0(RK2, WK0 \oplus P_{(4\cdots7)} \oplus Y0\text{-}1^P)_i \\ & \oplus F0(RK2, WK0 \oplus Q_{(4\cdots7)} \oplus Y0\text{-}1^Q)_i\rangle,\end{aligned}$$

where $0 \leq i < 4$, and $Y0$-1 is as defined in Figure 3. $Y0$-1 can be computed using the $RK0$ found in the first step of the attack. Differential properties of the $F0$ function and 4 plaintext blocks can be used to completely determine $RK2 \oplus WK0$.

Analysis: Finding $WK0 \oplus RK2$, requires plaintext bytes P_4 to P_7 to be set up in such a way that there are 4 collisions in the second round. Setting up these collisions requires $4 \cdot 2^5$ encryptions. Next, to find a single byte of $(WK0 \oplus RK2)$, a third round cache hit is required. It takes 2^5 encryptions to find a hit and an additional 2^5 encryptions to determine if the hit is interesting. In all, finding a single plaintext that causes the required cache hits in the third round requires $4 \cdot 2^5 + 4 \cdot 2^6$ encryptions. Four such plaintexts need to be found, therefore the encryptions required to obtain all bytes of $(WK0 \oplus RK2)$ is less than 2^{11}.

Finding WK1 ⊕ RK3: In a similar manner $RK3 \oplus WK1$ can be found in less than 2^{12} encryptions by considering collisions in $F1$ in round 3 and varying plaintext bytes $P_{(0 \cdots 3)}$. The difference equations that is to be satisfied is given by the following (where $0 \leq i < 4$):

$$\begin{aligned}\langle P_i \oplus Q_i \oplus P_{8+i} \oplus Q_{8+i} \rangle =& \langle F1(RK3, WK1 \oplus P_{(12 \cdots 15)} \oplus Y1\text{-}1^P)_i \\ & \oplus F1(RK3, WK1 \oplus Q_{(12 \cdots 15)} \oplus Y1\text{-}1^Q)_i \rangle\end{aligned}$$

4.4 Determining *RK*4 and *RK*5

$RK4$ and $RK5$ can be determined in 2^{13} encryptions using the same idea as the second step of the attack. To find $RK4$, a 2-round colliding state is first obtained from which cache hits in $F0$ of the fourth round is forced by varying the 4th word of the plaintext. $RK4$ can be determined from this using the equations:

$$\begin{aligned}&\langle P_i \oplus Q_i \oplus P_{12+i} \oplus Q_{12+i} \oplus Y1\text{-}1_i^P \oplus Y1\text{-}1_i^Q \rangle \\ &\qquad = \langle F0(RK4, X0\text{-}3^P)_i \oplus F0(RK4, X0\text{-}3^Q)_i \rangle,\end{aligned}$$

where $Y1\text{-}1^P$, $Y1\text{-}1^Q$, $X0\text{-}3^P$, and $X0\text{-}3^Q$ are computed from previously determined round keys and $0 \leq i < 4$. Similarly, $RK5$ is determined by cache hits in $F1$ in the 4th round. The equalities for determining $RK5$ are:

$$\begin{aligned}&\langle P_{4+i} \oplus Q_{4+i} \oplus P_{8+i} \oplus Q_{8+i} \oplus Y0\text{-}1_i^P \oplus Y0\text{-}1_i^Q \rangle \\ &\qquad = \langle F1(RK5, X1\text{-}3^P)_i \oplus F1(RK5, X1\text{-}3^Q)_i \rangle\end{aligned}$$

4.5 Determining *RK*2 and *RK*3

In the key expansion algorithm, if $i = 0$ then $T = (RK0|RK1|RK2|RK3)$, and $T = L \oplus (CON_{24}|CON_{25}|CON_{26}|CON_{27})$. Sixty four bits of the key dependent constant L can be computed using the values of $RK0$ and $RK1$, which were determined in the first step of the attack.

$$(L_0|L_1) = (RK0|RK1) \oplus (CON_{24}|CON_{25}) \tag{13}$$

The double swap operation on L places 57 known bits of L in the lower bit positions. This is given by $L'_{(0 \cdots 56)} = L_{(7 \cdots 63)}$. Again, in the key expansion algorithm, if $i = 1$, then $T = (RK4|RK5|RK6|RK7)$. This is represented as $T = L' \oplus (CON_{28}|CON_{29}|CON_{30}|CON_{31}) \oplus (WK0|WK1|WK2|WK3)$. Therefore,

$$WK0|WK1_{(0 \cdots 24)} = L'_{(0 \cdots 56)} \oplus (CON_{28}|CON_{29(0 \cdots 24)}) \oplus (RK4|RK5) \tag{14}$$

Using $RK4$ and $RK5$, which were determined in the third step of the attack, the whole of $WK0$ and 25 bits of $WK1$ can be determined. Then the result from the second step of the attack is used to obtain 57 bits of $(RK2|RK3)$. Thus 121 out of the 128 bits of $(RK0|RK1|RK2|RK3)$ is retrieved.

5 Experimental Results

We implemented the entire attack in two steps. First the power consumption of the attacked implementation is obtained and a binary vector of the cache access patterns is derived from the profile. A 1 in the binary vector denotes a cache hit while a 0 denotes a cache miss. The vector is fed into a key extraction program which reduces the key space using steps mentioned in the above algorithm. In this section we describe the experimental setup used to obtain the power traces, and discuss how the cache access patterns can be derived from the traces.

Test Platform: The Xilinx $XC2VP30$ FPGA[19] on the SASEBO side channel attack evaluation board[14] was used for the experimentation. The FPGA has a $300MHz$ PowerPC-405 core and $16KB$ two way set associative data cache. $32KB$ of the FPGA's block RAM was configured as the processor's memory. On a cache miss, eight 32 bit words are loaded from memory into cache by means of a $100MHz$ PLB bus. Sony's reference code for CLEFIA[2] was used in the attack with each sbox element occupying 4 bytes. Before the start of each encryption, the entire cache is cleaned using the *XCache_FlushDCacheLine* library function[18]. Power measurements are taken using a current probe across a 1Ω resistor connected in the FPGA's supply line.

5.1 Extracting Cache Trace Patterns from Power Profiles

Unlike [4], where a cache miss is easily evident from the simulations, in actual hardware a single cache miss is not easily distinguishable from a cache hit. Moreover, due to the non-blocking cache in PowerPC [20], it is difficult to pinpoint the exact memory access that is causing the miss to occur. The non-blocking cache would allow other memory accesses to proceed in parallel while the miss is being serviced, provided that there are no data dependencies. Due to the structure of CLEFIA, the sbox accesses within a single round are not interdependent, hence the accesses can be performed in any order. However, the sbox accesses depends on previous round results, therefore the accesses cannot be done until the previous rounds have completed. What this means is that the power profile for a single round cannot have accesses from any other round, in spite of the non-blocking cache. Hence the power profile of a single round of the cipher forms a characteristic signature of the cache access pattern for that round.

The first round of CLEFIA has 8 sbox accesses of which the first two accesses are compulsory misses. The remaining six accesses results in 64 different power profile signatures depending on whether an access results in a hit or a miss.

[2] Version 1.0 `(http://www.sony.net/clefia/.)`

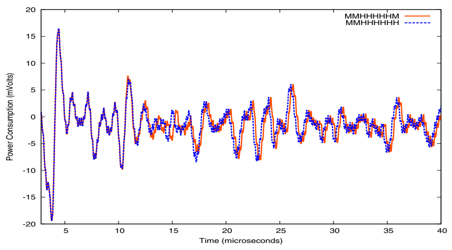


Fig. 5. First Round Power Profile for CLEFIA

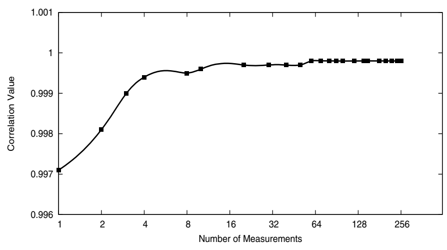


Fig. 6. Correlation Coefficient vs Number of Measurements

Figure 5, shows two power profiles for the first round of CLEFIA. The first profile is for an encryption that has the maximum of six hits in the first round ($MMHHHHHH$), while the second is for an encryption in which the eighth access is a cache miss ($MMHHHHHM$). It may be noted that, although there is just a single difference in the hit-miss pattern between the two encryptions, the power consumption profile carries a distinct signature.

For the attack, we initially have a learning phase in which all the 64 possible signature profiles are collected. During the attack phase, the first round power profile of the unknown key is correlated against the 64 signature profiles. The pattern with the highest correlation is taken as the cache access pattern for the round. It was found that power profiles with identical cache access patterns have a correlation coefficient close to 1, while the correlation with a different cache access pattern have values around 0.8. With identical cache access profiles, the high value of correlation coefficient is obtained with just one measurement. Figure 6, shows that a correlation of 0.997 is obtained from a single measurement. To further strengthen the result, more measurements are made and the average power profile is determined. With 8 measurements the correlation value of 0.9995 is obtained. However, the extra measurements increase the total number of encryptions required for the attack from 2^{14} to 2^{17}.

The proposed attack requires cache access patterns of the second, third, and fourth rounds to be known. This is done in a similar manner by first obtaining all possible power profile signatures for these rounds. Since there are 8 table accesses in each round, and each access could be a hit or a miss, therefore a round has 256 different profile signatures instead of 64.

6 Countermeasures against Cache Attacks

Countermeasures for cache attacks are classified depending on how they are applied. The countermeasures can be applied in the hardware, operating system, algorithm, or in the implementation. In this paper we restrict our discussion to countermeasures applied to the implementation of algorithms as they are the most applicable to existing systems. Countermeasures also differ in the degree of security against cache attacks and in the overhead on the performance.

A common countermeasure is to eliminate key related table look-ups by either warming[10] or preloading tables into the cache. This increases the difficulty of attacks but does not completely prevent them[3]. Security is also increased by adding noise to the encryption profile by dummy accesses or random delays during encryption. Another potential countermeasure is to dynamically mix the contents of the table entries or have redundant or alternate look-up tables[17]. Masking of table accesses has also been proposed as a countermeasure, though not very effective. Although these countermeasures increase security, they do not guarantee complete prevention against cache attacks.

The most obvious way of guaranteeing complete protection against cache attacks is to either disable cache memory or implement algorithms without any key related table look-ups. The latter is done by replacing table look-ups with logical equations for the sbox. Although bit-slicing[5] provides an efficient way to implement block ciphers without table look-ups, its use is restricted to non-feedback and non-chaining modes of encryption, which is not the most secured way of doing the encryption. For the standard encrypting modes, the available techniques of disabling cache and non-table look up implementations have tremendous overhead on the performance. For example, disabling the cache is known to deteriorate performance up to 100 times[17], while 180 logical operations are required to implement a single AES sbox[7].

Proposed Countermeasure: The proposed countermeasure is based on the fact that if the entire sbox were to fit in a single cache line, then the first access would load the complete sbox into cache and all other accesses would result in cache hits. This simple technique would not only guarantee complete security against cache attacks but also have a comparatively lower performance overhead. The countermeasure would work on existing ciphers but for the fact that the size of sboxes generally exceeds that of a cache line, therefore several cache lines would be required to completely store the sbox. We therefore propose to use a combination of logical equations and look-up tables as a countermeasure. The look-up tables will be made to fit in one cache line, and will be used to reduce the number of operations required in the sbox equations. This countermeasure can be applied to any sbox which can be represented by logical equations.

6.1 Cache Attack Countermeasure for CLEFIA

For demonstrating the countermeasure we take a cache line size of 32 bytes. This is the size of a cache line in PowerPC. The sbox $S0$ of CLEFIA is built using 4 smaller tables $SS0$, $SS1$, $SS2$, and $SS3$ (Equation 2). Each of the small tables occupy 8 bytes in memory. Therefore the four tables would occupy the entire cache line. Implementing $S0$ using these 4 tables is simple. In all the small tables save 129 operations.

Sbox $S1$ is constructed using an inverse in the field $GF(2^8)$ with the irreducible polynomial $x^8 + x^4 + x^3 + x^2 + 1$. The inverse computation is preceded and succeeded by affine transformations. Composite field isomorphism is an efficient method to implement such sboxes[9]. Although this method is well known,

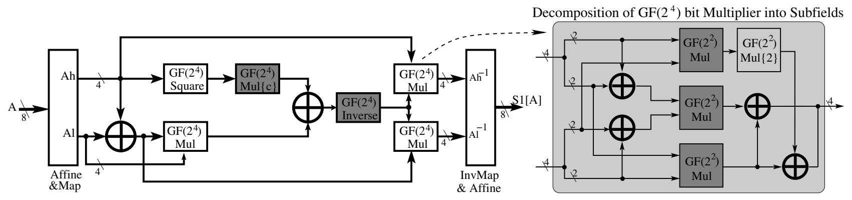


Fig. 7. Sbox $S1$ Implemented to Fit in One Cache Line (The shaded blocks are implemented as look-up tables)

it has been mainly used to increase the speed and efficiency of hardware implementations. *To the best of our knowledge, this is the first time that this method is being adapted to prevent cache attacks.* We use the composite field $GF(((2^2)^2)^2)$ generated by the irreducible polynomials x^2+x+1, $\quad x^2+x+\phi$, and $x^2+x+\lambda$ respectively, where $\phi = \{10\}_2$ and $\lambda = \{1100\}_2$ [15]. The use of composite fields allow the use of small tables for part of the computation. In particular, lookup-tables are used for the multiplication by constant and inverse in $GF(2^2)^2$; and for $GF(2^2)$ multiplication. This reduces the number of operations required for the sbox by 75. The mapping and reverse mapping between $GF(2^8)$ and $GF(((2^2)^2)^2)$ is shown below:

$$T = \begin{pmatrix} 1&0&1&1&1&0&1&1\\ 0&1&1&1&1&0&0&0\\ 0&0&0&1&0&1&0&0\\ 0&1&0&0&1&1&1&1\\ 0&1&0&1&0&0&0&1\\ 0&1&0&1&1&1&0&0\\ 0&1&1&1&1&1&1&0\\ 0&0&0&0&0&1&0&0 \end{pmatrix} \qquad T^{-1} = \begin{pmatrix} 1&0&1&0&1&0&1&0\\ 0&1&0&1&1&1&1&1\\ 0&1&0&0&0&1&0&1\\ 0&0&1&0&0&0&0&1\\ 0&1&1&1&1&0&1&1\\ 0&0&0&0&0&0&0&1\\ 0&1&0&0&0&0&1&1\\ 0&1&1&1&0&1&1&0 \end{pmatrix}$$

Using the chosen composite field, sbox $S1$ can be computed as shown in Figure 7. The inversion and constant multiplication in $GF(2^2)^2$ require 8 byte each. The 16 remaining bytes in the cache line can be used to fit in the $GF(2^2)$ multiplication table. With the experimental setup described in Section 5, Sony's reference code for CLEFIA took $655\mu s$. The reference implementation modified to not use any look-up tables took $3.04ms$, while an implementation with the proposed countermeasure took $1.5ms$. Also, it was found that the power profile of an encryption with the proposed countermeasure did not reflect the cache access patterns, thus the proposed countermeasure prevents cache attacks at a lesser performance overhead.

7 Conclusion

In this paper we develop a differential cache-trace attack and apply it on CLEFIA. The attack exploits the differential properties of the F function and key expansion. The complexity of the attack in less than 2^{14} encryptions. The attack was performed on Sony's reference code running on the PowerPC processor present in the Xilinx FPGA. Each cache access pattern in a round has a distinct power profile which can be used to identify an unknown cache trace pattern from a given power profile. Further, the paper proposes a countermeasure for cache attacks which would guarantee security, while at the same time has less performance overhead compared to non-lookup implementations.

Acknowledgments

The authors acknowledge Dr. Akashi Satoh, National Institute of Advanced Industrial Science and Technology (AIST), Japan, for providing us with the SASEBO board for our experimentation.

References

1. Acıiçmez, O., Koç, Ç.K.: Trace-Driven Cache Attacks on AES. Cryptology ePrint Archive, Report 2006/138 (2006), http://eprint.iacr.org/
2. Aciiçmez, O., Koç, Ç.K.: Trace-Driven Cache Attacks on AES (Short Paper). In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 112–121. Springer, Heidelberg (2006)
3. Bernstein, D.J.: Cache-timing Attacks on AES. Tech. rep. (2005)
4. Bertoni, G., Zaccaria, V., Breveglieri, L., Monchiero, M., Palermo, G.: AES Power Attack Based on Induced Cache Miss and Countermeasure. In: ITCC, vol. (1), pp. 586–591. IEEE Computer Society, Los Alamitos (2005)
5. Biham, E.: A Fast New DES Implementation in Software. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 260–272. Springer, Heidelberg (1997)
6. Bogdanov, A., Kizhvatov, I., Pyshkin, A.: Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 251–265. Springer, Heidelberg (2008)
7. Canright, D.: A Very Compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
8. Lauradoux, C.: Collision Attacks on Processors with Cache and Countermeasures. In: Wolf, C., Lucks, S., Yau, P.W. (eds.) WEWoRC. LNI, vol. 74, pp. 76–85. GI (2005)
9. Paar, C.: Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields. Ph.D. thesis, Institute for Experimental Mathematics, Universität Essen, Germany (June 1994)
10. Page, D.: Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel (2002)
11. Rebeiro, C., Mukhopadhyay, D.: Differential Cache Trace Attack Against CLEFIA. Cryptology ePrint Archive, Report 2010/012 (2010), http://eprint.iacr.org/
12. Renauld, M., Standaert, F.X., Veyrat-Charvillon, N.: Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009)
13. Renauld, M., Standaert, F.X.: Algebraic Side-Channel Attacks. Cryptology ePrint Archive, Report 2009/279 (2009), http://eprint.iacr.org/
14. Research Center for Information Security National Institute of Advanced Industrial Science and Technology: Side-channel Attack Standard Evaluation Board Specification, Version 1.0 (2007)
15. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)

16. Sony Corporation: The 128-bit Blockcipher CLEFIA: Algorithm Specification (2007)
17. Tromer, E., Osvik, D.A., Shamir, A.: Efficient Cache Attacks on AES, and Countermeasures. Journal of Cryptology 23(2), 37–71 (2010)
18. Xilinx Corporation: EDK9.1i: Standalone Board Support Package Document (2007)
19. Xilinx Corporation: Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet (Product Specification) (2007)
20. Xilinx Corporation: PowerPC 405 Processor Block Reference Guide (Embedded Development Kit) (2010)
21. Zhao, X., Wang, T.: Improved Cache Trace Attack on AES and CLEFIA by Considering Cache Miss and S-box Misalignment. Cryptology ePrint Archive, Report 2010/056 (2010), http://eprint.iacr.org/

Improving Differential Power Analysis by Elastic Alignment

Jasper G.J. van Woudenberg[1], Marc F. Witteman[1], and Bram Bakker[2]

Riscure BV, 2628 XJ Delft, The Netherlands
{vanwoudenberg,witteman}@riscure.com
University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
bram@science.uva.nl

Abstract. To prevent smart card attacks using Differential Power Analysis (DPA), manufacturers commonly implement DPA countermeasures that create misalignment in power trace sets and decrease the effectiveness of DPA. We design and investigate the *elastic alignment* algorithm for non-linearly warping trace sets in order to align them. Elastic alignment uses FastDTW, originally a method for aligning speech utterances in speech recognition systems, to obtain so-called warp paths that can be used to perform alignment. We show on traces obtained from a smart card with random process interrupts that misalignment is reduced significantly, and that even under an unstable clock the algorithm is able to perform alignment.

Keywords: Differential Power Analysis, unstable clock, random process interrupts, Elastic Alignment, time series analysis, dynamic time warping.

1 Introduction

Modern smart cards are devices designed for secure operation in an environment outside the control of the issuer. Because of this, they must be protected against a wide range of attacks, including side channel analysis. A powerful and well-studied technique is differential power analysis (DPA, [Koch99]). DPA analyzes the statistics of power measurements on a device while it is performing (a part of) its security function. Repeated measurements are taken of the same process, and by relating the power usage and the data values being processed, secret data may be revealed.

Smart card manufacturers are aware of these issues, and implement various countermeasures to reduce the effectiveness of attacks such as DPA. Entirely preventing DPA is often very hard, but a card can be considered secure if the resources needed for breaking it outweigh the resources an attacker has available.

Countermeasures are typically aimed at breaking the assumptions that underlie known attacks. DPA assumes that the cryptographic operations take place at exactly the same time in each power measurement. By using an internal clock with varying frequency, or randomly inserting dummy wait states into the execution of an algorithm, it is no longer time constant. Moreover, the cryptographic operations do not take place at the same instant but are shifted in time with respect to each other. Inducing this misalignment is currently one of the common countermeasures used in secure devices. Although it does not completely prevent DPA, it can make it very costly in terms of the number of traces that need to be processed.

In cryptographic implementations that do not actively induce timing differences as a countermeasure, misalignment is typically caused by inaccuracies in triggering the power measurements. This means that traces can be aligned by determining the duration of the timing inaccuracies, and shifting the traces accordingly. This process is called *static alignment* [Mang07].

In contrast, when a cryptographic implementation actively induces random time delays or varying clock frequencies, static shifting cannot fully align the traces. *Dynamic alignment* is a general term for algorithms that match parts of several traces at different offsets, and perform nonlinear resampling of the traces. This is done such that afterwards, these parts are located at the same offsets.

The starting point for our work is the observation that there are parallels between time series analysis, in particular speech recognition techniques, and alignment of power traces. This paper describes an alignment algorithm called *elastic alignment*, which is based on dynamic time warping, a well established algorithm for time series matching. We also show our algorithm is practically applicable and can be used to recover secret key leakage from misaligned traces.

1.1 Previous Work

DPA research is focused on performing an analysis that is less sensitive to misalignment. This contrasts our proposed method, which can be considered a preprocessing step before performing DPA: it directly modifies individual traces such that they are aligned. This is especially useful for the class of template attacks, which are very powerful given a correctly learned leakage model [Char03]. As the method does not explicitly take into account misalignment, its classification can be improved if it is applied to n correctly aligned traces versus n misaligned traces. This can be of significant importance, as in practice n is bounded. Also, the number of interest points may be lower for a correctly aligned trace.

There are generally three ways of dealing with misalignment when performing DPA [Mang07]: running static alignment on just the DPA target area, preprocessing traces (integration of samples, convolutions or FFT), or simply running DPA on unmodified traces. Note that running DPA on unmodified traces requires a much larger trace set, which may be infeasible.

Integration and other preprocessing methods can yield good results in terms of increasing the DPA peak; however, the increase can be limited due to the spectral components of the traces and the choice of algorithm parameters[Mang07].

In [Char05], a method is described to align traces based on the wavelet transform for denoising, and simulated annealing for resynchronization. The wavelet transform has the property that each trace can be viewed at different resolutions. It is shown that by performing DPA at lower resolutions, high frequency noise is reduced and the DPA peaks become stronger. In addition, the simulated annealing algorithm is used to optimize a resynchronization function that relates a pair of wavelet-transformed traces. This further improves the DPA peak.

Using wavelets for denoising is in our opinion a viable method; however, as we are purely considering an alignment algorithm, we do not wish to reduce the information in a trace.

In [Clav00] a method called *sliding window DPA* (SW-DPA) is introduced. SW-DPA targets random process interrupts (RPIs) by replacing each clock cycle in the original traces with an average of itself and a number of previous cycles. Its two parameters are the number of cycles to average, and the length of a cycle. Effectively, SW-DPA integrates leakage spread out over a few clock cycles back into one place, such that DPA peaks are restored.

There are two aspects of SW-DPA that need to be taken into account: the clock cycles are assumed be of fixed length, and the number of cycles to average must be specified carefully based on the number of RPIs. Our aim is to be able to deal with targets with an unstable clock as well as automatically overcome RPIs.

1.2 Organization of This Paper

This paper is organized as follows. Section 2 describes the background and design of the dynamic time warping algorithm and its linear-time successor FastDTW. In Section 3 we first introduce elastic alignment as our approach to trace set alignment, and how it is based on the warp paths, a side effect of FastDTW. Next, we analyze the alignment performance in Section 4. Final conclusions and ideas for further work are given in Section 5, and supplementary information is present in Appendix A.

2 Dynamic Time Warping

The dynamic time warping (DTW) algorithm originates from speech recognition research [Sakh78]. Matching spoken words to a database containing prerecorded words is a nontrivial problem, as words are always spoken with variances in timing. Traditionally, calculating distances between two word utterances is performed by using a measure that compares recorded words sample by sample. These are based on, e.g., the sum of squared differences between the samples or the correlation of the sample values.

However, in cases where we have two similar utterances with differences in timing, the distance under such measures will be larger than if the utterances were 'aligned'. This follows from the property that these sample-by-sample measures do not explicitly consider unaligned utterances.

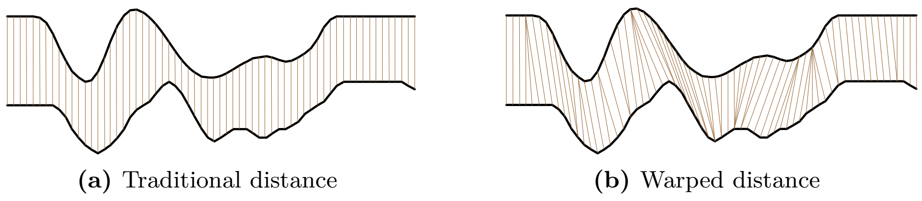


Fig. 1. Dynamic time warping distance calculation (from [Chu02])

Being confronted with this problem, Sakoe et al. introduced a dynamic programming approach to match utterances using nonlinear time paths [Sakh78]. DTW measures the distance between two utterances by 'elastically' warping them in time (see Figure 1), and then measuring the distance. Warping is performed based on the *warp path* the algorithm produces, which gives the alignment under which the signals have a minimum distance. DTW thereby allows utterances from processes with variable timing to be matched more accurately.

Traditionally, DTW is used for calculating a distance between two speech utterances. However, we are interested in trace alignment. We note that the warp path internally produced by DTW for measuring distances represents a matching between the time axes of two utterances. In this paper we use the same principle to align measured power traces from smart cards. Note that the DTW algorithm can only align two traces, so like other alignment algorithms we will be dependent on a reference trace.

The remainder of this section explains the original DTW algorithm, the improved FastDTW algorithm and how to apply the algorithm to trace pair alignment.

2.1 Obtaining the Warp Path

For our alignment we are interested in the *warp path.* The warp path is a list of indexes in both traces that represents which samples correspond to each other. Formally, if we have two traces X and Y, we define a warp path F

$$F = (c(1), c(2), \ldots, c(K)) \tag{1}$$

with $c(k) = (x(k), y(k))$ indexes in X and Y respectively. Figure 2 gives an example of a warp path.

There are several constraints on the warp path:

- **Monotonicity:** $x(k-1) \leq x(k)$ and $y(k-1) \leq y(k)$.
- **Continuity:** $x(k) - x(k-1) \leq 1$ and $y(k) - y(k-1) \leq 1$.
- **Boundary:** $x(1) = y(1) = 1$, $x(K) = T$ and $y(K) = T$, with T the number of samples in X and Y.

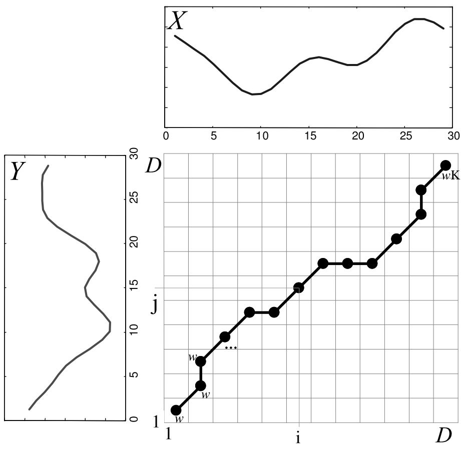


Fig. 2. Example warp path [Keog99] for traces X and Y. The warp path shows the optimal matching of the two traces by index pairs i and j.

Combined these constraints restrict the warp path to three possible steps (see Figure 3):

$$c(k+1) = (x(k+1), y(k+1)) \tag{2}$$

$$= \begin{cases} (x(k), y(k)+1) & \text{or} \\ (x(k)+1, y(k)) & \text{or} \\ (x(k)+1, y(k)+1) \end{cases} \tag{3}$$

Furthermore, the length of the warp path can be deduced to be bounded by

$$T \leq K < 2T \tag{4}$$

The monotonicity and continuity constraints are a natural choice for our problem domain: we do not allow going back in time, nor skipping any samples. The choice for the boundary constraint is based on the assumption that traces start at the same phase, and end at the same phase of a measured process. Starting at the

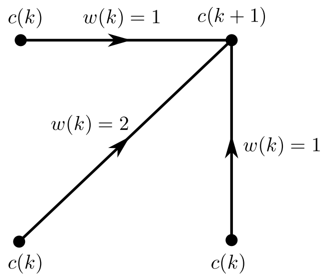


Fig. 3. Warp path steps and cost factor $w(k)$

same phase can usually be realized in side channel acquisition by timing the acquisition or static alignment. However, traces are usually of fixed length and, due to the introduction of time variations, do not necessarily end at the same phase of a process. This implies that matching the end of two traces can be difficult. DTW can overcome this by the inherent property that segments found in only one trace can be 'skipped' by matching the entire segment to a one or few samples in the other trace. Practical experience with the algorithm confirms the neither beginning nor the end of the traces need to be exactly aligned for DTW to find a good match.

Cost matrix. In order to find the minimum cost warp path, the DTW algorithm calculates a *cost matrix* d. This matrix contains the distances between all samples of X and Y, and is calculated as

$$d(i, j) = |X[i] - Y[j]| \tag{5}$$

The length of a warp path F depends on how the distances between samples combined with the path through them translate into a distance between X and Y. The measure L giving the distance for the minimum length warp path is:

$$L(X, Y) = \frac{1}{2T} \min_{F} \left[\sum_{k=1}^{K} d(c(k))w(k) \right] \tag{6}$$

where $w(k)$ is a weighting factor and T the length of the traces. The weighting factor was introduced to construct a measure with flexible characteristic. We use the symmetric measure from [Sakh78], which implies $w(k)$ is the number of steps made in each dimension:

$$w(k) = [x(k) - x(k-1)] + [y(k) - y(k-1)] \tag{7}$$

So, if we make a step only in X or only in Y then $w(k) = 1$, for diagonal step $w(k) = 2$ (see Figure 3).

Finding the minimum cost path. Having defined the distance $L(X, Y)$, we need an algorithm to find the minimum distance path. This corresponds to the optimized way of warping the two traces such that they are aligned.

A simple dynamic programming algorithm is implied by rewriting $L(X, Y)$ in recursive form:

$$g_1(c(1)) = d(c(1)) \cdot w(1) \tag{8}$$

$$g_k(c(k)) = \min_{c(k-1)} [g_{k-1}(c(k-1)) + d(c(k))w(k)] \tag{9}$$

$$L(X, Y) = \frac{1}{2T} g_K(c(K)) \tag{10}$$

If we only apply steps from Eq. (2), substitute for $w(k)$ (Eq. (7)), and drop the subscript k for simplicity, we obtain:

$$g(1,1) = 2d(1,1) \tag{11}$$

$$g(i,j) = \min \begin{bmatrix} g(i,j-1) + d(i,j) \\ g(i-1,j) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \end{bmatrix} \tag{12}$$

$$L(X,Y) = \frac{1}{2T} g(T,T) \tag{13}$$

The algorithm first calculates matrix d, and then starts at (T,T) to trace the minimum warp path according to Eq. (12). To avoid boundary problems, we define $d(0,j) = d(i,0) = \infty$. This procedure yields both the distance measure and the minimum distance warp path. An example calculation is found in Appendix A.

2.2 FastDTW: Iterative Reduction and Bounding

The time and space complexity of DTW can be restrictive in practice. Calculating $g(i,j)$ in Eq. (12) requires calculating $d(r,s)$ for all $1 \leq r \leq i$ and $1 \leq s \leq j$, and thus the complexity of calculating $g(T,T)$ is quadratic: $O(T^2)$. There are several approaches to overcome this problem which abstract the data or restrict the search space. We can constrain the search for an optimal warp path, but this should be based on knowledge (or assumptions) about the data. To balance the rigidity of constraints and algorithm performance, [Salv04] proposes the *FastDTW* algorithm.

The FastDTW algorithm restricts the warp path by bounding which elements of matrix d are calculated. This bounding is determined by the warp path at different resolutions of the traces, and, if the bounding is not too tight, produces the same results as DTW but with $O(T)$ complexity.

FastDTW uses a multilevel approach with three key operations. *Coarsening* reduces the size of a trace by averaging adjacent pairs of samples (see Figure 4).

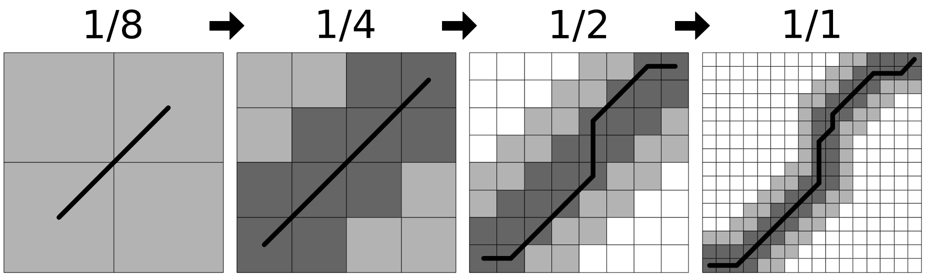


Fig. 4. Iterative DTW warp path refinement in FastDTW [Salv04]. Each iteration the resolution for both axes is doubled, and a new warp path is determined within the light gray bounds.

The resulting trace is a factor of two smaller than the original trace. Coarsening is performed several times to produce many different resolutions of the trace. At the lowest resolution, the original DTW algorithm is used to generate a warp path.

Projection takes a warp path calculated at a lower resolution and determines what cells the warp path passes through in the next higher resolution traces. This projected path is then used as a heuristic to bound the warp path at the higher resolution. *Refinement* takes this projected path and increases the bounds by a *radius*, which controls the space DTW can search beyond the projected path. Next, it finds the optimal warp path within these extended bounds by executing a bounded version of DTW. This bounded version of DTW can be understood as the original DTW, with all elements of d outside the search bounds set to ∞.

The difference between this method and the other approximations to full DTW is that the FastDTW algorithm does not rigidly determine the bounds a priori without consulting the data; the bounds are set based on the local shape of the warp path. This way a much closer approximation to the original DTW algorithm is achieved. The radius parameter affects the approximation: the higher the radius parameter, the higher the accuracy of representation of the original DTW algorithm is, but also the slower FastDTW runs. The radius parameter should not be set too high: if it is in the order of T the algorithm reduces to DTW speed, as the bounds always include all cells. Otherwise, [Salv04] shows FastDTW is in $O(T)$. In our experiments we need to find a reasonable value for this parameter.

3 Elastic Alignment Using FastDTW

In this section, we propose the *elastic alignment* algorithm. By using FastDTW on two traces, we obtain a sample-by-sample match of these traces. In order to align the traces, matching samples need to be projected onto new traces. This implies the warp path $F = (c(1), c(2), \ldots, c(K))$ between two traces X and Y gives rise to two projections onto aligned traces $\dot{X}$ and $\dot{Y}$.

Under the restriction not to increase the length of two aligned traces, we use the following asymmetric projections:

$$\dot{X}[i] = X[i] \tag{14}$$

$$\dot{Y}[j] = \frac{1}{|\{k \mid x(k) = j\}|} \sum_{x(k)=j} Y[y(k)] \tag{15}$$

with the minimal length warping path $c(k) = (x(k), y(k))$, $1 \le k \le K$, $1 \le i \le T$ and $1 \le j \le T$. These projections can be understood as elastically aligning Y to X by averaging and duplicating samples of Y based on the minimal length warping path. The length of the traces remains T.

The projections as described are only capable of aligning pairs of traces. However, the chosen asymmetric projections allow for a reference trace to be chosen. This reference trace can be used as a basis to elastically align an entire trace set

Algorithm 1. Elastic Alignment

```
1. Obtain a reference trace $X$ and a trace set $\mathcal{Y}$; all traces of length $T$
2. For each trace $Y \in \mathcal{Y}$
    (a) Calculate warp path $c(k) = (x(k), y(k))$, for $X$ and $Y$ using FastDTW
    (b) Calculate $\dot{Y}[j]$ for $1 \leq j \leq T$, output $\dot{Y}$
```

by aligning each trace to this reference trace, as described in Algorithm 1. Note this reference trace can be included in the trace set to align, but this is not a necessity.

The difference between this algorithm and FastDTW is that the latter focuses on distance calculation between two traces, producing a warp path as byproduct. Elastic alignment uses this warp path to resynchronize traces.

3.1 Computational Complexity

The complexity of elastic alignment is per trace $O(T)$ for FastDTW, and $O(K)$ for the resynchronisation step. Because $T \leq K < 2T$, this makes the total complexity linear in the trace length and the number of traces: $O(T \cdot |\mathcal{Y}|)$.

FastDTW has a radius parameter that trades off computation time and time series matching optimality. For DPA this means it is possible to tune the alignment quality and the computation time. The radius has an optimum value at which increasing it does not affect the alignment, but only increases the computation time. This is the point at which the optimal warp path is fully contained within the radius for each FastDTW iteration.

As this optimal value depends on the characteristics of the traces under analysis, it is non-trivial to give a general formula for the exact value. By starting with a low radius, and continuously increasing it until the alignment of a pair of traces becomes stable, we usually find the optimal radius value lies between 100 and 150.

3.2 Usage for DPA

One of the more practical problems encountered when performing DPA, is that of misalignment due to unstable clocks and random process interrupts. Because of the continous trace matching, elastic alignment synchronizes to the reference clock and process. Unstable clocks are therefore automatically accounted for.

However, random process interrupts (RPIs), an active countermeasure introducing misalignment, can also be overcome. If the RPI is present in the reference trace, the other trace can be stretched in order to 'skip' the RPI. Conversely, if the RPI is present in the other trace, it can be compressed down to a one or a few samples. An example of the effect of elastic alignment is shown in Figure 5.

Continuous synchronisation is important when a DPA attack on multiple serially executed targets is mounted. In e.g. an AES software implementation, there

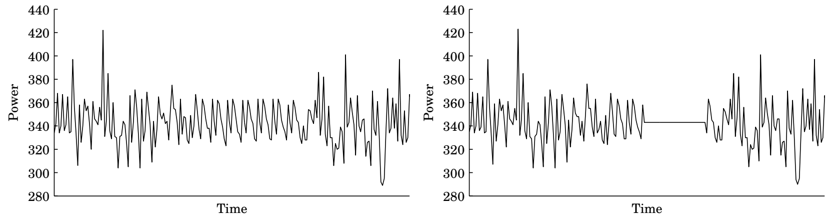


Fig. 5. Two traces with random length interrupt aligned by elastic alignment

are 16 consecutive S-box lookups that need to be targeted. With elastic alignment, a trace set with an unstable clock needs to be aligned only once, whereas with static alignment 16 different alignments need to be performed.

3.3 Elastic Alignment Considerations

Although appointing one trace from a set as reference trace seems arbitrary, in practice reference traces (or parts thereof) are used for trace alignment. We have experimented with elastically aligning trace sets without a reference trace by using a hierarchical alignment scheme that iterates trace pair alignment using symmetric projections. We found that even though the alignment works, the (linearly bounded) increase in trace length implied by this method is less practical and does not necessarily outperform elastic alignment with a reference trace.

We choose not to dispose of samples, but to merely compress or stretch areas. This is because we cannot say which amount of local misalignment is actually an RPI that can be disposed of, and which misalignment is caused by an unstable clock or slightly differing instruction paths. By not disposing any samples, we decrease the possibility of removing interesting information.

4 Experiments

In our experiments we test to what degree elastic alignment increases the effectiveness of a power attack when countermeasures are present that induce misalignment: random process interrupts, and an unstable clock. We compare elastic alignment with sliding window DPA (SW-DPA,[Clav00]), a technique that targets RPIs.

We target a smart card on which we implement the first round of the DES cipher in software, and introduce random process interrupts. These interrupts randomly halt the card for 0 or 1 cycles, before each of the 8 S-box lookups. This process causes misalignment in the acquired traces.

Unstable clocks are typical for cards with an internal clock. Our sample does not have an internal clock, and we are unaware of programmable cards with an internal clock that are vulnerable to DPA within a few thousand traces. Therefore, we choose to process the obtained traces and introduce an unstable clock by duplicating or removing a sample at the end of only a small fraction of clock cycles. This is the second set of traces we will be analysing.

Note that in our set with a stable clock, SW-DPA can exactly average consecutive cycles. With the unstable clock, it is not possible for SW-DPA to correctly average clock cycles, as it assumes a fixed clock length. We therefore expect SW-DPA to perform better with the stable clock than with the unstable clock. Elastic alignment should be able to deal with both scenarios, as it automatically synchronizes to the clock.

4.1 Measuring DPA Success Rate

To analyze the results, we perform correlation power analysis (CPA, [Brie04]) on trace sets of different number of traces, and calculate the first order success rate graph [Stan08]. This graph displays for increasing trace set size what the estimated probability is of finding the correct key as first candidate.

We know our target implementation strongly leaks the Hamming weight of the processed intermediates. For CPA, we therefore use the Hamming weight power model and correlate with the output of the DES S-boxes. We are targeting the first round of DES for efficiency reasons. Because of this, the total key we recover has 48 bits. As elastic alignment and SW-DPA are signal processing techniques, there is no reason to assume they perform differently if more rounds or even another cryptographic algorithm is used.

4.2 Trace Acquisition and Processing

We acquire one set of 100000 traces. All traces are acquired by measuring the instantaneous power consumption of the card by an oscilloscope sampling at 50MHz, using an analog low pass filter at 11MHz. The clock of the card runs at 4MHz, and we compress the traces by averaging consecutive samples resulting in one sample per clock period. The number of samples per trace is 5600.

From this original trace set we generate two derived trace sets: one with a stable cycle, and one with an unstable cycle. From Fourier transforms of measurements on various cards with unstable clocks we know the clock to be strongly centered around its base frequency, with a sharp dropoff in both tails. This sharp dropoff indicates the instability to be small, and we therefore choose to create the derived trace sets such that the instability is small as well: the stable cycle has 5 samples per clock, and the unstable cycle length is determined by a rounded Gaussian distribution: $\lfloor L + 0.5 \rfloor, L \sim N(5, 0.2)$, which yields about 98.7% cycles of 5 samples, and only 1.3% cycles of 4 or 6 samples. Each cycle is represented by one sample with the measured value, followed by samples that are the average of the previous sample and the minimum sample value of the trace. This corresponds with the observation that leakage is typically concentrated in a specific peak of the cycle.

For the experiments with elastic alignment, we set the radius parameter to 110. This is an experimentally derived value that balances computational performance and output quality. We start with a low value and increase it until

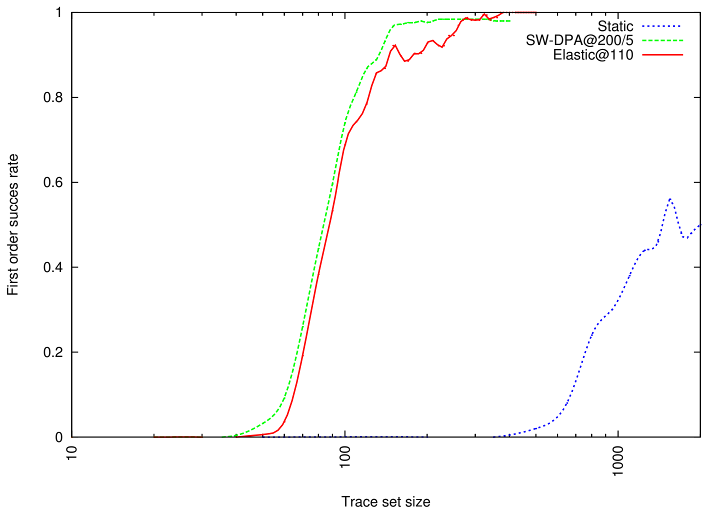


Fig. 6. CPA success rate for stable cycle length

it does not significantly improve the alignment quality. With this value for the radius parameter, aligning each trace takes about 2 seconds on a current 2.4GHz processor.

For SW-DPA, we choose to use the average length of a clock cycle in our measurements as the distance parameter, in these measurements 5 samples. The number of consecutive cycles to average is set to 200, which is the experimentally determined width of the distribution of the widest CPA peak (for the last S-Box in the calculation).

4.3 Results

For the experiments with fixed cycle length (Figure 6) we first see that the random process interrupts thwart DPA with static alignment: at about 1400 traces we only obtain a success rate around 0.5. For SW-DPA, we observe the effect of a perfect match between countermeasure and analysis technique: the averaging of fixed length clock cycles has restored the DPA peak in the face of random process interrupts. A success rate of close to 1 is already obtained at 160 traces.

Elastic alignment shows the same success rate around 270 traces. This is likely due to the fact that elastic alignment is an adaptive method, and noise may be affecting the matching of trace sections. However, compared with ordinary DPA it is within the same order of magnitude as SW-DPA.

The results become very different when unstable clocks are introduced. The success rate of SW-DPA, seen in Figure 7, goes to 0 for all experimented trace set sizes up to 1000. The same holds true for static alignment. In fact, we have attempted to perform DPA using this set at 100000 traces, and with SW-DPA the key could not be found; with static alignment it was found at around 67000

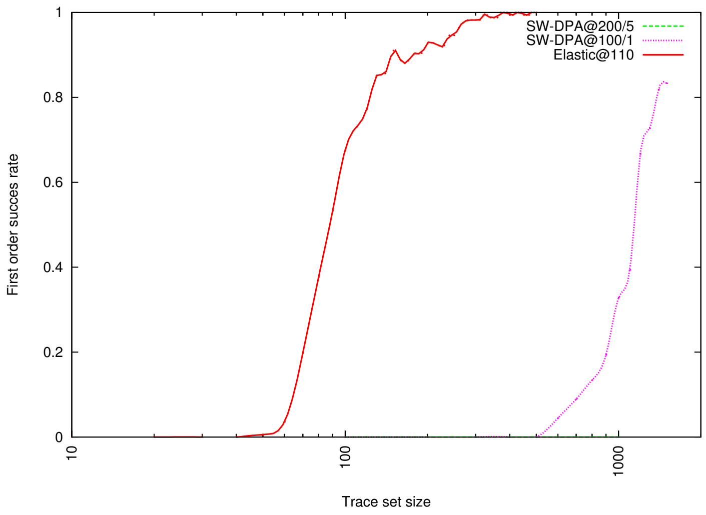


Fig. 7. CPA success rate for stable cycle length

traces. So, even at 1.3% cycles with a different length than average, SW-DPA gets desynchronized and harms the correlation peak.

We have been able to get SW-DPA somewhat back by tweaking the parameters: setting the clock cycle length to 1 and setting the number of clocks to average to 100, we could get a 50% success rate at about 1150 traces, as seen in Figure 7. Note SW-DPA then acts more like a moving average filter: it does not have the 'comb' effect normally used to accumulate specific samples within a clock cycle.

Elastic alignment is by design able to overcome these desynchronized traces. The results shows that it is relatively unaffected by the unstable clock: the smallest trace set size with a success rate close to 1 increases marginally. Preliminary experiments show this also holds for a wider distribution of 2.5%, 47.5%, 47.5% and 2.5% cycles of length 2, 3, 4 and 5 respectively.

These experiments show that elastic alignment is able to deal with random process interrupts, and is very suited to dealing with unstable clocks due to its ability to continuously adapt alignment to the reference trace. This adaptativeness does imply it may also 'adapt' to noise that is present. We have some ideas and preliminary experiments showing how to overcome this, as described in section 5.1.

5 Conclusions

In this paper we described and experimented with elastic alignment, an algorithm for aligning trace sets when misalignment is present. The elastic alignment is designed to be applicable practically in the context of performing side channel analysis in the presence of random process interrupts and unstable clocks.

We use FastDTW, a linear complexity variant of the dynamic time warping algorithm, for alignment. Dynamic time warping measures the distance between two traces and produces a warp path that describes a nonlinear time matching of the two. We use this warp path to align a pair of traces. By selecting one trace as a reference, we can thereby iterate this process to elastically align an entire trace set.

By design, elastic alignment attempts to globally optimize the alignment. This implies that at every resolution, be it process, instruction, clock or sub-clock, a good alignment can be found. For side channel analysis this is helpful as the traces may be analyzed at different resolutions.

The only parameter input to the algorithm is the FastDTW radius, which acts as a speed versus quality trade-off. This contrasts sliding window DPA, which requires knowledge of two target specific parameters: the clock cycle length, and the spread of the DPA peak over different clock cycles. Having fewer and easily selectable parameters makes the effectiveness of the side channel analysis less dependent on the user performing it.

Experiments were done based on traces obtained from a card with a fixed clock and the random process interrupt countermeasure enabled. These show sliding window DPA is moderately better than elastic alignment at dealing with only RPIs as countermeasure. This is probably due to noise affecting the dynamic adaptation of elastic alignment. However, as soon as even a slightly unstable clock is introduced, elastic alignment is much better at recovering the DPA peak due to its dynamic synchronization with the reference trace.

The experiments conform with our experiences on other cards, which are mostly implementations with hardware DES engines. Unfortunately we do not fully control these implementations, and they are therefore less suitable for structured experimentation. Elastic alignment also appears to work when traces are compressed down to the frequency of the internal clock, and, with proper signal integration, also on EM traces. In a number of cases, elastic alignment has played a key role in breaking a card using CPA within a bounded number of traces.

5.1 Discussion and Future Work

Besides the basic elastic alignment algorithm as presented in this paper, we have implemented a number of other experimental features. One is the possibility of decoupling warp path detection and application: we allow detecting the warp paths on one set of traces, and applying them to another set of traces. If the other set of traces has a different number of samples, the warp path is scaled to accommodate this. This allows us to e.g. calculate an alignment at the level of one sample per clock, while repairing misalignment in traces with multiple samples per clock. The experiments and results are preliminary, but show interesting potential.

When performing elastic alignment, we implicitly violate the preconditions that the first and last samples of the trace pair are aligned. Elastic alignment can accommodate for this by matching the initial or final part of one trace to only a few samples in the other; however, we envision the alignment can be improved if both the first and last samples of the trace pair are aligned. This can be implemented by allowing variable length traces and using pattern matches to find the locations of the first and last samples.

A way to reduce the effect of noise on the alignment is to change the way FastDTW measures the distance between individual samples. Now this is done by their absolute difference, but one could consider taking more samples into account. This is accomplished by for instance using the weighted average distance of neighboring samples, or by correlation of trace fragments. This effectively 'smoothes' the distance function and potentially cancels some of the effects of noise in the individual samples.

Acknowledgments

We would like to thank Fred de Beer for helpful comments and discussions regarding elastic alignment. Also, we are grateful for the helpful comments about the previous version of this paper from the anonymous reviewers. We also thank Nick Towner for reviewing an earlier version this paper. Finally, we are indebted to Federico Menarini and Eloi Sanfelix Gonzalez for providing us with the trace sets used in the experiments.

References

[Brie04] Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)

[Char03] Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)

[Char05] Charvet, X., Pelletier, H.: Improving the DPA attack using Wavelet transform. In: NIST Physical Security Testing Workshop (2005)

[Chu02] Chu, S., Keogh, E., Hart, D., Pazzani, M.: Iterative deepening dynamic time warping for time series. In: Proceedings 2 SIAM International Conference on Data Mining (2002)

[Clav00] Clavier, C., Coron, J.-S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)

[Keog99] Keogh, E., Pazzani, M.: Scaling up Dynamic Time Warping to Massive Datasets. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 1–11. Springer, Heidelberg (1999)

[Koch99] Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

[Mang07] Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of smart Cards. Springer, Heidelberg (2007)

[Sakh78] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoustics, Speech, and Signal Processing 26, 143–165 (1978)

[Salv04] Salvador, S., Chan, P.: FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In: Proc. KDD Workshop on Mining Temporal and Sequential Data (2004), Java implementation, `http://cs.fit.edu/~pkc/FastDTW/FastDTW.zip`

[Stan08] Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition vs. Comparison Side-Channel Distinguishers. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)

A DTW Calculation Example

Traces

X	1	1	2	3	4
Y	1	2	3	4	4

$d(i,j)$

	3	3	2	1	0
	3	3	2	1	0
$i \uparrow$	2	2	1	0	1
	1	1	0	1	2
	0	0	1	2	3

$j \rightarrow$

$g(i,j)$

	9	9	5	2	0
	6	6	3	1	0
$i \uparrow$	3	3	1	0	1
	1	1	0	1	3
	0	0	1	3	6

$j \rightarrow$

Warp path

$$F = ((1,1),(2,1),(3,2),(4,3)(5,4),(5,5))$$

NSA's Role in the Development of DES

Richard M. George

Information Assurance Directorate
National Security Agency
r.george@radium.ncsc.mil

Abstract. 38 years ago, NIST put out a call for submissions of candidates for data encryption standard to address the needs of encryption for the commercial world. Of the submissions, the IBM submission stood out as arguably the best candidate. However, before the algorithm was ready to be chosen as the Data Encryption Standard (DES), some changes were required. The National Security Agency (NSA) worked with IBM on the modification of the submitted algorithm. This talk will discuss what role NSA played in this effort, the rationale for the changes that were made, and the impact that DES had at that time.

Designing Efficient Authenticated Key Exchange Resilient to Leakage of Ephemeral Secret Keys

Atsushi Fujioka and Koutarou Suzuki

NTT Information Sharing Platform Laboratories
3-9-11 Midori-cho Musashino-shi Tokyo 180-8585, Japan
{fujioka.atsushi,suzuki.koutarou}@lab.ntt.co.jp

Abstract. We investigate a sufficient condition for constructing authenticated key exchange (AKE) protocols which satisfy security in the extended Canetti-Krawczyk (eCK) model proposed by LaMacchia, Lauter and Mityagin. To the best of our knowledge, this is the first approach for providing secure protocols based on the condition. With this condition, we propose a construction of two-pass AKE protocols, and the resulting two-pass AKE protocols are constructed with a single static key and a single ephemeral. In addition, the security proof does not require the Forking Lemma, which degrades the security of a protocol relative to the security of the underlying problem where it is used in the security proof. Therefore, these imply that the protocols constructed with the condition have an advantage in efficiency such as sizes of storage and communication data. The security of the resulting protocols is proved under the gap Diffie-Hellman assumption in the random oracle model.

Keywords: authenticated key exchange, eCK model, gap Diffie-Hellman assumption.

1 Introduction

In network security, one of the most important techniques is to establish secure channels. Secure channels provide secrecy and authenticity for both communication parties. When the parties can share information via a public communication channel, secure channels are constructed on (symmetric key) encryptions and message authentication codes with the shared information as keys. A key exchange protocol, called *authenticated key exchange* (AKE), provides a solution for sharing a key via a public communication channel, and both parties are assured that only their intended peers can derive the session key.

In AKE, each party has public information, called *static public key*, and the corresponding secret information, called *static secret key*. The static public key is expected to be certified with the party's identity by a system such as a public key infrastructure (PKI). A user who wants to share a key with some entity exchanges informations several times and then computes the shared key. In two-pass AKE, the user generates *ephemeral public keys* and the corresponding *ephemeral secret keys*, and sends the ephemeral public keys to the peer, and the receiving peer also generates ephemeral public keys and the corresponding ephemeral secret keys and returns the ephemeral public keys to the sender. Both parties compute *shared values* from their static public keys, the corresponding static secret keys, the exchanged ephemeral public keys, and the corresponding ephemeral secret keys, and then derive a *session key* from these values including the shared values. The session key is computed with a function called *key derivation function*, and in most cases, the key derivation function is a hash function regarded as a random oracle, where security is proved in the random oracle model [5].

The security model and definition for AKE were first proposed by Bellare and Rogaway [4]. They defined AKE's security based on an indistinguishability game, where an adversary is required to differentiate between a random key and a session key. After their investigation, several variations have been proposed, Canetti and Krawczyk proposed the Canetti-Krawczyk (CK) model to capture the desirable security notion [7], and recently, the CK model was extended by LaMacchia, Lauter, and Mityagin, the extended Canetti-Krawczyk (eCK) model [14], which is one of the most significant models since the adversary in this model is allowed to access the secret information of either the static or ephemeral keys in the test session. Although the eCK-secure AKE protocols satisfy a strong security requirement, the security proofs are difficult and complex.

We propose a construction of two-pass AKE protocols, and give a sufficient condition for constructing eCK-secure protocols under the gap Diffie-Hellman (GDH) assumption [25]. The resulting AKE protocols are constructed with a single static key, a single ephemeral key, and several shared values using a single hash function. We give requirements regarding the exponents of shared values computed as the intermediate value in the protocols.

In an original Diffie-Hellman protocol [9], a party uses a single key to compute a shared value, that is Y^x from x and Y, and the peer also computes X^y from y and X, where $X = g^x$, $Y = g^y$, and g is a generator of a cyclic group where it is generated by primitive element g. We extend this exponent of the shared value to weighted inner product of two-dimensional vectors related to the exponents of the static and ephemeral public keys. For two vectors $u = (u_0, u_1)$, $v = (v_0, v_1)$ and two-dimensional square matrix C, the shared value is computed as g^{uCv^T}, where T is a transposition operation. Then, the exponent of the shared value is given as a quadratic polynomial of u_0, u_1, v_0, and v_1. We introduce *admissible* polynomials, and when the exponents of the share value in an AKE protocol are expressed by admissible polynomials, we can construct a reduction algorithm, which interacts with the adversary and solves a computational Diffie-Hellman (CDH) problem with the help of a decisional Diffie-Hellman (DDH) oracle. The algorithm simulates all queries the adversary requires and extracts the answer of the CDH instance. The resulting AKE protocols based on admissible polynomials contain not only the existing efficient protocols but also new eCK-secure protocols. That is, our sufficient condition is useful for constructing two-pass AKE protocols.

Roughly speaking, the CDH problem is to compute the CDH value, g^{xy}, from X $(= g^x)$ and Y $(= g^y)$ in a cyclic group where it is generated by primitive element g [9]. The DDH problem is to decide whether Z is random or the CDH value of X and Y given X, Y, and Z, and the GDH problem is to solve the CDH problem with the help of the DDH oracle. The GDH (CDH) assumption is that the GDH (CDH) problem is assumed to be difficult for any polynomial-time algorithm to solve.

To the best of our knowledge, this is the first approach for providing secure protocols based on a sufficient condition. Once the exponents of the share values in an AKE protocol are expressed by admissible polynomials, the AKE protocol is eCK-secure. It is required only to confirm that the exponents are expressed by admissible polynomials, and this confirmation is an easier task than the proof of eCK-security.

Although the security of the protocols constructed under the proposed condition is proved in the random oracle model, its security proof is done without the Forking Lemma [26]. Notice that in the case of using the Forking Lemma, the security parameter in the protocols must be bigger than the expected one in the underlying problem since the security degrades according to the number of hash queries. Thus, the protocols need longer key-length to meet the security parameter and they may loose the advantage in efficiency. The resulting protocols have an advantage in efficiency as number of the static keys and the ephemeral keys are related to the sizes of storage and communication data in the system, respectively.

The eCK-secure protocols using a single hash function were proposed as SMEN$^-$ [30], Kim-Fujioka-Ustaoğlu's Protocol 1 [11] (denoted as KFU1), and Kim-Fujioka-Ustaoğlu's Protocol 2 [11] (denoted as KFU2), but these protocols need not only two static keys but also two ephemeral keys or more shared values. Thus, they are less efficient regarding storage size or communication data size. It is an interesting question to construct an eCK-secure protocol with a single static key and a single ephemeral key using a single hash function.

Organization. In **Section 2**, the eCK model for AKE is reviewed, and we propose a construction of two-pass AKE protocols, and discuss security arguments in **Section 3**. In **Section 4**, we compare protocols based on the proposed sufficient condition with other relevant protocols, and then, conclude the paper in **Section 5**. A discussion on security is given in the **Appendix**.

2 eCK-Security Model

In this section, we review the eCK-security model for two-pass PKI-based AKEs by the LaMacchia, Lauter, and Mityagin [14].

We denote a user as U_i, and user U_i and other parties are modeled as probabilistic polynomial-time Turing machines w.r.t. security parameter κ. For user U_i, we denote the static secret (public) key as s_i (S_i) and ephemeral secret (public) key as x_i (X_i, respectively).

Session. An invocation of a protocol is called a *session.* Session activation is done by an incoming message of the form $(\Pi, \mathcal{I}, U_A, U_B)$ or $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, where we equate Π with a protocol identifier, $\mathcal{I}$ and $\mathcal{R}$ with role identifiers, and

U_A and U_B with user identifiers. If U_A was activated with $(\Pi, \mathcal{I}, U_A, U_B)$, then U_A is called the session *initiator*. If U_B was activated with $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, then U_B is called the session *responder*. The initiator U_A outputs X_A, then may receive an incoming message of the form $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ from the responder U_B, and computes the session key K if U_A received the message. On the other hand, responder U_B outputs X_B and computes the session key K.

If U_A is the initiator of a session, the session is identified as $\mathtt{sid} = (\Pi, \mathcal{I}, U_A, U_B, X_A)$ or $\mathtt{sid} = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$. If U_B is the responder of a session, the session is identified as $\mathtt{sid} = (\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$. We say that U_A is the *owner* of session $\mathtt{sid}$ if the 3-rd coordinate of session $\mathtt{sid}$ is U_A, and that U_A is the *peer* of session $\mathtt{sid}$ if the 4-th coordinate of session $\mathtt{sid}$ is U_A. We say that a session is *completed* if its owner computes the session key. The *matching session* of $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ is a session with identifier $(\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$ and vice versa.

Adversary. The adversary $\mathcal{A}$, which is modeled as a probabilistic polynomial-time Turing machine, controls all communications between parties, including session activation, by performing the following adversary query.

– Send(message): The message has one of the following forms: $(\Pi, \mathcal{I}, U_A, U_B)$, $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, or $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$. Adversary $\mathcal{A}$ obtains the response from the user.

To capture leakage of secret information, adversary $\mathcal{A}$ is allowed to issue the following queries.

– $\mathsf{SessionKeyReveal}(\mathtt{sid})$: Adversary $\mathcal{A}$ obtains the session key K for the session $\mathtt{sid}$ if the session is completed.
– $\mathsf{EphemeralKeyReveal}(\mathtt{sid})$: Adversary $\mathcal{A}$ obtains the ephemeral secret key x of owner of the session $\mathtt{sid}$.
– $\mathsf{StaticKeyReveal}(U_i)$: Adversary $\mathcal{A}$ obtains the static secret key s_i of user U_i.
– $\mathsf{EstablishParty}(U_i, S_i)$: This query allows adversary $\mathcal{A}$ to register the static public key U_i on behalf of the party U_i and adversary $\mathcal{A}$ totally controls that party. If a party is established by $\mathsf{EstablishParty}(U_i, S_i)$ query issued by adversary $\mathcal{A}$, then we call party U_i *dishonest*. If not, we call the party *honest*.

Freshness. For the security definition, we need a notion of freshness.

Definition 1 (Freshness). *Let* $\mathtt{sid}^* = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ *or* $(\Pi, \mathcal{R}, U_A, U_B, X_B, X_A)$ *be a completed session between honest user* U_A *and* U_B*. If a matching session exists, then let* $\overline{\mathtt{sid}^*}$ *be the matching session of* $\mathtt{sid}^*$*. We say session* $\mathtt{sid}^*$ *is* fresh *if none of the following conditions hold:*

1. *Adversary* $\mathcal{A}$ *issues* $\mathsf{SessionKeyReveal}(\mathtt{sid}^*)$*, or* $\mathsf{SessionKeyReveal}(\overline{\mathtt{sid}^*})$ *if* $\overline{\mathtt{sid}^*}$ *exists,*

2. $\overline{\mathtt{sid}^*}$ *exists and adversary* $\mathcal{A}$ *makes either of the following queries*
   - *both* StaticKeyReveal(U_A) *and* EphemeralKeyReveal($\mathtt{sid}^*$)*, or*
   - *both* StaticKeyReveal(U_B) *and* EphemeralKeyReveal($\overline{\mathtt{sid}^*}$)*,*
3. $\overline{\mathtt{sid}^*}$ *does not exist and adversary* $\mathcal{A}$ *makes either of the following queries*
   - *both* StaticKeyReveal(U_A) *and* EphemeralKeyReveal($\mathtt{sid}^*$)*, or*
   - StaticKeyReveal(U_B)*.*

Security Experiment. For the security definition, we describe the following security experiment. Initially, adversary $\mathcal{A}$ is given a set of honest users and makes any sequence of the queries described above. During the experiment, adversary $\mathcal{A}$ makes the following query.

- Test($\mathtt{sid}^*$): Here, $\mathtt{sid}^*$ must be a fresh session. Select random bit $b \in_U \{0, 1\}$, return the session key held by $\mathtt{sid}^*$ if $b = 0$, and return a random key if $b = 1$.

The experiment continues until adversary $\mathcal{A}$ makes a guess b'. Adversary $\mathcal{A}$ *wins* the game if the test session $\mathtt{sid}^*$ is still fresh and if the guess of adversary $\mathcal{A}$ is correct, i.e., $b' = b$. The advantage of adversary $\mathcal{A}$ in the AKE experiment with the PKI-based AKE protocol Π is defined as

$$\mathrm{Adv}_{\Pi}^{\mathrm{AKE}}(\mathcal{A}) = \Pr[\mathcal{A}\ wins] - \frac{1}{2}.$$

We define the security as follows.

Definition 2 (Security). *We say that a PKI-based AKE protocol* Π *is* secure in the eCK model *if the following conditions hold:*

1. *If two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key.*
2. *For any probabilistic polynomial-time bounded adversary* $\mathcal{A}$*,* $\mathrm{Adv}_{\Pi}^{\mathrm{AKE}}(\mathcal{A})$ *is negligible in security parameter* κ*.*

3 Proposed AKE Protocol

In this section, we first define the notion of *admissible* polynomials and then provide the proposed AKE protocol based on the admissible polynomials. The proposed AKE protocol is a natural extension of the Diffie-Hellman key exchange, where shared value g^{xy} is computed w.r.t. ephemeral public keys g^x of user U_A and g^y of user U_B. The protocol is a two-dimensional generalization of the Diffie-Hellman key exchange, i.e., shared value $g^{p(u,v)}$ is computed w.r.t. the static and ephemeral public keys (g^a, g^x) of user U_A and the static and ephemeral public keys (g^b, g^y) of user U_B, where

$$p(u, v) = \begin{pmatrix} a & x \end{pmatrix} \begin{pmatrix} c_{0,0} & c_{0,1} \\ c_{1,0} & c_{1,1} \end{pmatrix} \begin{pmatrix} b \\ y \end{pmatrix}$$

is a weighted inner product of vectors $v = (a, x)$ and $v = (b, y)$ of secret keys.

3.1 Admissible Polynomials

We define the notion of admissible polynomials over $\mathbb{Z}_q$, where $\mathbb{Z}_q$ is the additive group with prime modulus q.

Definition 3 (Admissible Polynomials). *We say m polynomials $p_i \in \mathbb{Z}_q[u_0, u_1, v_0, v_1]$ $(i = 1, ..., m)$ are* admissible *if the following conditions are satisfied.*

1. $p_i(u_0, u_1, v_0, v_1) = c_{i,0,0}u_0v_0 + c_{i,0,1}u_0v_1 + c_{i,1,0}u_1v_0 + c_{i,1,1}u_1v_1$.
2. *For any f $(= 0, 1)$, there exist i, j $(1 \le i, j \le m)$, s.t.*

$$(c_{i,f,0}, c_{i,f,1}) \text{ and } (c_{j,f,0}, c_{j,f,1})$$

are linearly independent, and for any $f' = 0, 1$, there exist i, j $(1 \le i, j \le m)$, s.t.

$$(c_{i,0,f'}, c_{i,1,f'}) \text{ and } (c_{j,0,f'}, c_{j,1,f'})$$

are linearly independent.

3. *For any i $(= 1, ..., m)$, either of the following conditions holds: a) $p_i(u_0, u_1, v_0, v_1)$ is expressed as a product of $\ell_i(u_0, u_1)$ and $\ell'_i(v_0, v_1)$, where $\ell_i(u_0, u_1)$ and $\ell'_i(v_0, v_1)$ are linear combinations of u_0, u_1 and v_0, v_1, respectively, s.t.*

$$p_i(u_0, u_1, v_0, v_1) = \ell_i(u_0, u_1)\ell'_i(v_0, v_1).$$

Or b) for any f $(= 0, 1)$, $c_{i,f,0}u_fv_0 + c_{i,f,1}u_fv_1$ is expressed as a product of $\ell_{i,f,*}(u_0, u_1)$ and $\ell'_{i,f,*}(v_0, v_1)$, where $\ell_{i,f,*}(u_0, u_1)$ and $\ell'_{i,f,*}(v_0, v_1)$ are linear combinations of u_0, u_1 and v_0, v_1, respectively, s.t.

$$c_{i,f,0}u_fv_0 + c_{i,f,1}u_fv_1 = \ell_{i,f,*}(u_0, u_1)\ell'_{i,f,*}(v_0, v_1),$$

and for any f' $(= 0, 1)$, $c_{i,0,f'}u_0v_{f'} + c_{i,1,f'}u_1v_{f'}$ is expressed as a product of $\ell_{i,*,f'}(u_0, u_1)$ and $\ell'_{i,*,f'}(v_0, v_1)$, where $\ell_{i,*,f'}(u_0, u_1)$ and $\ell'_{i,*,f'}(v_0, v_1)$ are linear combinations of u_0, u_1 and v_0, v_1, respectively, s.t.

$$c_{i,0,f'}u_0v_{f'} + c_{i,1,f'}u_1v_{f'} = \ell_{i,*,f'}(u_0, u_1)\ell'_{i,*,f'}(v_0, v_1).$$

From admissible polynomials p_i $(i = 1, ..., m)$, we construct an AKE protocol, where m shared values $Z_i = g^{p_i(u_0,u_1,v_0,v_1)}$ are computed w.r.t. the static and ephemeral public keys (g^{u_0}, g^{u_1}) of user U_A and the static and ephemeral public keys (g^{v_0}, g^{v_1}) of user U_B. We denote the AKE protocol as $\Pi_{p_1,...,p_m}$. It may be worth noting that $p_i(u_0, u_1, v_0, v_1)$ is expressed by using a matrix as

$$p_i(u_0, u_1, v_0, v_1) = \begin{pmatrix} u_0 & u_1 \end{pmatrix} \begin{pmatrix} c_{i,0,0} & c_{i,0,1} \\ c_{i,1,0} & c_{i,1,1} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \end{pmatrix}.$$

From the first condition, both users can compute the shared values. From the second condition, the simulator can extract the answer of a GDH instance in the security proof. From the third condition, the simulator can check that the shared values are correctly formed w.r.t. static and ephemeral public keys in the security proof. See sketch of the proof of Theorem 1 in Appendix A for details. We provide examples of admissible polynomials below.

Example 1. The first example of admissible polynomials is

$$m = 2,\ p_1(a, x, b, y) = (a + x)(b + y),\ p_2(a, x, b, y) = ay + xb.$$

We show that the example satisfies the conditions of the definition. The first condition is satisfied since we have

$$p_1(a, x, b, y) = (a + x)(b + y) = \begin{pmatrix} a & x \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} b \\ y \end{pmatrix},$$

$$p_2(a, x, b, y) = ay + xb = \begin{pmatrix} a & x \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} b \\ y \end{pmatrix}.$$

The second condition is satisfied since there exist $i = 1$ and $j = 2$, s.t.

$$(c_{i,0,0}, c_{i,0,1}) = (1, 1) \text{ and } (c_{j,0,0}, c_{j,0,1}) = (0, 1),$$
$$(c_{i,1,0}, c_{i,1,1}) = (1, 1) \text{ and } (c_{j,1,0}, c_{j,1,1}) = (1, 0),$$
$$(c_{i,0,0}, c_{i,1,0}) = (1, 1) \text{ and } (c_{j,0,0}, c_{j,1,0}) = (0, 1),$$
$$(c_{i,0,1}, c_{i,1,1}) = (1, 1) \text{ and } (c_{j,0,1}, c_{j,1,1}) = (1, 0)$$

are linearly independent.

The third condition is satisfied since we have

$$p_1(a, x, b, y) = \ell_i \ell'_i, \text{ where } \ell_i = a + x,\ \ell'_i = b + y$$

for $i = 1$, and we have

$$c_{2,0,0}ab + c_{2,0,1}ay = \ell_{2,0,*}\ell'_{2,0,*}, \text{ where } \ell_{2,0,*} = a,\ \ell'_{2,0,*} = y,$$
$$c_{2,1,0}xb + c_{2,1,1}xy = \ell_{2,1,*}\ell'_{2,1,*}, \text{ where } \ell_{2,1,*} = x,\ \ell'_{2,1,*} = b,$$
$$c_{2,0,0}ab + c_{2,1,0}xb = \ell_{2,*,0}\ell'_{2,*,0}, \text{ where } \ell_{2,*,0} = x,\ \ell'_{2,*,0} = b,$$
$$c_{2,0,1}ay + c_{2,1,1}xy = \ell_{2,*,1}\ell'_{2,*,1}, \text{ where } \ell_{2,*,1} = a,\ \ell'_{2,*,1} = y,$$

for $i = 2$.

The AKE protocol $\Pi_{(a+x)(b+y),ay+xb}$ constructed from these admissible polynomials requires 3 exponential operations (excluding the exponentiation for the ephemeral public key) and 2 shared values.

Example 2. The second example of admissible polynomials is

$$m = 2,\ p_1(a, x, b, y) = (a + x)(b + y),\ p_2(a, x, b, y) = (ca + x)(cb + y),$$

where c is a small integer not equal to 1, e.g., $c = -1, 2, 3$.

The AKE protocol, $\Pi_{(a+x)(b+y),(ca+x)(cb+y)}$, constructed from these admissible polynomials requires 2 exponential operations (excluding the exponentiation for the ephemeral public key) and 2 shared values. Furthermore, the AKE protocol, $\Pi_{(a+x)(b+y),(-a+x)(-b+y)}$, where $c = -1$, is suitable for an elliptic curve where the inverse operation is efficiently performed, the AKE protocol, $\Pi_{(a+x)(b+y),(2a+x)(2b+y)}$, where $c = 2$, is suitable for an elliptic curve over $\mathbb{F}_{2^n}$ where the square operation is efficiently performed, and the AKE protocol, $\Pi_{(a+x)(b+y),(3a+x)(3b+y)}$, where $c = 3$, is suitable for an elliptic curve over $\mathbb{F}_{3^n}$ where the cubic operation is efficiently performed.

Example 3. The third example of admissible polynomials is

$$m = 4,\ p_1(a,x,b,y) = ab,\ p_2(a,x,b,y) = ay,\ p_3(a,x,b,y) = xb,\ p_4(a,x,b,y) = xy.$$

The AKE protocol, $\Pi_{ab,ay,xb,xy}$, constructed from these admissible polynomials requires 4 exponential operations (excluding the exponentiation for the ephemeral public key) and 4 shared values.

Example 4. The fourth example of admissible polynomials is

$$m = 2,\ p_1(a,x,b,y) = ab + xy,\ p_2(a,x,b,y) = ay + xb.$$

The AKE protocol, $\Pi_{ab+xy,ay+xb}$, constructed from these admissible polynomials requires 4 exponential operations (excluding the exponentiation for the ephemeral public key) and 2 shared values.

3.2 Proposed AKE Protocol

We describe the proposed AKE protocol using admissible polynomials p_i $(i = 1, ..., m)$. For each set of admissible polynomials p_i $(i = 1, ..., m)$, we have concrete instance $\Pi_{p_1,...,p_m}$ of an AKE protocol.

Let κ be the security parameter, G be a cyclic group with generator g and order a κ-bit prime q, and $H : \{0,1\}^* \to \{0,1\}^\kappa$ be a cryptographic hash function modeled as a random oracle. Let p_i $(i = 1, ..., m)$ be admissible polynomials. We denote the protocol identifier of the AKE protocol $\Pi_{p_1,...,p_m}$ as Π. These are provided as part of the system parameters.

User I's static private and public keys are $a \in_U \mathbb{Z}_q$ and $A = g^a \in G$. Similarly, user R's static private and public keys are $b \in_U \mathbb{Z}_q$ and $A = g^b \in G$.

In the description, user I is the session initiator and user R is the session responder.

1. I selects a random ephemeral private key $x \in_U \mathbb{Z}_q$, computes the ephemeral public key $X = g^x$, and sends (Π, R, I, X) to R.
2. Upon receiving (Π, R, I, X), R selects a random ephemeral private key $y \in_U \mathbb{Z}_q$, computes the ephemeral public key $Y = g^y$, and sends (Π, I, R, X, Y) to I.

   R computes m shared values

   $$Z_i = A^{c_{i,0,0}b + c_{i,0,1}y} X^{c_{i,1,0}b + c_{i,1,1}y} \quad (i = 1, ..., m),$$

   computes the session key $K = H(Z_1, ..., Z_m, \Pi, I, R, X, Y)$, and completes the session.
3. Upon receiving (Π, I, R, X, Y), I checks if I has sent (Π, R, I, X) to R or not, and aborts the session if not.

   I computes m shared values

   $$Z_i = B^{c_{i,0,0}a + c_{i,1,0}x} Y^{c_{i,0,1}a + c_{i,1,1}x} \quad (i = 1, ..., m),$$

   computes the session key $K = H(Z_1, ..., Z_m, \Pi, I, R, X, Y)$, and completes the session.

Both parties compute the same shared values $Z_i = g^{p_i(a,x,b,y)}$ $(i = 1, ..., m)$ and compute the same session key K.

The proposed AKE protocol requires $2m$ exponential operations (excluding the exponentiation for the ephemeral public key) and m shared values.

3.3 Security

We need the gap Diffie-Hellman (GDH) assumption, where one tries to compute $\mathrm{CDH}(U, V)$ by accessing the DDH oracle. We denote $\mathrm{CDH}(U, V) = g^{\log U \log V}$, and the DDH oracle on input (g^u, g^v, g^x) returns bit 1 if $uv = x$, or bit 0 otherwise.

The proposed AKE protocol is secure in the eCK-security model under the GDH assumption and in the random oracle model.

Theorem 1. *If G is a group where the GDH assumption holds and H is a random oracle, the proposed AKE protocol is secure in the eCK model described in Section 2.*

The proof is provided in Appendix A. We provide a rough discussion here.

From the first condition of admissible polynomials, both users can compute the shared values as follows. User I, who knows secret keys a, x, can compute

$$\begin{pmatrix} s_i & t_i \end{pmatrix} = \begin{pmatrix} a & x \end{pmatrix} \begin{pmatrix} c_{i,0,0} & c_{i,0,1} \\ c_{i,1,0} & c_{i,1,1} \end{pmatrix}$$

and shared values as $Z_i = g^{p_i(a,x,b,y)} = B^{s_i} Y^{t_i}$. User R, who knows secret keys b, y, can compute

$$\begin{pmatrix} s'_i \\ t'_i \end{pmatrix} = \begin{pmatrix} c_{i,0,0} & c_{i,0,1} \\ c_{i,1,0} & c_{i,1,1} \end{pmatrix} \begin{pmatrix} b \\ y \end{pmatrix}$$

and shared values as $Z_i = g^{p_i(a,x,b,y)} = A^{s'_i} X^{t'_i}$.

The GDH solver $\mathcal{S}$ extracts the answer g^{uv} of an instance $(U = g^u, V = g^v)$ of the GDH problem using adversary $\mathcal{A}$. For instance, we assume that test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ queries $\mathsf{StaticKeyReveal}(I)$, and adversary $\mathcal{A}$ does not query $\mathsf{EphemeralKeyReveal}(\mathtt{sid}^*)$ and $\mathsf{StaticKeyReveal}(R)$ from the condition of freshness. In this case, solver $\mathcal{S}$ embeds the instance as $X = U$ $(= g^u)$ and $B = V$ $(= g^v)$ and extracts g^{uv} from the shared values $Z_i = g^{p_i}$ $(i = 1, ..., m)$. Solver $\mathcal{S}$ randomly selects static private key $a \in_U \mathbb{Z}_q$ and computes static public key $A = g^a \in G$.

From the second condition of admissible polynomials, solver $\mathcal{S}$ can extract the answer of the GDH instance as follows. From the second condition, there exist i, j $(1 \le i, j \le m)$, s.t. $(c_{i,1,0}, c_{i,1,1})$ and $(c_{j,1,0}, c_{j,1,1})$ are linearly independent. Using static private key a, solver $\mathcal{S}$ can compute

$$Z'_i = g^{c_{i,1,0}xb + c_{i,1,1}xy} = Z_i / (B^{c_{i,0,0}a} Y^{c_{i,0,1}a}),$$

$$Z'_j = g^{c_{j,1,0}xb + c_{j,1,1}xy} = Z_j / (B^{c_{j,0,0}a} Y^{c_{j,0,1}a}).$$

Solver $\mathcal{S}$ can compute the answer g^{uv} of the GDH instance from Z'_i, Z'_j as

$$({Z'}_i^{c_{j,1,1}}/{Z'}_j^{c_{i,1,1}})^{1/(c_{i,1,0}c_{j,1,1}-c_{j,1,0}c_{i,1,1})} = g^{xb} = g^{uv}$$

since $(c_{i,1,0}, c_{i,1,1})$ and $(c_{j,1,0}, c_{j,1,1})$ are linearly independent.

From the third condition of admissible polynomials, solver $\mathcal{S}$ can check if the shared values are correctly formed w.r.t. the static and ephemeral public keys, and can simulate H and $\mathsf{SessionKeyReveal}$ queries consistently, i.e., in the simulation of the $H(Z_1, ..., Z_m, \Pi, I, R, X, Y)$ query, solver $\mathcal{S}$ needs to check that the shared values Z_i $(i = 1, ..., m)$ are correctly formed, and if so return session key K being consistent with the previously answered $\mathsf{SessionKeyReveal}(\Pi, \mathcal{I}, I, R, X, Y)$ and $\mathsf{SessionKeyReveal}(\Pi, \mathcal{R}, R, I, X, Y)$ queries. For all i $(= 1, ..., m)$, solver $\mathcal{S}$ performs the following procedure. If condition a) of the third condition holds, $p_i(u_0, u_1, v_0, v_1) = \ell_i(u_0, u_1)\ell'_i(v_0, v_1)$, where $\ell_i(u_0, u_1)$ and $\ell'_i(v_0, v_1)$ are linear combinations of u_0, u_1 and v_0, v_1, respectively. Then, solver $\mathcal{S}$ can check if shared value Z_i is correctly formed w.r.t. the static and ephemeral public keys by verifying

$$\mathrm{DDH}(g^{\ell_i(a,x)}, g^{\ell'_i(b,y)}, Z_i) = 1.$$

Here solver $\mathcal{S}$ can compute $g^{\ell_i(a,x)} = A^{d_a}X^{d_x}, g^{\ell'_i(b,y)} = B^{d_b}Y^{d_y}$ since $\ell_i(a, x)$, $\ell'_i(b, y)$ are linear, that is, they are expressed as $\ell_i(a, x) = d_a a + d_x x$, $\ell'_i(b, y) = d_b b + d_y y$. Otherwise, from condition b) of the third condition, $c_{i,f,0}u_f v_0 + c_{i,f,1}u_f v_1 = \ell_{i,f,*}(u_0, u_1)\ell'_{i,f,*}(v_0, v_1)$, where $\ell_{i,f,*}(u_0, u_1)$ and $\ell'_{i,f,*}(v_0, v_1)$ are linear combinations of u_0, u_1 and v_0, v_1, respectively. Using static private key a, solver $\mathcal{S}$ can compute

$$Z'_i = g^{c_{i,1,0}xb + c_{i,1,1}xy} = Z_i/(B^{c_{i,0,0}a}Y^{c_{i,0,1}a}).$$

Then, solver $\mathcal{S}$ can check if shared value Z'_i is correctly formed w.r.t. the static and ephemeral public keys, by verifying

$$\mathrm{DDH}(g^{\ell_{i,1,*}(a,x)}, g^{\ell'_{i,1,*}(b,y)}, Z'_i) = 1,$$

and this implies Z_i is correctly formed. Here solver $\mathcal{S}$ can compute $g^{\ell_{i,1,*}(a,x)} = A^{d_a}X^{d_x}, g^{\ell'_{i,1,*}(b,y)} = B^{d_b}Y^{d_y}$ since $\ell_{i,1,*}(a, x)$, $\ell'_{i,1,*}(b, y)$ are linear, that is, they are expressed as $\ell_{i,1,*}(a, x) = d_a a + d_x x$, $\ell'_{i,1,*}(b, y) = d_b b + d_y y$.

4 Comparison

In this section, we compare instantiations on our construction with other related PKI-based two-pass AKE protocols in terms of efficiency and security. In Table 1, the comparisons from the points of computational and storage efficiency are provided. In the table, the number #Hash of hash functions, the number #sPK of static public keys in terms of group elements, the number #ePK of ephemeral public keys in terms of group elements, the number #SV of shared values in terms of group elements, and the number #Exp of exponentiation in the cyclic group are described.

Table 1. Protocol Comparison (Efficiency)

Protocol	#Hash	#sPK	#ePK	#SV	#Exp
HMQV [12]	2	1	1	1	2.5 (2.17)
CMQV [28]	3	1	1	1	3 (2.17)
FHMQV [27]	2	1	1	1	3 (2.17)
NAXOS [14]	2	1	1	3	4 (3.17)
NETS [17]	2	1	1	2	3
UP [29]	2	1	1	2	3.5 (3.17)
SMEN$^-$ [30]	1	2	2	1	6 (2.46)
KFU1 [11]	1	2	1	2	3
$\Pi_{p_1,\ldots,p_m}$	1	1	1	m	$2m+1$ $(1.17m+1)$
$\Pi_{(a+x)(b+y),ay+xb}$	1	1	1	2	4 (3.17)
$\Pi_{(a+x)(b+y),(ca+x)(cb+y)}$	1	1	1	2	3
NAXOS+ [16]	2	1	1	4	5 (3.34)
HC [10]	2	2	1	4	5 (4.17)
KFU2 [11]	1	2	1	4	5 (3.34)
Okamoto [24]	3	2	3	1	8 (4.14)
MO [23]	2	6	3	1	12 (5.51)

Table 2. Protocol Comparison (Security)

Protocol	Security Model	Forking Lemma	Assumption
HMQV [12]	CK+KCI+wPFS+LEP	required	KEA1, GDH, RO
CMQV [28]	eCK	required	GDH, RO
FHMQV [27]	eCK	required	GDH, RO
NAXOS [14]	eCK	not required	GDH, RO
NETS [17]	eCK	not required	GDH, RO
UP [29]	eCK	not required	GDH, RO
SMEN$^-$ [30]	eCK	not required	GDH, RO
KFU1 [11]	eCK	not required	GDH, RO
$\Pi_{p_1,\ldots,p_m}$	eCK	not required	GDH, RO
$\Pi_{(a+x)(b+y),ay+xb}$	eCK	not required	GDH, RO
$\Pi_{(a+x)(b+y),(ca+x)(cb+y)}$	eCK	not required	GDH, RO
NAXOS+ [16]	eCK	not required	CDH, RO
HC [10]	eCK	not required	CDH, RO
KFU2 [11]	eCK	not required	CDH, RO
Okamoto [24]	eCK	not required	DDH, πPRF
MO [23]	eCK	not required	DDH, πPRF

The number #Exp of exponentiation includes the exponentiation to compute the ephemeral public key. For instance, in the protocol $\Pi_{(a+x)(b+y),ay+xb}$, four exponentiations are needed, which includes one exponentiation for the ephemeral public key $Y = g^y$, one exponentiation for shared value $(AX)^{b+y} = g^{(a+x)(b+y)}$, and two exponentiations for shared value $A^y X^b = g^{ay+xb}$. Besides the naive group exponentiations count, the numbers in parentheses reflect exponentiations using speedup techniques from [18, §2.3] and [20, Alg. 14.88]. The reduced numbers for (i) HMQV [12], CMQV [28], FHMQV [27], Okamoto's protocol [24], MO [23], UP [29], $\Pi_{p_1,\ldots,p_m}$, and $\Pi_{(a+x)(b+y),ay+xb}$ can result by applying simultaneous exponentiation [20, Alg. 14.88]; and for (ii) NAXOS [14], NAXOS+ [16], HC [10], and KFU2 [11], which have the same base, can result by applying the Right-to-Left binary method. For instance, in the protocol $\Pi_{(a+x)(b+y),ay+xb}$, 3.17 exponentiations are needed, which includes one exponentiation for the ephemeral public key $Y = g^y$, one exponentiation for shared value $(AX)^{b+y} = g^{(a+x)(b+y)}$, 1.17 exponentiations for shared value $A^y X^b = g^{ay+xb}$ using speedup techniques.

We do not take public-key validation into account, since it is a necessary procedure for all protocols to prevent potential leakage of secret information, similar to invalid-curve attacks [1] and small subgroup attacks [15], see also [19,21]. Okamoto's protocol [24] and MO [23] are secure in the standard model, and the proof depends on a rather strong assumption of the existence of the πPRF family. That is, they use πPRFs instead of random oracles, and we add the number of πPRFs to #Hash.

In Table 2, the security properties of the protocols are provided. When the security proof requires the Forking Lemma [26] in the protocol, "required" is indicated in the entry, if not, "not required" is. All protocols are eCK-secure except for HMQV [12], which is a modification of MQV [13]. HMQV is also secure in a modified CK [7] model and has additional security properties such as resistance to KCI attack, wPFS, and LEP under the GDH and Knowledge of Exponent assumptions (KEA1) [3].

In each security proof of HMQV, CMQV, and FHMQV, the reduction argument is less tight since the Forking Lemma is essential for the arguments. In comparison, the rest of the protocols in Table 2, including $\Pi_{p_1,\ldots,p_m}$, $\Pi_{(a+x)(b+y),ay+xb}$, and $\Pi_{(a+x)(b+y),(ca+x)(cb+y)}$, have tighter security reductions without the Forking Lemma, and so can use shorter keys.

From the view point of the number of static and ephemeral keys, which affect the size of storage and communicated massage, our protocols $\Pi_{(a+x)(b+y),ay+xb}$ and $\Pi_{(a+x)(b+y),(ca+x)(cb+y)}$ require one static key and one ephemeral key, which is comparable with the best existing protocols.

Moreover, our protocols $\Pi_{(a+x)(b+y),ay+xb}$ and $\Pi_{(a+x)(b+y),(ca+x)(cb+y)}$ require two shared values and one hash function, which is comparable with existing protocols. The number of shared values affects the computational cost for the key derivation hash function, and the number of the hash functions affects implementation and computational costs for hash functions.

It is easy to see that all protocols eCK-secure under the CDH assumption and all protocols eCK-secure in the standard model need several hash functions or several static keys. In addition, they require much computation rather than other protocols.

In UP [29], the shared values are computed as $Z_1 = (YB)^{(x+a)}$ and $Z_2 = (YB^E)^{(x+Da)}$, where $D = H'(X)$, $E = H'(Y)$, and H' is a hash function. Therefore, $\Pi_{(a+x)(b+y),(ca+x)(cb+y)}$ can be regarded as a constant version of UP, where the output of the hash function H' is a constant c. When $c = -1$, it is suitable for the protocol constructed on a group where the inversion is easily computed in the group. An additive group over an elliptic curve is an example of such a group as the inversion operation is done with changing the sign of the y-coordinate. When $c = 2$, it is suitable for the protocol constructed on a group where doubling is easily computed in the group. A multiplicative group on a finite field is an example of such a group as the doubling operation is done with one multiplication. It is worth noting that $\Pi_{(a+x)(b+y),ay+xb}$ seems not to be regarded as a constant version of UP.

It is worthy to note that the security of UP is proved in the model which is an extension of the eCK model for timing attack [22]. In the timing attack, the adversary is additionally allowed to issue a query to obtain the ephemeral public key that will be used in next session. Our protocols are also expected to be secure in the model since the GDH solver can simulates the added queries with a ephemeral key list prepared in advance and all other queries as shown in the case of the eCK model.

Like other AKE protocols under the CDH assumption, such as NAXOS+ [16], HC [10], and KFU2 [11], it would be possible to modify our protocols to be secure under the CDH assumption by using the twin Diffie-Hellman technique [8]. Although this modification may bring about more keys, more shared values or more computation, the protocol would be practical.

5 Conclusion

We presented a sufficient condition for constructing eCK-secure two-pass AKE protocols with a single static key and a single ephemeral key using a single hash function. The constructed protocols consist of several two-dimensional versions of the DH protocols, and their security proofs do not depend on the Forking Lemma. As a result, our protocols provide strong security assurances without compromising too much on efficiency.

References

1. Antipa, A., Brown, D., Menezes, A., Struik, R., Vanstone, S.: Validation of elliptic curve public keys. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 211–223. Springer, Heidelberg (2002)
2. Bao, F., Deng, R.H., Zhu, H.: Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003)

3. 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)
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.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS 1993, pp. 62–73 (1993)
6. Bellare, M., Rogaway, P.: Minimizing the use of random oracles in authenticated encryption schemes. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 1–16. Springer, Heidelberg (1997)
7. 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)
8. 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)
9. Diffie, W., Hellman, H.: New directions in cryptography. IEEE transactions of Information Theory 22(6), 644–654 (1976)
10. Huang, H., Cao, Z.: Strongly secure authenticated key exchange protocol based on computational Diffie-Hellman problem. In: Yung, M., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, Springer, Heidelberg (2009)
11. Kim, M., Fujioka, A., Ustaoğlu, B.: Strongly secure authenticated key exchange without NAXOS' approach. In: Takagi, T., Mambo, M. (eds.) IWSEC 2009. LNCS, vol. 5824, pp. 174–191. Springer, Heidelberg (2009)
12. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
13. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography 28, 119–134 (2003)
14. LaMacchia, B., 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)
15. Lim, C., Lee, P.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 249–263. Springer, Heidelberg (1997)
16. Lee, J., Park, J.: Authenticated key exchange secure under the computational Diffie-Hellman assumption, http://eprint.iacr.org/2008/344
17. Lee, J., Park, C.: An efficient key exchange protocol with a tight security reduction, http://eprint.iacr.org/2008/345
18. M'Raïhi, D., Naccache, D.: Batch exponentiation: a fast DLP-based signature generation strategy. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security, CCS 1996, pp. 58–61 (1996)
19. Menezes, A.: Another look at HMQV. Journal of Mathematical Cryptology 1(1), 47–64 (2007)
20. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
21. Menezes, A., Ustaoğlu, B.: On the importance of public-key validation in the MQV and HMQV key agreement protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 133–147. Springer, Heidelberg (2006)

22. Menezes, A., Ustaoğlu, B.: Comparing the pre- and post-specified peer models for key agreement. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 53–68. Springer, Heidelberg (2008)
23. Moriyama, D., Okamoto, T.: An eCK-secure authenticated key exchange protocol without random oracles. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 154–167. Springer, Heidelberg (2009)
24. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474–484. Springer, Heidelberg (2007)
25. Okamoto, T., Pointcheval, D.: The gap-problems: a new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
26. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. of Cryptology 13(3), 361–396 (2000)
27. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A secure and efficient authenticated Diffie-Hellman protocol, `http://eprint.iacr.org/2009/408`
28. Ustaoğlu, B.: Obtaining a secure and efficient key agreement protocol for (H)MQV and NAXOS. Designs, Codes and Cryptography 46(3), 329–342 (2008), Extended version available at `http://eprint.iacr.org/2007/123`
29. Ustaoğlu, B.: Comparing *SessionStateReveal* and *EphemeralKeyReveal* for Diffie-Hellman protocols. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 183–197. Springer, Heidelberg (2009)
30. Wu, J., Ustaoğlu, B.: Efficient key exchange with tight security reduction. Technical Report CACR 2009-23, University of Waterloo (2009), `http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr2009-23.pdf`

A Proof of Theorem 1

We need the gap Diffie-Hellman (GDH) assumption in group G with generator g, where one tries to compute $\mathrm{CDH}(U, V)$ accessing the DDH oracle. Here, we denote $\mathrm{CDH}(U, V) = g^{\log U \log V}$, and the DDH oracle on input (g^u, g^v, g^x) returns bit 1 if $uv = x$, or bit 0 otherwise. We also need a variant of the GDH assumption where one tries to compute $\mathrm{CDH}(U, U)$ instead of $\mathrm{CDH}(U, V)$. We call the variant as the square GDH assumption, which is equivalent to the GDH assumption if group G has prime order q [2] as follows. Given a challenge U of the square GDH assumption, one sets $V = U^s$ for random integers $s \in_R [1, q-1]$ and can compute $\mathrm{CDH}(U, U) = \mathrm{CDH}(U, V)^{1/s}$. Given a challenge U, V of the GDH assumption, one sets $U_1 = UV, U_2 = UV^{-1}$ and can compute $\mathrm{CDH}(U, V) = (\mathrm{CDH}(U_1, U_1)/\mathrm{CDH}(U_2, U_2))^{1/4}$.

We show that if polynomially bounded adversary $\mathcal{A}$ can distinguish the session key of a fresh session from a randomly chosen session key, we can solve the GDH problem. Let κ denote the security parameter, and let $\mathcal{A}$ be a polynomial-time bounded adversary w.r.t. security parameter κ. We use adversary $\mathcal{A}$ to construct the GDH solver $\mathcal{S}$ that succeeds with non-negligible probability. Adversary $\mathcal{A}$ is said to be successful with non-negligible probability if adversary $\mathcal{A}$ wins the distinguishing game with probability $\frac{1}{2} + f(\kappa)$, where $f(\kappa)$ is non-negligible, and the event M denotes a successful adversary $\mathcal{A}$.

Let the test session be $\mathtt{sid}^* = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ or $(\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$, which is a completed session between honest users U_A and U_B, where user U_A is the initiator and user U_B is the responder of the test session $\mathtt{sid}^*$. Let H^* be the event that adversary $\mathcal{A}$ queries $(Z_1, ..., Z_m, \Pi, X_A, X_B)$ to H. Let $\overline{H^*}$ be the complement of event H^*. Let $\mathtt{sid}$ be any completed session owned by an honest user such that $\mathtt{sid} \neq \mathtt{sid}^*$ and $\mathtt{sid}$ is non-matching to $\mathtt{sid}^*$. Since $\mathtt{sid}$ and $\mathtt{sid}^*$ are distinct and non-matching, the inputs to the key derivation function H are different for $\mathtt{sid}$ and $\mathtt{sid}^*$. Since H is a random oracle, adversary $\mathcal{A}$ cannot obtain any information about the test session key from the session keys of non-matching sessions. Hence, $\Pr(M \wedge \overline{H^*}) \leq \frac{1}{2}$ and $\Pr(M) = \Pr(M \wedge H^*) + \Pr(M \wedge \overline{H^*}) \leq \Pr(M \wedge H^*) + \frac{1}{2}$, whence $f(\kappa) \leq \Pr(M \wedge H^*)$. Henceforth, the event $M \wedge H^*$ is denoted by M^*.

We denote a user as U_i, and user U_i and other parties are modeled as probabilistic polynomial-time Turing machines w.r.t. security parameter κ. For user U_i, we denote static secret (public) keys as s_i (S_i) and ephemeral secret (public) keys as x_i (X_i, respectively). We also denote the session key as K. Assume that adversary $\mathcal{A}$ succeeds in an environment with n users and activates at most s sessions within a user.

We consider the non-exclusive classification of all possible events in Tables 3 and 4. Here, users U_A and U_B are initiator and responder of the test session $\mathtt{sid}^*$, respectively. We denote the static and ephemeral keys as $a = s_A, A = S_A, x = x_A, X = X_A$, $b = s_B, B = S_B, y = x_B, Y = X_B$ for the test session $\mathtt{sid}^*$. Table 3 classifies events when static public keys A, B are distinct, and Table 4 classifies events when static public keys $A = B$ are the same. In these tables, "ok" means the static key is not revealed, or the matching session exists and the ephemeral key is not revealed. "r" means the static or ephemeral key may be revealed. "n" means no matching session exists. The "instance embedding" row shows how the simulator embeds an instance of the GDH problem. The "succ. prob." row

Table 3. Classification of attacks when static public keys A, B are distinct. "ok" means the static key is not revealed, or the matching session exists and the ephemeral key is not revealed. "r" means the static or ephemeral key may be revealed. "n" means no matching session exists. The "instance embedding" row shows how the simulator embeds an instance of the GDH problem. The "succ. prob." row shows the probability of success of the simulator, where $p_i = Pr(E_i \wedge M^*)$ and n and s are the numbers of parties and sessions.

	$a = s_A$	$x = x_A$	$b = s_B$	$y = x_B$	instance embedding	succ. prob.
E_1	r	ok	ok	n	$X = U, B = V$	$p_1/n^2 s$
E_2	ok	r	ok	n	$A = U, B = V$	p_2/n^2
E_3	r	ok	ok	r	$X = U, B = V$	$p_3/n^2 s$
E_4	ok	r	ok	r	$A = U, B = V$	p_4/n^2
E_5	r	ok	r	ok	$X = U, Y = V$	$p_5/n^2 s^2$
E_6	ok	r	r	ok	$A = U, Y = V$	$p_6/n^2 s$

Table 4. Classification of attacks when static public keys $A = B$ are the same

	$a = s_A$	$x = x_A$	$a = s_A$	$y = x_B$	instance embedding	succ. prob.
E'_2	ok	r	ok	n	$A = B = U$	p_2/n^2
E'_4	ok	r	ok	r	$A = B = U$	p_4/n^2
E'_5	r	ok	r	ok	$X = U, Y = V$	p_5/n^2s^2

shows the probability of success of the simulator, where $p_i = Pr(E_i \wedge M^*)$ and n and s are the numbers of parties and sessions.

Since the classification covers all possible events, at least one event $E_i \wedge M^*$ in the tables occurs with non-negligible probability if event M^* occurs with non-negligible probability. Thus, the GDH problem can be solved with non-negligible probability, which means that the proposed protocol is secure under the GDH assumption. We investigate each of these events in the following subsections.

A.1 Event $E_1 \wedge M^*$

In event E_1, test session $\mathtt{sid}^*$ has no matching session, adversary $\mathcal{A}$ may query $\mathsf{StaticKeyReveal}(U_A)$, and adversary $\mathcal{A}$ does not query either $\mathsf{EphemeralKeyReveal}$ $(\mathtt{sid}^*)$ or $\mathsf{StaticKeyReveal}(U_B)$. We embed the instance as $X = U$ $(= g^u)$ and $B = V$ $(= g^v)$ and extract g^{uv} from $Z_i = g^{p_i}$ $(i = 1, ..., m)$. In event $E_1 \wedge M^*$, solver $\mathcal{S}$ performs the following steps.

Setup. The GDH solver $\mathcal{S}$ establishes n honest users $U_1, ..., U_n$, randomly selects static secret key $s_i \in_U \mathbb{Z}_q$, computes static public key $S_i = g^{s_i} \in G$, and assigns static secret and public keys (s_i, S_i) to user U_i. In addition to the above steps, solver $\mathcal{S}$ embeds instance $(U = g^u, V = g^v)$ of the GDH problem as follows.

Solver $\mathcal{S}$ randomly selects two users U_A and U_B and integer $t \in_R [1, s]$, which is a guess of the test session with probability $1/n^2 s$. Solver $\mathcal{S}$ sets the ephemeral public key of t-th session of user U_A as $X = U$ $(= g^u)$, and sets the static public key of user U_B as $B = V$ $(= g^v)$, Solver $\mathcal{S}$ randomly selects static secret key $a \in_U \mathbb{Z}_q$ and computes static public key $A = g^a \in G$.

Solver $\mathcal{S}$ activates adversary $\mathcal{A}$ on this set of users and awaits the actions of adversary $\mathcal{A}$. We next describe the actions of $\mathcal{S}$ in response to user activations and oracle queries.

Simulation. Solver $\mathcal{S}$ maintains a list L_H that contains queries and answers of H oracle, and a list L_S that contains queries and answers of $\mathsf{SessionKeyReveal}$. Solver $\mathcal{S}$ simulates oracle queries as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_i, U_j)$: $\mathcal{S}$ selects ephemeral secret key $x \in_U \mathbb{Z}_q$, computes ephemeral public key X_i honestly, records (Π, U_i, U_j, X_i), and returns it.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_j, U_i, X_i)$: $\mathcal{S}$ selects ephemeral secret key $y \in_U \mathbb{Z}_q$, computes ephemeral public key X_j honestly, records $(\Pi, U_i, U_j, X_i, X_j)$. and returns it.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_i, U_j, X_i, X_j)$: If (Π, U_i, U_j, X_i) is not recorded, $\mathcal{S}$ records the session $(\Pi, \mathcal{I}, U_i, U_j, X_i, X_j)$ as not completed. Otherwise, $\mathcal{S}$ records the session as completed.

4. $H(Z_1, ..., Z_m, \Pi, U_i, U_j, X_i, X_j)$:
   (a) If $(Z_1, ..., Z_m, \Pi, U_i, U_j, X_i, X_j)$ is recorded in list L_H, then return recorded value K.
   (b) Else if the session $(\Pi, \mathcal{I}, U_i, U_j, X_i, X_j)$ or $(\Pi, \mathcal{R}, U_j, U_i, X_i, X_j)$ is recorded in list L_S, then $\mathcal{S}$ checks that the shared values Z_i $(i = 1, ..., m)$ are correctly formed w.r.t. static and ephemeral public keys S_i, S_j, X_i, X_j using knowledge of secret keys s_i or x_i by the procedure Check described below.

       If the shared values are correctly formed, then return recorded value K and record it in list L_H.
   (c) Else if $i = A, j = B$, and the session is t-th session of user U_A, then $\mathcal{S}$ checks that the shared values Z_i $(i = 1, ..., m)$ are correctly formed w.r.t. static and ephemeral public keys A, B, X, Y using knowledge of secret key a by the procedure Check described below.

       If the shared values are correctly formed, then $\mathcal{S}$ computes the answer of the GDH instance from the shared values and public keys using knowledge of secret key a by the procedure Extract described below, and is successful by outputting the answer.
   (d) Otherwise, $\mathcal{S}$ returns random value K and records it in list L_H.
5. SessionKeyReveal($(\Pi, \mathcal{I}, U_i, U_j, X_i, X_j)$ or $(\Pi, \mathcal{R}, U_j, U_i, X_i, X_j)$):
   (a) If the session $(\Pi, \mathcal{I}, U_i, U_j, X_i, X_j)$ or $(\Pi, \mathcal{R}, U_j, U_i, X_i, X_j)$ (= `sid`) is not completed, return error.
   (b) Else if `sid` is recorded in list L_S, then return recorded value K.
   (c) Else if $(Z_1, ..., Z_m, \Pi, U_i, U_j, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ checks that the shared values Z_i $(i = 1, ..., m)$ are correctly formed w.r.t. static and ephemeral public keys S_i, S_j, X_i, X_j using knowledge of secret keys s_i or x_i by the procedure Check described below.

       If the shared values are correctly formed, then return recorded value K and record it in list L_S.
   (d) Otherwise, $\mathcal{S}$ returns random value K and records it in list L_S.
6. EphemeralKeyReveal(`sid`): If the ephemeral public key X of the session `sid` is U, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ returns ephemeral secret key x selected in Setup.
7. StaticKeyReveal(U_i): If $i = B$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ returns ephemeral secret key s selected in Setup.
8. EstablishParty(U_i, S_i): $\mathcal{S}$ responds to the query faithfully.
9. Test(`sid`): If ephemeral public key X is not U or static public key B is not V in session `sid`, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ responds to the query faithfully.
10. If adversary $\mathcal{A}$ outputs a guess γ, $\mathcal{S}$ aborts with failure.

Extract : The procedure Extract computes $g^{u_{\overline{f}} v_0}$ from the shared values $Z_i = g^{p_i(u_0, u_1, v_0, v_1)}$ $(i = 1, ..., m)$ and public keys $U_0 = g^{u_0}, U_1 = g^{u_1}, V_0 = g^{v_0}, V_1 = g^{v_1}$ using knowledge of secret key u_f as follows.

From the second condition of admissible polynomials, there exist i, j $(1 \leq i, j \leq m)$, s.t. $(c_{i,\overline{f},0}, c_{i,\overline{f},1})$ and $(c_{j,\overline{f},0}, c_{j,\overline{f},1})$ are linearly independent. Using u_f, the procedure Extract computes

$$Z'_i = g^{c_{i,\overline{f},0} u_{\overline{f}} v_0 + c_{i,\overline{f},1} u_{\overline{f}} v_1} = Z_i / (V_0^{c_{i,f,0} u_f} V_1^{c_{i,f,1} u_f}),$$

$$Z'_j = g^{c_{j,\overline{f},0} u_{\overline{f}} v_0 + c_{j,\overline{f},1} u_{\overline{f}} v_1} = Z_j / (V_0^{c_{j,f,0} u_f} V_1^{c_{j,f,1} u_f}).$$

The procedure Extract computes $g^{u_{\overline{f}} v_0}$ from Z'_i, Z'_j as

$$({Z'}_i^{c_{j,\overline{f},1}} / {Z'}_j^{c_{i,\overline{f},1}})^{1/(c_{i,\overline{f},0} c_{j,\overline{f},1} - c_{j,\overline{f},0} c_{i,\overline{f},1})} = g^{u_{\overline{f}} v_0}$$

since $(c_{i,\overline{f},0}, c_{i,\overline{f},1})$ and $(c_{j,\overline{f},0}, c_{j,\overline{f},1})$ are linearly independent.

The procedure Extract can compute $g^{u_{\overline{f}} v_1}$ using knowledge of secret key u_f same as above. The procedure Extract can compute $g^{u_f v_{\overline{f'}}}$ $(f = 0, 1)$ using knowledge of secret key $u_{f'}$ same as above.

Check : The procedure Check checks that the shared values $Z_i = g^{p_i(u_0,u_1,v_0,v_1)}$ $(i = 1, ..., m)$ are correctly formed w.r.t. public keys $U_0 = g^{u_0}, U_1 = g^{u_1}, V_0 = g^{v_0}, V_1 = g^{v_1}$ using knowledge of secret key u_f as follows.

For all i $(= 1, ..., m)$, the procedure Check performs the following. If condition a) of the second condition of admissible polynomials holds, there exist linear combination $\ell_i(u_0, u_1)$ of u_0, u_1 and linear combination $\ell'_i(v_0, v_1)$ of v_0, v_1, s.t. $p_i(u_0, u_1, v_0, v_1) = \ell_i(u_0, u_1)\ell'_i(v_0, v_1)$. Then, the procedure Check checks if shared value Z_i is correctly formed w.r.t. public keys by verifying

$$\mathrm{DDH}(g^{\ell_i(u_0,u_1)}, g^{\ell'_i(v_0,v_1)}, Z_i) = 1.$$

Here, we can compute $g^{\ell_i(u_0,u_1)} = U_0^{d_{u_0}} U_1^{d_{u_1}}$, $g^{\ell'_i(v_0,v_1)} = V_0^{d_{v_0}} V_1^{d_{v_1}}$ since $\ell_i(u_0, u_1)$ and $\ell'_i(v_0, v_1)$ are expressed as $\ell_i(u_0, u_1) = d_{u_0} u_0 + d_{u_1} u_1$ and $\ell'_i(v_0, v_1) = d_{v_0} v_0 + d_{v_1} v_1$.

Otherwise, from condition b) of the second condition of admissible polynomials, there exist linear combination $\ell_{i,\overline{f},*}(u_0, u_1)$ of u_0, u_1 and linear combination $\ell'_{i,\overline{f},*}(v_0, v_1)$ of v_0, v_1, s.t. $c_{i,\overline{f},0} u_{\overline{f}} v_0 + c_{i,\overline{f},1} u_{\overline{f}} v_1 = \ell_{i,\overline{f},*}(u_0, u_1)\ell'_{i,\overline{f},*}(v_0, v_1)$. Using knowledge of secret key u_f, the procedure Check computes

$$Z'_i = g^{c_{i,\overline{f},0} u_{\overline{f}} v_0 + c_{i,\overline{f},1} u_{\overline{f}} v_1} = Z_i / (V_0^{c_{i,f,0} u_f} V_1^{c_{i,f,1} u_f}).$$

Then, the procedure Check checks that shared value Z'_i is correctly formed w.r.t. public keys by verifying

$$\mathrm{DDH}(g^{\ell_{i,\overline{f},*}(u_0,u_1)}, g^{\ell'_{i,\overline{f},*}(v_0,v_1)}, Z'_i) = 1,$$

and this implies that shared value Z_i is correctly formed w.r.t. public keys. Here, we can compute $g^{\ell_{i,\overline{f},*}(u_0,u_1)} = U_0^{d_{u_0}} U_1^{d_{u_1}}$, $g^{\ell'_{i,\overline{f},*}(v_0,v_1)} = V_0^{d_{v_0}} V_1^{d_{v_1}}$ since $\ell_{i,\overline{f},*}(u_0, u_1)$ and $\ell'_{i,\overline{f},*}(v_0, v_1)$ are expressed as $\ell_{i,\overline{f},*}(u_0, u_1) = d_{u_0} u_0 + d_{u_1} u_1$ and $\ell'_{i,\overline{f},*}(v_0, v_1) = d_{v_0} v_0 + d_{v_1} v_1$.

The procedure Check can check that the shared values $Z_i = g^{p_i(u_0,u_1,v_0,v_1)}$ $(i = 1, ..., m)$ are correctly formed w.r.t. public keys $U_0 = g^{u_0}, U_1 = g^{u_1}, V_0 = g^{v_0}, V_1 = g^{v_1}$ using knowledge of secret key v_f same as above.

Analysis. The simulation of the environment for adversary $\mathcal{A}$ is perfect except with negligible probability. The probability that adversary $\mathcal{A}$ selects the session, where ephemeral public key X is U and static public key B is V, as the test session $\mathtt{sid}^*$ is at least $\frac{1}{n^2 s}$. Suppose this is indeed the case, solver $\mathcal{S}$ does not abort in Step 9.

Suppose event E_1 occurs, solver $\mathcal{S}$ does not abort in Steps 6 and 7.

Suppose event M^* occurs, adversary $\mathcal{A}$ queries correctly formed $Z_1, ..., Z_m$ to H. Therefore, solver $\mathcal{S}$ is successful as described in Step 4c, and does not abort as in Step 10.

Hence, solver $\mathcal{S}$ is successful with probability $Pr(S) \geq \frac{p_1}{n^2 s}$, where p_1 is probability that $E_1 \wedge M^*$ occurs.

A.2 Event $E_2 \wedge M^*$

In event E_2, test session $\mathtt{sid}^*$ has no matching session ($\overline{\mathtt{sid}^*}$), adversary $\mathcal{A}$ may query $\mathsf{EphemeralKeyReveal}(\mathtt{sid}^*)$, and adversary $\mathcal{A}$ does not query either $\mathsf{StaticKeyReveal}(U_A)$ or $\mathsf{StaticKeyReveal}(U_B)$. Solver $\mathcal{S}$ performs the same reduction as in event $E_1 \wedge M^*$ in Subsection A.1, except the following points.

In Setup, solver $\mathcal{S}$ embeds GDH instance (U, V) as $A = U, B = V$, solver $\mathcal{S}$ randomly selects static secret key $x \in_U \mathbb{Z}_q$ and computes ephemeral public key $X = g^x \in G$.

In Simulation, using knowledge of x, solver $\mathcal{S}$ extracts the answer of the GDH instance and checks if the shared values are correctly formed.

A.3 Event $E_3 \wedge M^*$

In event E_3, test session $\mathtt{sid}^*$ has the matching session, $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ may query $\mathsf{StaticKeyReveal}(U_A)$ and $\mathsf{EphemeralKeyReveal}(\overline{\mathtt{sid}^*})$, and adversary $\mathcal{A}$ does not query either $\mathsf{EphemeralKeyReveal}(\mathtt{sid}^*)$ or $\mathsf{StaticKeyReveal}(U_B)$. Solver $\mathcal{S}$ performs the same reduction as in event $E_1 \wedge M^*$ in Subsection A.1, except the following points.

In Setup, solver $\mathcal{S}$ embeds GDH instance (U, V) as $X = U, B = V$, solver $\mathcal{S}$ randomly selects static secret key $a \in_U \mathbb{Z}_q$ and computes static public key $A = g^a \in G$.

In Simulation, using knowledge of a, solver $\mathcal{S}$ extracts the answer of the GDH instance and checks if the shared values are correctly formed.

A.4 Event $E_4 \wedge M^*$

In event E_4, test session $\mathtt{sid}^*$ has the matching session, $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ may query $\mathsf{EphemeralKeyReveal}(\mathtt{sid}^*)$ and $\mathsf{EphemeralKeyReveal}(\overline{\mathtt{sid}^*})$, and adversary $\mathcal{A}$ does not query either $\mathsf{StaticKeyReveal}(U_A)$ or $\mathsf{StaticKeyReveal}(U_B)$. Solver $\mathcal{S}$ performs the same reduction as in event $E_1 \wedge M^*$ in Subsection A.1, except the following points.

In Setup, solver $\mathcal{S}$ embeds GDH instance (U, V) as $A = U, B = V$, solver $\mathcal{S}$ randomly selects static secret key $x \in_U \mathbb{Z}_q$ and computes ephemeral public key $X = g^x \in G$.

In Simulation, using knowledge of x, solver $\mathcal{S}$ extracts the answer of the GDH instance and checks if the shared values are correctly formed.

A.5 Event $E_5 \wedge M^*$

In event E_5, test session $\mathtt{sid}^*$ has the matching session, $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ may query $\mathsf{StaticKeyReveal}(U_A)$ and $\mathsf{StaticKeyReveal}(U_B)$, and adversary $\mathcal{A}$ does not query either $\mathsf{EphemeralKeyReveal}(\mathtt{sid}^*)$ or $\mathsf{EphemeralKeyReveal}(\overline{\mathtt{sid}^*})$. Solver $\mathcal{S}$ performs the same reduction as in event $E_1 \wedge M^*$ in Subsection A.1, except the following points.

In Setup, solver $\mathcal{S}$ embeds GDH instance (U, V) as $X = U, Y = V$, solver $\mathcal{S}$ randomly selects static secret key $a \in_U \mathbb{Z}_q$ and computes static public key $A = g^a \in G$.

In Simulation, using knowledge of a, solver $\mathcal{S}$ extracts the answer of the GDH instance and checks if the shared values are correctly formed.

A.6 Event $E_6 \wedge M^*$

In event E_6, test session $\mathtt{sid}^*$ has the matching session, $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ may query $\mathsf{EphemeralKeyReveal}(\mathtt{sid}^*)$ and $\mathsf{StaticKeyReveal}(U_B)$, and adversary $\mathcal{A}$ does not query either $\mathsf{StaticKeyReveal}(U_A)$ or $\mathsf{EphemeralKeyReveal}(\overline{\mathtt{sid}^*})$. Solver $\mathcal{S}$ performs the same reduction as in event $E_1 \wedge M^*$ in Subsection A.1, except the following points.

In Setup, solver $\mathcal{S}$ embeds GDH instance (U, V) as $A = U, Y = V$, solver $\mathcal{S}$ randomly selects static secret key $x \in_U \mathbb{Z}_q$ and computes ephemeral public key $X = g^x \in G$.

In Simulation, using knowledge of x, solver $\mathcal{S}$ extracts the answer of the GDH instance and checks if the shared values are correctly formed.

A.7 Other Cases

Events E'_2, E'_4 in Table 4 can be handled the same as events E_2, E_4 in Table 3, with condition $A = B$ under the square GDH assumption, which is equivalent to the GDH assumption.

Event E'_5 in Table 4 can be handled the same as event E_5 in Table 3, with condition $A = B$ under the GDH assumption.

Contributory Password-Authenticated Group Key Exchange with Join Capability

Michel Abdalla[1], Céline Chevalier[2], Louis Granboulan[3], and David Pointcheval[1]

[1] ENS/CNRS/INRIA, Paris, France
[2] LSV – ENS Cachan/CNRS/INRIA*, France
[3] EADS, France

Abstract. Password-based authenticated group key exchange allows any group of users in possession of a low-entropy secret key to establish a common session key even in the presence of adversaries. In this paper, we propose a new generic construction of password-authenticated group key exchange protocol from any two-party password-authenticated key exchange with explicit authentication. Our new construction has several advantages when compared to existing solutions. First, our construction only assumes a common reference string and does not rely on any idealized models. Second, our scheme enjoys a simple and intuitive security proof in the universally composable framework and is optimal in the sense that it allows at most one password test per user instance. Third, our scheme also achieves a strong notion of security against insiders in that the adversary cannot bias the distribution of the session key as long as one of the players involved in the protocol is honest. Finally, we show how to easily extend our protocol to the dynamic case in a way that the costs of establishing a common key between two existing groups is significantly smaller than computing a common key from scratch.

1 Introduction

Password-authenticated key exchange (PAKE) allows any two parties in possession of a short (*i.e.*, low-entropy) secret key to establish a common session key even in the presence of an adversary. Since its introduction by Bellovin and Merritt [14], PAKE has become an important cryptographic primitive due to its simplicity and ease of use, which does not rely on expensive public-key infrastructures or high-entropy secret keys.

In the universally composable (UC) framework [18], the authors of [20] show how their new model (based on the ideal functionality $\mathcal{F}_{\mathsf{pwKE}}$) relates to previous PAKE models, such as [12] or [8]. In particular, they show that any protocol that realizes $\mathcal{F}_{\mathsf{pwKE}}$ is also a secure password-authenticated key-exchange protocol in the model of [12]. Other works in the UC framework include [24] and [26], where the authors study static corruptions without random oracles as well.

* Work done while being at Télécom ParisTech, Paris, France.

In this paper, we consider password-authenticated key exchange in the group setting (GPAKE) where the number of users involved in the computation of a common session key can be large. With few exceptions (*e.g.*, [1]), most protocols in this setting are built from scratch and are quite complex. Among these protocols, we can clearly identify two types of protocols: constant-round protocols (*e.g.*, [9,15,5]) and those whose number of communication rounds depends on the number of users involved in the protocol execution (*e.g.*, [16]). Since constant-round protocols are generally easier to implement and less susceptible to synchronization problems when the number of user increases, we focus our attention on these protocols. More precisely, we build upon the works of Abdalla, Catalano, Chevalier, and Pointcheval [5] and Abdalla, Bohli, González Vasco, and Steinwandt [1] and propose a new generic compiler which converts any two-party password-authenticated key exchange protocol into a password-authenticated group key exchange protocol. Like [1], our protocol relies on a common reference string (CRS) which seems to be a reasonable assumption when one uses a public software, that is somewhat "trusted". This is also a necessary assumption for realizing PAKE schemes in the UC framework as shown by [20]. Like [5], our protocol achieves a strong notion of contributiveness in the UC framework. In particular, even if it can control all the network communications, the adversary cannot bias the key as long as one of the players involved in the protocol is honest. We indeed assume that all the communications are public, and such a network can be seen as a (non-reliable) broadcast channel, controlled by the adversary: the latter can delay, block, alter and/or replay messages. Players thus do not necessarily all receive the same messages. Since the adversary can block messages, we have to assume timeouts for each round. As a consequence, denial-of-service attacks are possible, but these are out of the scope of this paper.

Contributions. There are three main contributions in this paper. The first one regards the optimality of the security, which only allows one password test per subgroup. As mentioned in [5] and in Barak *et al.* [10], without any strong authentication mechanisms, which is the case in the password-based scenario, the adversary can always partition the players into disjoint subgroups and execute independent sessions of the protocol with each subgroup, playing the role of the other players. As a result, an adversary can always use each one of these partitions to test the passwords used by each subgroup. Since this attack is unavoidable, this is the best security guarantee that we can hope for. In contrast, the protocol in [5] required an additional password test for each user in the group.

The second contribution is the construction itself, which astutely combines several techniques: it applies the Burmester-Desmedt technique [17] to any secure two-party PAKE achieving (mutual) explicit authentication in the UC framework. The key idea used by our protocol is that, in addition to establishing pairwise keys between any pair of users in the ring, each user also chooses an additional random secret value to be used in the session key generation. In order to achieve the contributiveness property, our protocol enforces these random secret values to be chosen independently so that the final session key will be uniformly distributed as long as one of the players is honest. In order to prove our protocol secure in the UC framework, we also make use of a particular randomness extractor, which possesses a type of partial invertibility property which we use in the proof. The proof of security assumes the existence of a common reference string and does not rely on any idealized model. We note that UC-secure authenticated group key exchange protocols with contributiveness were already known [25,5], but they either relied on idealized models [5] or were not applicable to the password-based scenario [25].

Our final contribution is to show how to extend our protocol to the dynamic case, with forward-secrecy, so that the cost of merging two subgroups is relatively small in comparison to generating a *new and independent* common group key from scratch. This is because given two subgroups, each with its own subgroup key, we only need to execute two instances of the PAKE protocol in order to merge these two groups and generate a new group key. Note that, if one were to compute a common group key from scratch, the number of PAKE executions would be proportional to the number of users in the group. Since the PAKE execution is the most computationally expensive part of the protocol, our new merge protocol significantly improves upon the trivial solution.

2 UC Two-Party PAKE

Notations and Security Model. We denote by k the security parameter. An event is said to be negligible if it happens with probability less than the inverse of any polynomial in k. If X is a finite set, $x \overset{R}{\leftarrow} X$ indicates the process of selecting x uniformly and at random in X (we thus implicitly assume that X can be sampled efficiently).

Throughout this paper, we assume basic familiarity with the universal composability framework. The interested reader is referred to [18,20] for details. The model considered in this paper is the UC framework with joint state proposed by Canetti and Rabin [21] (the CRS will be in the joint state).

In this paper, we consider adaptive adversaries which are allowed to arbitrarily corrupt players at any moment during the execution of the protocol, thus getting complete access to their internal memory. In a real execution of the protocol, this is modeled by letting the adversary $\mathcal{A}$ obtain the password and the internal state of the corrupted player. Moreover, $\mathcal{A}$ can arbitrarily modify the player's strategy. In an ideal execution of the protocol, the simulator $\mathcal{S}$ gets the corrupted player's password and has to simulate its internal state in a way that remains consistent to what was already provided to the environment.

Split Functionalities. Without any strong authentication mechanisms, the adversary can always partition the players into disjoint subgroups and execute independent sessions of the protocol with each subgroup, playing the role of the other players. Such an attack is unavoidable since players cannot distinguish the

Given a functionality $\mathcal{F}$, the split functionality $s\mathcal{F}$ proceeds as follows:
Initialization:

- Upon receiving (Init, sid) from party P_i, send $(\mathsf{Init}, sid, P_i)$ to the adversary.
- Upon receiving a message $(\mathsf{Init}, sid, P_i, G, H, sid_H)$ from $\mathcal{A}$, where $H \subset G$ are sets of party identities, check that P_i has already sent (Init, sid) and that for all recorded $(H', sid_{H'})$, either $H = H'$ and $sid_H = sid_{H'}$ or H and H' are disjoint and $sid_H \neq sid_{H'}$. If so, record the pair (H, sid_H), send $(\mathsf{Init}, sid, sid_H)$ to P_i, and invoke a new functionality $(\mathcal{F}, sid_H)$ denoted as $\mathcal{F}_H$ on the group G and with set of initially honest parties H.

Computation:

- Upon receiving (Input, sid, m) from party P_i, find the set H such that $P_i \in H$ and forward m to $\mathcal{F}_H$.
- Upon receiving $(\mathsf{Input}, sid, P_j, H, m)$ from $\mathcal{A}$, such that $P_j \notin H$, forward m to $\mathcal{F}_H$ as if coming from P_j (it will be ignored if $P_j \notin G$ for the functionality $\mathcal{F}_H$).
- When $\mathcal{F}_H$ generates an output m for party $P_i \in H$, send m to P_i. If the output is for $P_j \notin H$ or for the adversary, send m to the adversary.

Fig. 1. Split Functionality $s\mathcal{F}$

case in which they interact with each other from the case where they interact with the adversary. The authors of [10] addressed this issue by proposing a new model based on *split functionalities* which guarantees that this attack is the only one available to the adversary.

The split functionality is a generic construction based upon an ideal functionality. The original definition was for protocols with a fixed set of participants. Since our goal is to deal with dynamic groups, not known in advance, we let the adversary not only split the honest players into subsets H in each execution of the protocol, but also specify the players it will control. The functionality will thus start with the actual list of players in G, where H is the subgroup of the honest players in this execution. Note that H is the subset of the *initially* honest players, which can later get corrupted in case we consider adaptive adversaries. The restriction of the split functionality is to have disjoint sets H, since it models the fact that the adversary splits the honest players in several concurrent but independent executions of the protocol. The new description can be found on Figure 1. In the initialization stage, the adversary adaptively chooses disjoint subsets H of the honest parties (with a unique session identifier that is fixed for the duration of the protocol) together with the lists G of the players for each execution. More precisely, the protocol starts with a session identifier sid. Then, the initialization stage generates some random values which, combined together and with sid, create the new session identifier sid', shared by all parties which have received the same values – that is, the parties of the disjoint subsets. The important point here is that the subsets create a *partition* of the declared honest players, thus forbidding communication among the subsets. During the computation, each subset H activates a separate instance of the functionality $\mathcal{F}$ on the group G. All these functionality instances are independent: The executions of the protocol for each subset H can only be related in the way the adversary chooses the inputs of the players it controls. The parties $P_i \in H$ provide their own inputs and receive their own outputs (see the first item of "computation" in Figure 1), whereas the adversary plays the role of all the parties $P_j \notin H$, but in G (see the second item).

UC 2-PAKE Protocols. Canetti *et al.* first proposed in [20] the ideal functionality for universally composable two-party password-based key exchange (2-PAKE), along with the first protocol to achieve such a level of security. This protocol is based on the Gennaro-Lindell extension of the KOY protocol [27,23], and is not known to achieve adaptive security.

Later on, Abdalla *et al.* proposed in [4] an improvement of the ideal functionality, adding client authentication, which provides a guarantee to the server that when it accepts a key, the latter is actually known to the expected client. They also give a protocol realizing this functionality, and secure against adaptive corruptions, in the random oracle model. More recently, they presented another protocol in [7], based on the Gennaro-Lindell protocol, secure against adaptive corruptions in the standard model, but with no explicit authentication.

Mutual Authentication. Our generic compiler from a 2-PAKE to a GPAKE, that we present in Section 4, achieves security against static (*resp.* adaptive) adversaries, depending on the level of security achieved by the underlying 2-PAKE. Furthermore, the 2-PAKE needs to achieve mutual authentication. For the sake of completeness, we give here the modifications of the ideal functionality to capture this property: both client authentication and server authentication. Furthermore, to be compatible with the GPAKE functionality, we use the split functionality model. For the 2-PAKE, this model is equivalent to the use of TestPwd queries in the functionality. They both allow the adversary to test the password of a player (a dictionary attack) either by explicitly asking a TestPwd query, or by playing with this player. More precisely, an adversary willing to test the password of a player will play on behalf of its partner, with the trial password: If the execution succeeds, the password is correct. Finally, the 2-PAKE functionality with mutual authentication $\mathcal{F}_{\mathsf{PAKE}}^{MA}$, presented in Figure 2, is very close to the GPAKE functionality, see Section 3. As in the GPAKE one, we added the contributiveness property. Note that the protocols mentioned earlier can realize this functionality given very small modifications.

3 UC Group PAKE

We give here a slightly modified version of the ideal functionality for GPAKE presented in [5], by suppressing the TestPwd queries, which was left as an open problem in [5], since their protocol could not be proven without them. Our new functionality thus models the optimal security level: the adversary can test only one password per subgroup (split functionality). This is the same improvement as done in another context between [2] and [3]. Furthermore, the players in [5] were assumed to share the same passwords. We consider here a more general

The functionality $\mathcal{F}_{\mathsf{PAKE}}^{MA}$ is parameterized by a security parameter k, and the parameter $t \in \{1, 2\}$ of the contributiveness. It maintains a list L initially empty of values of the form $((\mathsf{sid}, P_k, P_l, \mathsf{pw}, \mathsf{role}), *)$ and interacts with an adversary $\mathcal{S}$ and dynamically determined parties P_i and P_j via the following queries:

- **Initialization.**
  Upon receiving a query $(\mathsf{NewSession}, \mathsf{sid}, P_i, \mathsf{pw}, \mathsf{role})$ from $P_i \in \mathcal{H}$:
  - Send $(\mathsf{NewSession}, \mathsf{sid}, P_i, \mathsf{role})$ to $\mathcal{S}$.
  - If this is the first $\mathsf{NewSession}$ query, or if it is the second one and there is a record $((\mathsf{sid}, P_j, P_i, \mathsf{pw}', \overline{\mathsf{role}}), \mathsf{fresh}) \in L$, then record $((\mathsf{sid}, P_i, P_j, \mathsf{pw}, \mathsf{role}), \mathsf{fresh})$ in L. If it is the second $\mathsf{NewSession}$ query, record the tuple $(\mathsf{sid}, \mathsf{ready})$.
- **Key Generation.** Upon receiving a message $(\mathsf{sid}, \mathsf{ok}, \mathsf{sk})$ from $\mathcal{S}$ where there exists a recorded tuple $(\mathsf{sid}, \mathsf{ready})$, then, denote by n_c the number of corrupted players, and
  - If P_i and P_j have the same password and $n_c < t$, choose $\mathsf{sk}' \in \{0, 1\}^k$ uniformly at random and store $(\mathsf{sid}, \mathsf{sk}')$. Next, mark the records $((\mathsf{sid}, P_i, P_j, \mathsf{pw}_i, \mathsf{role}), *)$ and $((\mathsf{sid}, P_j, P_i, \mathsf{pw}_j, \overline{\mathsf{role}}), *)$ $\mathsf{complete}$.
  - If P_i and P_j have the same passwords and $n_c \geq t$, store $(\mathsf{sid}, \mathsf{sk})$. Next, mark the records $((\mathsf{sid}, P_i, P_j, \mathsf{pw}_i, \mathsf{role}), *)$ and $((\mathsf{sid}, P_j, P_i, \mathsf{pw}_j, \overline{\mathsf{role}}), *)$ $\mathsf{complete}$.
  - In any other case, store $(\mathsf{sid}, \mathsf{error})$ and mark the records $((\mathsf{sid}, P_i, P_j, \mathsf{pw}_i, \mathsf{role}), *)$ and $((\mathsf{sid}, P_j, P_i, \mathsf{pw}_j, \overline{\mathsf{role}}), *)$ error.

  When the key is set, report the result (either error or $\mathsf{complete}$) to $\mathcal{S}$.
- **Key Delivery.** Upon receiving a message $(\mathsf{deliver}, \mathsf{b}, \mathsf{sid}, P_i)$ from $\mathcal{S}$, then if $P_i \in \mathcal{H}$ and there is a recorded tuple (sid, α) where $\alpha \in \{0, 1\}^k \cup \{\mathsf{error}\}$, send (sid, α) to P_i if b equals yes or $(\mathsf{sid}, \mathsf{error})$ if b equals no.
- **Player Corruption.** If $\mathcal{S}$ corrupts $P_i \in \mathcal{H}$ where there is a recorded tuple $((\mathsf{sid}, P_i, P_j, \mathsf{pw}_i, \mathsf{role}), *)$, then reveal pw_i to $\mathcal{S}$. If there also is a recorded tuple $(\mathsf{sid}, \mathsf{sk})$, that has not yet been sent to P_i, then send $(\mathsf{sid}, \mathsf{sk})$ to $\mathcal{S}$.

Fig. 2. Functionality $\mathcal{F}_{\mathsf{PAKE}}^{MA}$

scenario where each user P_i owns a pair of passwords $(\mathsf{pw}_i^L, \mathsf{pw}_i^R)$, each one shared with one of his neighbors, P_{i-1} and P_{i+1}, when players are organized around a ring. This is a quite general scenario since it covers the case of a unique common password: for each user, we set $\mathsf{pw}_i^L = \mathsf{pw}_i^R$. The ring structure is also general enough since a centralized case could be converted into a ring, where the center is duplicated between the users. Recall that thanks to the use of the split functionality, the GPAKE functionality invoked knows the group of the players, as well as the order among them. The following description is strongly based on that of [5].

Contributory Protocols. As in [5], we consider a stronger corruption model against insiders than the one proposed by Katz and Shin in [28]: in the latter model, one allows the adversary to choose the session key as soon as there is one corruption; as in the former case, in this paper we consider the notion of contributiveness, which guarantees the distribution of the session keys to be random as long as there are enough honest participants in the session: the adversary cannot bias the distribution unless it controls a large number of players. Namely, this notion formally defines the difference between a key distribution system and a key agreement protocol. More precisely, a protocol is said to be (t, n)-contributory if the group consists of n people and if the adversary cannot bias the key as long as it has corrupted (strictly) less than t players. The authors of [5] achieved $(n/2, n)$-contributiveness in an efficient way, and even $(n-1, n)$-contributiveness by running parallel executions of the protocol. We claim that our proposed protocol directly achieves (n, n)-contributiveness (or full-contributiveness), which means that the adversary cannot bias the key as long as there is at least one honest player in the group. Note that this definition remains very general: letting $t = 1$, we get back to the case in which $\mathcal{A}$ can set the key when it controls at least one player, as in [20].

Ideal Functionality for GPAKE with Mutual Authentication. We assume that every player owns two passwords $(\mathsf{pw}_i^L, \mathsf{pw}_i^R)$, and that for all i, $\mathsf{pw}_i^R = \mathsf{pw}_{i-1}^L$. Our functionality builds upon that presented in [5]. In particular, note that the functionality is not in charge of providing the passwords to the participants. Rather we let the environment do this. As already pointed out in [20], such an approach allows to model, for example, the case where some users may use the same password for different protocols and, more generally, the case where passwords are chosen according to some arbitrary distribution (*i.e.*, not necessarily the uniform one). Moreover, notice that allowing the environment to choose the passwords guarantees forward secrecy, basically for free. More generally, this approach allows to preserve security[1] even in those situations where the password is used (by the same environment) for other purposes.

Since we consider the (improved) split functionality model, the functionality is parameterized by an ordered group $\mathsf{Pid} = \{P_1, \ldots, P_n\}$, dynamically defined, consisting of all the players involved in the execution (be they real players or players controlled by the adversary). Thus, we note that it is unnecessary to impose that the players give this value Pid when notifying their interest to join an execution via a $\mathsf{NewSession}$ query, as was done in [5]. This additional simplification has some interest in practice, since the players do not always know the exact number of players involved, but rather a common characteristic (such as a Facebook group).

We thus denote by n the number of players involved (that is, the size of Pid) and assume that every player starts a new session of the protocol with input $(\mathsf{NewSession}, \mathsf{sid}, P_i, (\mathsf{pw}_i^L, \mathsf{pw}_i^R))$, where P_i is the identity of the player and $(\mathsf{pw}_i^L, \mathsf{pw}_i^R)$ its passwords. Once all the players in Pid, sharing the same sid, have sent their notification message, $\mathcal{F}_{\mathsf{GPAKE}}$ informs the adversary that it is ready to start a new session of the protocol.

In principle, after the initialization stage is over, all the players are ready to receive the session key. However the functionality waits for $\mathcal{S}$ to send an "ok" message before proceeding. This allows $\mathcal{S}$ to decide the exact moment when the key should be sent to the players and, in particular, it allows $\mathcal{S}$ to choose

[1] By "preserved" here we mean that the probability of breaking the scheme is basically the same as the probability of guessing the password.

the exact moment when corruptions should occur (for instance $\mathcal{S}$ may decide to corrupt some party P_i before the key is sent but after P_i decided to participate to a given session of the protocol, see [28]). One could imagine to get rid of this query and ask the functionality to generate the session key when the adversary asks the first delivery query, but it is easier to deal with the corruptions with the choice made here (which is the same as in [28]). Once the functionality receives a message $(\mathsf{sid}, \mathsf{ok}, \mathsf{sk})$ from $\mathcal{S}$, it proceeds to the key generation phase. This is done as in [5], except that, instead of checking whether the players all share the same passwords, $\mathcal{F}_{\mathsf{GPAKE}}$ checks whether the neighbors (the group is assumed to be ordered) share the same password. If all the players share the same passwords as their neighbors and less than t players are corrupted, $\mathcal{F}_{\mathsf{GPAKE}}$ chooses a key sk' uniformly and at random in the appropriate key space. If all the players share the same passwords as their neighbors but t or more players are corrupted, then the functionality allows $\mathcal{S}$ to fully determine the key by letting $\mathsf{sk}' = \mathsf{sk}$. In all the remaining cases no key is established.

This definition of the $\mathcal{F}_{\mathsf{GPAKE}}$ functionality deals with corruptions of players in a way quite similar to that of $\mathcal{F}_{\mathsf{GPAKE}}$ in [28], in the sense that if the adversary has corrupted some participants, it may determine the session key, but here only if there are enough corrupted players. Notice however that $\mathcal{S}$ is given such power only before the key is actually established. Once the key is set, corruptions allow the adversary to know the key but not to choose it.

In any case, after the key generation, the functionality informs the adversary about the result, meaning that the adversary is informed on whether a key was actually established or not. In particular, this means that the adversary is also informed on whether the players use compatible passwords or not: in practice, the adversary can learn whether the protocol succeeded or not by simply monitoring its execution (if the players follow the communication or stop it). Finally the key is sent to the players according to the schedule chosen by $\mathcal{S}$. This is formally modeled by means of key delivery queries. We assume that, as always in the UC framework, once $\mathcal{S}$ asks to deliver the key to a player, the key is sent immediately.

Notice that, the mutual authentication indeed means that if one of the players terminates with a session key (not an error), then all players share the key material; but, it doesn't mean that they all successfully terminated. Indeed, we cannot assume that all the flows are correctly forwarded by the adversary: it can modify just one flow, or at least omit to deliver one flow. This attack, called *denial of service*, is modeled in the functionality by the key delivery: the adversary can choose whether it wants the player to receive or not the good key/messages simply with the help of the keyword b set to yes or no.

4 Scheme

Intuition. The main idea of our protocol is to apply the Burmester-Desmedt technique [17] to any secure two-party PAKE achieving (mutual) explicit authentication in the UC framework. More precisely, the players execute such a protocol in flows $(2a)$ and $(2b)$ (see Figure 4) and use the obtained value in flows (3) and (4) as in a classical Burmester-Desmedt-based protocol.

The functionality $\mathcal{F}_{\mathsf{GPAKE}}$ is parameterized by a security parameter k, and the parameter t of the contributiveness. It interacts with an adversary $\mathcal{S}$ and an ordered set of parties $\mathsf{Pid} = \{P_1, \dots, P_n\}$ via the following queries:

- **Initialization.** Upon receiving $(\mathsf{NewSession}, \mathsf{sid}, P_i, (\mathsf{pw}_i^L, \mathsf{pw}_i^R))$ from player P_i for the first time, record $(\mathsf{sid}, P_i, (\mathsf{pw}_i^L, \mathsf{pw}_i^R))$, mark it fresh, and send (sid, P_i) to $\mathcal{S}$.
  If there are already $n-1$ recorded tuples $(\mathsf{sid}, P_j, (\mathsf{pw}_j^L, \mathsf{pw}_j^R))$ for players $P_j \in \mathsf{Pid} \setminus \{P_i\}$, then record $(\mathsf{sid}, \mathsf{ready})$ and send it to $\mathcal{S}$.
- **Key Generation.** Upon receiving a message $(\mathsf{sid}, \mathsf{ok}, \mathsf{sk})$ from $\mathcal{S}$ where there exists a recorded tuple $(\mathsf{sid}, \mathsf{ready})$, then, denote by n_c the number of corrupted players, and
  - If for all i, $\mathsf{pw}_i^R = \mathsf{pw}_{i+1}^L$ and $n_c < t$, choose $\mathsf{sk}' \in \{0,1\}^k$ uniformly at random and store $(\mathsf{sid}, \mathsf{sk}')$. Next, for all $P_i \in \mathsf{Pid}$ mark the record $(\mathsf{sid}, P_i, (\mathsf{pw}_i^L, \mathsf{pw}_i^R))$ $\mathsf{complete}$.
  - If for all i, $\mathsf{pw}_i^R = \mathsf{pw}_{i+1}^L$ and $n_c \geq t$, store $(\mathsf{sid}, \mathsf{sk})$. Next, for all $P_i \in \mathsf{Pid}$ mark $(\mathsf{sid}, P_i, (\mathsf{pw}_i^L, \mathsf{pw}_i^R))$ $\mathsf{complete}$.
  - In any other case, store $(\mathsf{sid}, \mathsf{error})$. For all $P_i \in \mathsf{Pid}$ mark the record $(\mathsf{sid}, P_i, (\mathsf{pw}_i^L, \mathsf{pw}_i^R))$ error.

  When the key is set, report the result (either error or $\mathsf{complete}$) to $\mathcal{S}$.
- **Key Delivery.** Upon receiving a message $(\mathsf{deliver}, \mathsf{b}, \mathsf{sid}, P_i)$ from $\mathcal{S}$, then if $P_i \in \mathsf{Pid}$ and there is a recorded tuple (sid, α) where $\alpha \in \{0,1\}^k \cup \{\mathsf{error}\}$, send (sid, α) to P_i if b equals yes or $(\mathsf{sid}, \mathsf{error})$ if b equals no.
- **Player Corruption.** If $\mathcal{S}$ corrupts $P_i \in \mathsf{Pid}$ where there is a recorded tuple $(\mathsf{sid}, P_i, (\mathsf{pw}_i^L, \mathsf{pw}_i^R))$, then reveal $(\mathsf{pw}_i^L, \mathsf{pw}_i^R)$ to $\mathcal{S}$. If there also is a recorded tuple $(\mathsf{sid}, \mathsf{sk})$, that has not yet been sent to P_i, then send $(\mathsf{sid}, \mathsf{sk})$ to $\mathcal{S}$.

Fig. 3. Functionality $\mathcal{F}_{\mathsf{GPAKE}}$

The split functionality is emulated thanks to the first flow, where the players engage in their signature verification key, as well as the elements used for the splitting part of the two-party protocols. They are then (after the dotted line in the figure) partitioned according to the values they received during this first round.

Finally, the contributiveness is ensured by the following trick: In addition to establishing pairwise keys between any two pair of neighbors, the players also choose on their own a random secret value K_i, which will also be used in the session key generation. An important point is that these values are chosen independently thanks to the commitment between flows $(2a)$ and $(2b)$. This will ensure the session key to be uniformly distributed as long as at least one player is honest.

Building Blocks. We assume to be given a universally composable two-party password-based authenticated key exchange with mutual authentication 2PAKE, achieving or not security against adaptive corruptions. This key exchange is assumed (as defined by the ideal functionality) to give as output a uniformly distributed random string. Due to the mutual authentication, this protocol results in an error message in case it does not succeed: Either the two players end with the same key, or they end with an error. Note, however, that one player can have

(1) $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{SKG}$
$s_i^L = \mathsf{split2PAKE}(\mathsf{ssid}; P_{i-1}, \mathsf{pw}_{i-1}^R; P_i, \mathsf{pw}_i^L)$
$s_i^R = \mathsf{split2PAKE}(\mathsf{ssid}; P_{i-1}, \mathsf{pw}_{i-1}^R; P_i, \mathsf{pw}_i^L)$ $\xrightarrow{(\mathrm{VK}_i, s_i^L, s_i^R)}$

. . . .

After this point, the session identifier becomes
$\mathsf{ssid}' = \mathsf{ssid} \| \mathrm{VK}_1 \| s_1^L \| s_1^R \| \ldots \| \mathrm{VK}_n \| s_n^L \| s_n^R$.

$(2a)$ executes $\mathsf{2PAKE}(\mathsf{ssid}'; P_{i-1}, \mathsf{pw}_{i-1}^R; P_i, \mathsf{pw}_i^L)$,
obtaining $K_i^L = K_{i-1}^R$ shared with P_{i-1} $\xrightarrow{\ldots}$
executes $\mathsf{2PAKE}(\mathsf{ssid}'; P_i, \mathsf{pw}_i^R; P_{i+1}, \mathsf{pw}_{i+1}^L)$,
obtaining $K_i^R = K_{i+1}^L$ shared with P_{i+1} $\xrightarrow{\ldots}$
chooses at random $K_i \overset{\$}{\leftarrow} \{0,1\}^k$
computes $X_i^L = K_i^L \oplus K_i$ and $X_i^R = K_i \oplus K_i^R$
computes and sends $c_i = \mathsf{com}(\mathsf{ssid}', i, X_i^L, X_i^R)$ $\xrightarrow{c_i}$

$(2b)$ opens X_i^L, X_i^R $\xrightarrow{X_i^L, X_i^R}$

(3) checks $c_j = \mathsf{com}(\mathsf{ssid}', j, X_j^L, X_j^R) \quad \forall j \neq i$ and $X_1^L \oplus X_1^R \oplus \cdots \oplus X_n^L \oplus X_n^R = 0$
and aborts if one of these values is incorrect
computes $K_{j+1}^L = X_j^R \oplus K_j$, $K_{j+1} = X_{j+1}^L \oplus K_{j+1}^L$ $\forall j = i, \ldots, n+i-1 \pmod n$
computes $\mathsf{sk}_0 \| \mathsf{sk}_1 = f(K_1, \ldots, K_n)$, $\mathsf{Auth}_i = \mathsf{Mac}(\mathsf{sk}_1; \mathsf{ssid}', i, \{X_j^L, X_j^R\}_j)$,
and $\sigma_i = \mathsf{Sign}(\mathrm{SK}_i; \mathsf{ssid}', i, \mathsf{Auth}_i, \{X_j^L, X_j^R\}_j)$ $\xrightarrow{\mathsf{Auth}_i, \sigma_i}$

(4) checks $\mathsf{Ver}(\mathsf{sk}_1; \mathsf{ssid}', j, \{X_k^L, X_k^R\}_k; \mathsf{Auth}_j)$
and $\mathsf{Verify}(\mathrm{VK}_j; \mathsf{ssid}', \mathsf{Auth}_j, \{X_k^L, X_k^R\}_k; \sigma_j) \quad \forall j \neq i$
If they are correct, then marks the session as $\mathsf{complete}$ and sets $\mathsf{sk}_i = \mathsf{sk}_0$.
Otherwise, sets $\mathsf{sk}_i = \mathsf{error}$.

Fig. 4. Description of the protocol for player P_i, with index i and passwords pw_i^L and pw_i^R

a key while the other is still waiting since the adversary can retain a message: This is a denial-of-service attack, since a timeout will stop the execution of the protocol. Mutual authentication guarantees that the players cannot end with two different keys.

Let $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{Verify})$ be a one-time signature scheme, SKG being the signature key generation, Sign the signing algorithm and Verify the verifying algorithm. Note that we do not require a *strong* one-time signature: Here, the adversary is allowed to query the signing oracle at most once, and should not be able to forge a signature on a *new* message.

Let $(\mathsf{Mac}, \mathsf{Ver})$ be a message authentication code scheme, Mac being the authenticating algorithm and Ver the verifying algorithm. A pseudo-random function could be used, since this is a secure MAC [11].

As usual, we will need a randomness extractor, in order to generate the final session key, as well as an authentication key (for the key confirmation round, guaranteed by a Mac computation). But because of the UC framework, and the definition of the functionality, in the case of a corrupted player, the adversary will learn all the inputs of the extractor, chosen by the players, and the session key chosen by the functionality as well. We will thus have to be able to choose the inputs for the honest players so that they lead to the expected output. We thus use a specific randomness extractor, with a kind of partial invertibility: we consider a finite field $\mathbb{F} = \mathbb{F}_q$. The function

$$\begin{aligned} f : (\mathbb{F}^* \times \ldots \times \mathbb{F}^*) \times (\mathbb{F} \times \ldots \times \mathbb{F}) &\to \mathbb{F} \\ (\alpha_1, \ldots, \alpha_n \quad ; \quad h_1, \ldots, h_n) &\mapsto \sum \alpha_i h_i \end{aligned}$$

is a randomness extractor from tuples $(h_1, \ldots, h_n) \in \mathbb{F}^n$ where at least one h_i is uniformly distributed and independent of the others. Since it can be shown as a universal hash function, using similar techniques to [22], if we consider any distribution $\mathcal{D}_i$ on $\mathbb{F}^n$, for which the distribution $\{h_i | (h_1, \ldots, h_n) \leftarrow \mathcal{D}_i\}$ is the uniform distribution in $\mathbb{F}$, then the distributions

$$\begin{aligned} &(\alpha_1, \ldots, \alpha_n, f(\alpha_1, \ldots, \alpha_n; h_1, \ldots, h_n)), && (\alpha_1, \ldots, \alpha_n) \overset{\$}{\leftarrow} \mathbb{F}^{*n}, (h_1, \ldots, h_n) \leftarrow \mathcal{D}_i \\ &(\alpha_1, \ldots, \alpha_n, U), && (\alpha_1, \ldots, \alpha_n) \overset{\$}{\leftarrow} \mathbb{F}^{*n}, U \overset{\$}{\leftarrow} \mathbb{F} \end{aligned}$$

are perfectly indistinguishable. The tuple $(\alpha_1, \ldots, \alpha_n)$ is the public key of the randomness extractor, and it is well-known that it can be fixed in the CRS [29], with a loss of security linear in the number of queries. Since n might not be fixed in advance, we can use a pseudo-random generator that generates the sequence $\alpha_1, \ldots,$ from a key k in the CRS. Anyway, we generically use f as the variable input-length randomness extractor in the following. As said above, we will have to invert f to adapt the input of an honest user to the expected session key: for a fixed key, some fixed inputs $I_i = (h_1, \ldots, \hat{h_i} \ldots, h_n) \in \mathbb{F}^{n-1}$ (possibly all but one, here h_i), and the output U, the function $g_i(I_i, U)$ completes the input so that the output by f is U. With our function f, we have $g_i(I_i, U) = (U - \sum_{j \neq i} \alpha_j h_j)/\alpha_i$.

Finally, we will also need a commitment scheme. In addition to being hiding and binding, we will require it to be extractable, equivocable and non-malleable, such as those of [19,1,7]. Even if this latter commitment is only *conditionally* extractable, this will not matter here since the commitment will be opened later: The user cannot try to cheat otherwise the protocol stops. Note that the extractable property allows the simulator to obtain the values committed to by the adversary, the equivocable property allows him to open his values to something consistent with them, and the non-malleable property ensures that when $\mathcal{A}$ sees a commitment, he is not able to construct another one with a related distribution. Because of extractability and equivocability, both the hiding and the binding properties are computational only.

Description of the Protocol. For the sake of completeness, we describe the case where each player owns two different passwords (pw_i^L and pw_i^R), and each pair of neighbors (while the ring is set) shares a common password ($\mathsf{pw}_i^R = \mathsf{pw}_{i+1}^L$). The case where the players all share the same password is easily derived from here, by letting $\mathsf{pw}_i^L = \mathsf{pw}_i^R$. Note that both cases will UC-emulate the GPAKE functionality presented earlier.

We do not assume that the members of the actual group are known in advance. Then one has to imagine a system of timeouts after which the participants consider that no one else will notify its interest in participating to the protocol, and continue the execution. Once the players are known, we order them using a public pre-determined technique (*e.g.*, the alphabetical order on the first flow). Then, for the sake of simplicity we rename the players actually participating $P_1, \ldots, P_n$ according to this order.

Furthermore, such timeouts will also be useful in Flow $(2a)$ in case a player has aborted earlier, in order to avoid other players to wait for it indefinitely. After a certain amount of time has elapsed, the participants should consider that the protocol has failed and abort. Such a synchronization step is useful for the contributiveness, see later on.

Informally, and omitting the details, the algorithm (see Figure 4) can be described as follows: First, each player applies SKG to generate a pair $(\mathrm{SK}_i, \mathrm{VK}_i)$ of signature keys, and sends the value VK_i. They also engage in two two-party key exchange protocols with each of their neighbors: We denote $\mathsf{split2PAKE}$ the corresponding first flow of this protocol, used for the split functionality. The players will be split after this round according to the values received. At this point, the session identifier becomes $\mathsf{ssid}' = \mathsf{ssid} \| \mathrm{VK}_1 \| s_1^L \| s_1^R \| \ldots \| \mathrm{VK}_n \| s_n^L \| s_n^R$ (more details follow). We stress that the round $(2a)$ does not begin until all commitments have been received. In this round, the players open to the values committed.

In round $(2a)$, the players check the commitments received (and abort if one of them is incorrect). Next, player P_i chooses at random a bitstring K_i. It also gets involved into two $\mathsf{2PAKE}$ protocols, with each of its neighbors P_{i-1} and P_{i+1}, and the passwords pw_i^L and pw_i^R, respectively. This creates two random strings: $K_i^L = \mathsf{2PAKE}(\mathsf{ssid}'; P_{i-1}, \mathsf{pw}_{i-1}^R; P_i, \mathsf{pw}_i^L)$, shared with P_{i-1}, and $K_i^R = \mathsf{2PAKE}(\mathsf{ssid}'; P_i, \mathsf{pw}_i^R; P_{i+1}, \mathsf{pw}_{i+1}^L)$, shared with P_{i+1}. It finally computes $X_i^L = K_i^L \oplus K_i$ and $X_i^R = K_i \oplus K_i^R$ and commits to these values. Pictorially, the situation can be summarized as follows:

$$\begin{array}{ccccccc}
P_{i-1}(\mathsf{pw}_{i-1}^R) & & P_i(\mathsf{pw}_i^L) & & P_i(\mathsf{pw}_i^R) & & P_{i+1}(\mathsf{pw}_{i+1}^L) \\
& K_{i-1}^R = K_i^L & & K_i \xleftarrow{\$} \{0,1\}^k & & K_i^R = K_{i+1}^L & \\
X_{i-1}^R & & X_i^L = K_i^L \oplus K_i & & X_i^R = K_i \oplus K_i^R & & X_{i+1}^L
\end{array}$$

where $X_{i-1}^R = K_{i-1} \oplus K_{i-1}^R = K_{i-1} \oplus K_i^L$ and $X_{i+1}^L = K_{i+1}^L \oplus K_{i+1} = K_i^R \oplus K_{i+1}$.

Once P_i has received all these commitments (again, we stress that no player begins this round before having received all the commitments previously sent), it opens to the values committed (round $(2b)$).

In round (3), the players check the commitments received (and abort if one of them is incorrect). Next, player P_i iteratively computes all the K_j's required to compute the session keys $\mathsf{sk}_0 \| \mathsf{sk}_1$ and the key confirmation $\mathsf{Auth}_i = \mathsf{Mac}(\mathsf{sk}_1; \mathsf{ssid}', i, \{X_j^L, X_j^R\}_j)$. It also signs this authenticator along with all the commitments received in the previous flow.

Finally, in round (4), after having checked the authenticators and the signatures, the players mark their session as $\mathsf{complete}$ (or abort if one of these values is incorrect) and set their session key $\mathsf{sk}_i = \mathsf{sk}_0$.

Remarks. As soon as a value received by P_i doesn't match with the expected value, it aborts, setting the key $\mathsf{sk}_i = \mathsf{error}$. In particular, every player checks the commitments $c_j = \mathsf{com}(\mathsf{ssid}', j, X_j^L, X_j^R)$, the signatures $\sigma_j = \mathsf{Sign}(\mathrm{SK}_j; \mathsf{ssid}', \mathsf{Auth}_j, \{X_k^L, X_k^R\}_k)$, and finally the key confirmations $\mathsf{Auth}_j = \mathsf{Mac}(\mathsf{sk}_1; \mathsf{ssid}', j, \{X_k^L, X_k^R\}_k)$. This enables the protocol to achieve mutual authentication.

The protocol also realizes the split functionality due to the two following facts: First, the players are partitioned according to the values VK_j and $\mathsf{split2PAKE}$ they received during the first round (*i.e.*, before the dotted line in Figure 4). All the VK_i are shared among them and their session identifier becomes $\mathsf{ssid}' = \mathsf{ssid} \| \mathrm{VK}_1 \| s_1^L \| s_1^R \| \ldots \| \mathrm{VK}_n \| s_n^L \| s_n^R$. Furthermore, in round 3, the signature added to the authentication flow prevents the adversary from being able to change an X_i^L or X_i^R to another value. Since the session identifier ssid' is included in all the commitments, and in the latter signature, only players in the same subset can accept and conclude with a common key.

Then, the contributory property is ensured by the following trick: At the beginning of each flow, the players wait until they have received all the other values of the previous flow before sending their new one. This is particularly important between ($2a$) and ($2b$). Thanks to the commitments sent in this flow, it is impossible for a player to compute its values X_i^L and X_i^R once it has seen the others: Every player has to commit its values at the same time as the others, and cannot make them depend on the other values sent by the players (recall that the commitment is non-malleable). This disables it from being able to bias the key (more details can be found in the proof, see the full version [6]).

Finally we point out that, in our proof of security, we don't need to assume that the players erase any ephemeral value before the end of the computation of the session key.

Our Main Theorem. Let $\widehat{s\mathcal{F}}_{GPAKE}$ be the multi-session extension of the split functionality $s\mathcal{F}_{\mathsf{GPAKE}}$.

Theorem 1. *Assuming that the protocol* $\mathsf{2PAKE}$ *is a universally composable two-party password-based authenticated key exchange with mutual authentication secure against adaptive (*resp. *static) corruptions,* $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{Verify})$ *a one-time signature scheme,* com *a non-malleable, extractable and equivocable commitment scheme,* $(\mathsf{Mac}, \mathsf{Ver})$ *a message authentication code scheme, and f a randomness extractor as defined earlier, the protocol presented in Figure 4 securely realizes $\widehat{s\mathcal{F}}_{GPAKE}$ in the CRS model, in the presence of adaptive (*resp. *static) adversaries, and is fully-contributory.*

5 Merging Two Groups

Since the case in which a single user joins an existing group is a particular case of merging two groups, we concentrate on the latter more general case. Let $G = \{P_1, \ldots, P_n\}$ and $G' = \{P'_1, \ldots, P'_m\}$ be two groups which have already created two group session keys via the protocol described in Section 4. Using the same notations, we assume that each player P_k in G has kept in memory its own private value K_k as well as all the public values $\{X_1^L, X_1^R, \ldots, X_n^L, X_n^R\}$. Similarly, assume that each player P'_ℓ in G' has kept in memory its own private value K'_ℓ as well as all the public values $\{X_1'^L, X_1'^R, \ldots, X_m'^L, X_m'^R\}$.

In other words, we ask each player to keep in memory all the values necessary to the computation of the group's session key. Remarkably, note that they only have to keep a single private value, and that all the other values are public, and can be kept publicly in a single place accessible to the players.

The goal of our dynamic merge protocol is to allow the computation of a joint group session key between G and G', without asking the whole new group $G \cup G'$ to start a key-exchange protocol from scratch. In addition, the protocol we describe here has two nice properties: First, it does not increase the memory requirements of each player. Second, it is done in such a way that the situation of each player after the merge protocol is the same as its situation before it. That way, future join or merge protocols can easily take place iteratively without any change.

For sake of simplicity, we first describe a basic version of our protocol, in which only one representative of each group participates in the new exchange of messages between the two groups. Clearly, this version is not fully contributory since only two participants take place in the protocol. We then show how to extend it into a fully contributory protocol, in which all $n+m$ participants will take part in the message exchange.

Basic Version. Let P_i and P'_j denote the particular members of G and G' that are acting as the representative of these groups. Only these two participants will take part in the merge protocol. In order to construct a session key for the new group, these two players are assumed to share a common password, denoted as pw for P_i and pw' for P'_j. The situation is summarized in Figure 5, where the upper part (1) represents the former second group, with the values computed during the execution of the GPAKE protocol, and the lower part (2) represents the former first group, with the values computed during the execution of the GPAKE protocol. The hatched lines represent the abandoned "links". Indeed, both P_i and P'_j will erase their values K_i and K'_j and create two new connections between them, thus creating the new group

$$G'' = \{P_1, \ldots, P_{i-1}, P_i, P'_j, P'_{j+1}, \ldots, P'_m, P'_1, \ldots, P'_{j-1}, P'_j, P_i, P_{i+1}, \ldots, P_n\}$$

These connections are represented vertically in the middle part (2) of the figure. We stress that during the merge protocol, no value is computed in parts (1) and (2). The core of the protocol is part (3).

For lack of space, we refer the interested reader to the full version [6] for the precise description of the protocol. Informally, the merge protocol consists in the execution of a simplified GPAKE protocol with the whole group G", but the interesting point is that only P_i and P'_j participate and exchange messages, executing two 2PAKE protocols, instead of the $n+m-1$ that would be necessary for an execution from scratch with this new group. Merging two groups is thus much more efficient. The two executions are performed once for the left part of (3) in Figure 5, and once for the right part. For P_i and P'_j, the steps are similar to those of a normal GPAKE protocol execution. Additionnaly, P_i and P'_j have to broadcast the necessary (old) values X_k^L, X_k^R and $X_l'^L, X_l'^R$ to the

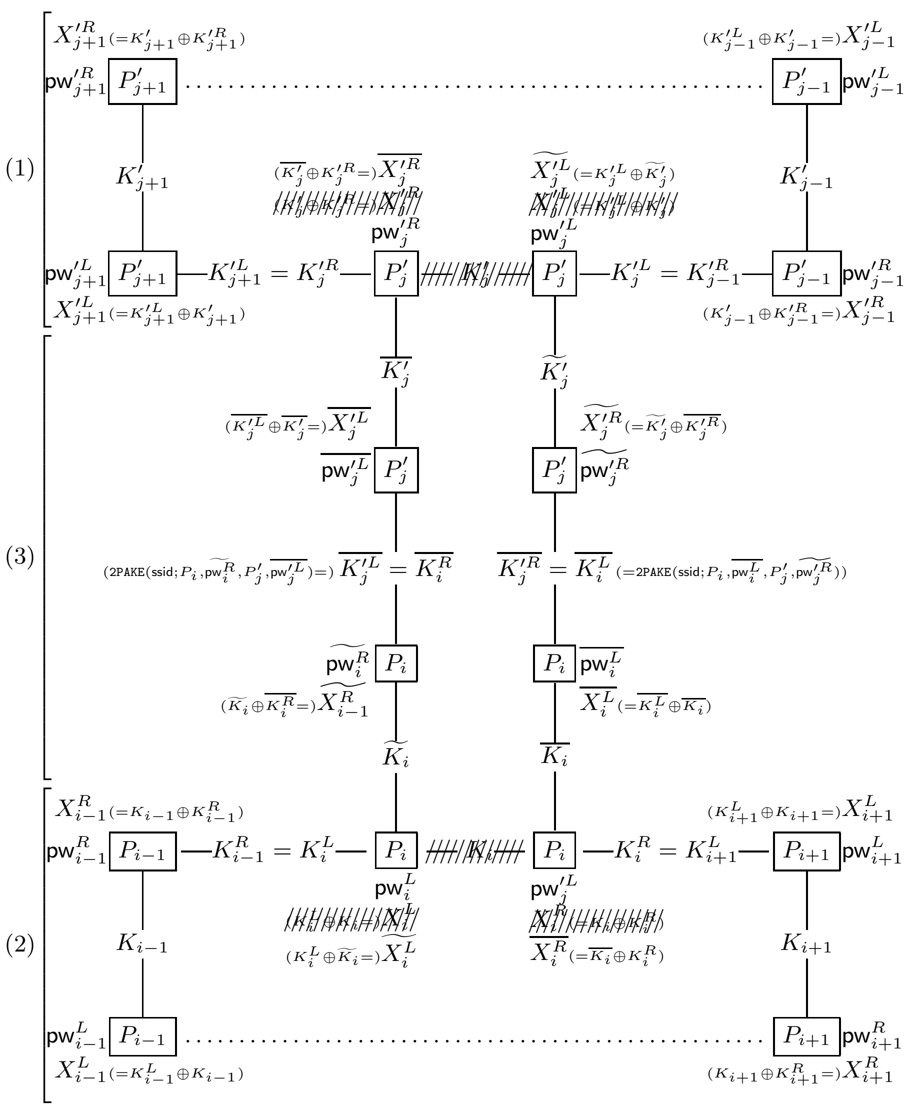

(1)
(2)
(3)
$X'^R_{j+1}(=K'_{j+1}\oplus K'^R_{j+1})$
$(K'^L_{j-1}\oplus K'_{j-1}=)X'^L_{j-1}$
pw'^R_{j+1} P'_{j+1}
P'_{j-1} pw'^L_{j-1}
K'_{j+1}
K'_{j-1}
pw'^L_{j+1} P'_{j+1} — $K'^L_{j+1} = K'^R_j$ — P'_j P'_j — $K'^L_j = K'^R_{j-1}$ — P'_{j-1} pw'^R_{j-1}
$X'^L_{j+1}(=K'^L_{j+1}\oplus K'_{j+1})$
$(K'_{j-1}\oplus K'^R_{j-1}=)X'^R_{j-1}$
pw'^R_j
pw'^L_j
$(2\mathsf{PAKE}(\mathsf{ssid};P_i,\widetilde{\mathsf{pw}^R_i},P'_j,\overline{\mathsf{pw}'^L_j})=)\ \overline{K'^L_j} = \overline{K^R_i}$
$\overline{K'^R_j} = \overline{K^L_i}\ (=2\mathsf{PAKE}(\mathsf{ssid};P_i,\overline{\mathsf{pw}^L_i},P'_j,\widetilde{\mathsf{pw}'^R_j}))$
$X^R_{i-1}(=K_{i-1}\oplus K^R_{i-1})$
$(K^L_{i+1}\oplus K_{i+1}=)X^L_{i+1}$
pw^R_{i-1} P_{i-1} — $K^R_{i-1} = K^L_i$ — P_i P_i — $K^R_i = K^L_{i+1}$ — P_{i+1} pw^L_{i+1}
K_{i-1}
K_{i+1}
pw^L_{i-1} P_{i-1}
P_{i+1} pw^R_{i+1}
$X^L_{i-1}(=K^L_{i-1}\oplus K_{i-1})$
$(K_{i+1}\oplus K^R_{i+1}=)X^R_{i+1}$

Fig. 5. Merging two Groups: (1) represents the former group $(P'_1, P'_2, \ldots, P'_m)$; (2) represents the former group $(P_1, P_2, \ldots, P_n)$; (3) is the proper merge protocol, between the inviter P_i and the invited P'_j

other members of each subgroup, to enable them derive the new key. These other players only participate passively, listening to broadcasts so as to learn the values needed to compute the new key of the merged group.

This merge protocol is thus only partially contributory since P_i and P'_j are the only players participating and exchanging messages. Furthermore, it is not forward-secure since the players of both groups become able to compute the former key of the other group thanks to the values broadcasted by P_i and P'_j. Also note that we could simplify this protocol by merging the commitments, signatures and MACs, doing only one for each player. But we chose to keep the protocol symmetric, the values $\tilde{x}$ representing roughly the unnecessary values (of the vanishing players, see the next paragraph) and the values $\overline{x}$ representing roughly the needed values.

We claim that after this execution, the players will find themselves in a similar situation than after a normal GPAKE protocol. For the moment, this is not the case since P_i and P'_j appear twice in the ring (see Figure 5). For both of them, we have to get rid of one instance of the player. To this aim, once this protocol is executed, P_i "vanishes" on the left part of (3) in Figure 5, letting the player P_{i-1} with a new value X^R_{i-1} equal to $X^R_{i-1} \oplus \widetilde{X^L_i}$ and the player P'_j with a new value $\overline{X'^L_j}$ equal to $\overline{X'^L_j} \oplus \widetilde{X^R_i}$. The new 2PAKE-value shared between them is $\widetilde{K_i}$. The same thing happens on the right part of (3) in Figure 5: P'_j vanishes, letting the player P'_{j-1} with the new value X'^R_{j-1} equal to $X'^R_{j-1} \oplus \widetilde{X'^L_j}$ and P_i with the new value $\overline{X^L_i}$ equal to $\widetilde{X'^R_j} \oplus \overline{X^L_i}$. The new 2PAKE-value shared between them is $\widetilde{K'_j}$. This way, it is as if the players P_i and P'_j had only participated once in the new protocol: P_i between P'_{j-1} and P_{i+1}, and P'_j between P_{i-1} and P'_{j+1}. Finally, we will only need to keep the following values: $\overline{K'_j}$ secretly for P'_j, $\overline{K_i}$ secretly for P_i, and $X^R_{i-1} = X^R_{i-1} \oplus \widetilde{X^L_i}$, $\overline{X'^L_j} = \overline{X'^L_j} \oplus \widetilde{X^R_i}$, $X'^R_{j-1} = X'^R_{j-1} \oplus \widetilde{X'^L_j}$ and $\overline{X^L_i} = \widetilde{X'^R_j} \oplus \overline{X^L_i}$ publicly. The values of the rest of the group remain unchanged. This will allow to do another join of merge iteratively.

Pictorially, this leads to the new following situation. First, the left part of (3) in Figure 5 without P_i:

$$\begin{array}{ccccccc}
P_{i-1}(\mathsf{pw}^R_{i-1}) & & P'_j(\overline{\mathsf{pw}'^L_j}) & & P'_j(\mathsf{pw}'^R_j) & & P'_{j+1}(\mathsf{pw}'^L_{j+1}) \\
 & \widetilde{K_i} & & & \overline{K'_j} & K'^R_j = K'^L_{j+1} & \\
X^R_{i-1} \oplus \widetilde{X^L_i} = K_{i-1} \oplus \widetilde{K_i} & & \overline{X'^L_j} \oplus \widetilde{X^R_i} = \widetilde{K_i} \oplus K'_j & & \overline{X'^R_j} & & X'^L_{j+1}
\end{array}$$

with $\widetilde{K_i}, \overline{K'_j} \overset{\$}{\leftarrow} \{0,1\}^k$, $\overline{X'^R_j} = \overline{K'_j} \oplus K'^R_j$ and $X'^L_{j+1} = K'^L_{j+1} \oplus K'_{j+1}$. Then, the right part of (3) in Figure 5 without P'_j (with $\widetilde{K'_j}, \overline{K_i} \overset{\$}{\leftarrow} \{0,1\}^k$, $\overline{X^R_i} = \overline{K_i} \oplus K^R_i$ and $X^L_{i+1} = K^L_{i+1} \oplus K_{i+1}$):

$$\begin{array}{ccccccc}
P'_{j-1}(\mathsf{pw}'^R_{j-1}) & & P_i(\overline{\mathsf{pw}^L_i}) & & P_i(\mathsf{pw}^R_i) & & P_{i+1}(\mathsf{pw}^L_{i+1}) \\
 & \widetilde{K'_j} & & & \overline{K_i} & K^R_i = K^L_{i+1} & \\
X'^R_{j-1} \oplus \widetilde{X'^R_j} = K'_{j-1} \oplus \widetilde{K'_j} & & \overline{X^L_i} \oplus \widetilde{X'^R_j} = \overline{K_i} \oplus \widetilde{K'_j} & & \overline{X^R_i} & & X^L_{i+1}
\end{array}$$

Again, all the other values of the rest of the group remain unchanged.

Forward-Secure Fully-Contributory Protocol. The scheme presented in the previous section does not provide forward secrecy since the players in one group learn enough information to compute the previous key of the other group. It is also not fully contributory because P_i and P'_j are the only players to actively participate in the merge protocol: they have full control over the value of the new group session key. In order to achieve these goals, we make two significant changes to the above protocol. These changes, presented in the full version [6] for lack of space, are two-fold: First, to obtain the contributiveness, we impose to each player of both groups to participate in the later phases of the protocol, issuing a new fresh value K_k or K'_ℓ; Second, in order to achieve forward secrecy, we change the way in which we compute the local key values (all the K's used by a user) by using the initial ones as the seed or state of a forward-secure stateful pseudorandom generator [13] and then use this state to generate the actual K's values, as well as the next state.

6 Implementation Considerations

The protocols that have been described above were designed for their security properties, and for the quality of the proof of security. When it comes to practical implementations, some additional considerations have to be made.

Definition of the Group. We will consider a use case where the participants to the GPAKE are already members of a chat room, which is the communication means used to broadcast messages. The protocol has to be resistant to the fact that some members of the chat room are idle and will not participate to the GPAKE, and also that some members of the chat room might have difficulties to participate because of connectivity issues: this is thus a nice property the functionality (granted the split functionality) does not need to know the list of participants in advance.

Therefore, instead of ending the initialization phase when a number n of participants is reached (as in previous protocols), we end the initialization phase at the initiative of any of the participants or a timeout. From a practical point of view, it means that in the algorithm of Figure 4, going to step (2a) does not need that all commitments are received, on the opposite, these commitments will be used to dynamically define the group after a certain time, possibly defined by a timeout: the members of the chat room that have sent their commitments.

Another practical issue is the ordering on the ring, which defines the neighbors of each participant. Since the group is not known in advance, this ordering will be defined from the commitments sent in (1): *e.g.*, the alphabetical order.

Authentication within the Group. As explained in the description of the protocol, is accepted as a member of the group anyone that shares a password with another member of the group. This is the best authentication that can be achieved for a GPAKE because a unique shared key is generated for the group. But after the protocol execution, each user owns a pair (SK_i, VK_i) of signing/verification key. It can be used by each participant to sign his/her own messages, to avoid that one participant impersonates another. But then, a (multi-time) signature scheme has to be used, with some formatting constraint to avoid collisions between the use for the GPAKE protocol and the signature of a message.

Removal of one Participant. This protocol provides the functionality of adding members to the group in the full version [6], but does not provide the functionality of removing members. Indeed, while there is a possibility of telling two participants apart (cf. previous paragraph) there is no possibility of truly authenticating a participant. Only the alias (the signing keys) is known.

A functionality that could be implemented is the ban of a participant identified by his/her alias, *e.g.*, because this participant has sent inappropriate messages. However, because all the random K_i are known at step (3), it is necessary to generate new random values that are not known by the banned participant. Therefore, the recommended way to remove one participant from a group is to start again the GPAKE protocol with shared passwords that are not known by this participant.

Acknowledgments

This work was supported in part by the French ANR-07-SESU-008 PAMPA Project.

References

1. Abdalla, M., Bohli, J.-M., González Vasco, M.I., Steinwandt, R.: (Password) authenticated key establishment: From 2-party to group. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 499–514. Springer, Heidelberg (2007)
2. Abdalla, M., Boyen, X., Chevalier, C., Pointcheval, D.: Distributed public-key cryptography from weak secrets. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 139–159. Springer, Heidelberg (2009)
3. Abdalla, M., Boyen, X., Chevalier, C., Pointcheval, D.: Strong cryptography from weak secrets: Building efficient pke and ibe from distributed passwords in bilinear groups. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 297–315. Springer, Heidelberg (2010)
4. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Efficient two-party password-based key exchange protocols in the UC framework. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 335–351. Springer, Heidelberg (2008)
5. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Password-authenticated group key agreement with adaptive security and contributiveness. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 254–271. Springer, Heidelberg (2009)
6. Abdalla, M., Chevalier, C., Granboulan, L., Pointcheval, D.: Contributory Password-Authenticated Group Key Exchange with Join Capability. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 142–160. Springer, Heidelberg (2011); Full version available from the web page of the authors
7. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer, Heidelberg (2009)
8. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
9. Abdalla, M., Pointcheval, D.: A scalable password-based group key exchange protocol in the standard model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer, Heidelberg (2006)
10. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)

11. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. Journal of Computer and System Sciences 61(3), 362–399 (2000)
12. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
13. Bellare, M., Yee, B.S.: Forward-security in private-key cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (2003)
14. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy, May 1992, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
15. Bohli, J.-M., Vasco, M.I.G., Steinwandt, R.: Password-authenticated constant-round group key establishment with a common reference string. Cryptology ePrint Archive, Report 2006/214 (2006)
16. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic group Diffie-Hellman key exchange under standard assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
17. Burmester, M., Desmedt, Y.: A secure and scalable group key exchange system. Information Processing Letters 94(3), 137–143 (2005)
18. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, October 2001, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)
19. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
20. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
21. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
22. Carter, L., Wegman, M.N.: Universal classes of hash functions. J. Comput. Syst. Sci. 18(2), 143–154 (1979)
23. 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)
24. Gentry, C., MacKenzie, P., Ramzan, Z.: A method for making password-based key exchange resilient to server compromise. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 142–159. Springer, Heidelberg (2006)
25. Choudary Gorantla, M., Boyd, C., Nieto, J.M.G.: Universally composable contributory group key exchange. In: ASIACCS 2009, March 2009, pp. 146–156. ACM Press, New York (2009)
26. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: ACM CCS 2010. Springer, Heidelberg (2010)
27. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
28. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: ACM CCS 2005, November 2005, pp. 180–189. ACM Press, New York (2005)
29. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, Cambridge (2005)

Ideal Key Derivation and Encryption in Simulation-Based Security[⋆]

Ralf Küsters and Max Tuengerthal

University of Trier, Germany
{kuesters,tuengerthal}@uni-trier.de

Abstract. Many real-world protocols, such as SSL/TLS, SSH, IPsec, DNSSEC, IEEE 802.11i, and Kerberos, derive new keys from other keys. To be able to analyze such protocols in a composable way, in this paper we extend an ideal functionality for symmetric and public-key encryption proposed in previous work by a mechanism for key derivation. We also equip this functionality with message authentication codes (MACs), digital signatures, and ideal nonce generation. We show that the resulting ideal functionality can be realized based on standard cryptographic assumptions and constructions, hence, providing a solid foundation for faithful, composable cryptographic analysis of real-world security protocols.

Based on this new functionality, we identify sufficient criteria for protocols to provide universally composable key exchange and secure channels. Since these criteria are based on the new ideal functionality, checking the criteria requires merely information-theoretic or even only syntactical arguments, rather than involved reduction arguments.

As a case study, we use our method to analyze two central protocols of the IEEE 802.11i standard, namely the 4-Way Handshake Protocol and the CCM Protocol, proving composable security properties. As to the best of our knowledge, this constitutes the first rigorous cryptographic analysis of these protocols.

Keywords: security protocols, compositional analysis, simulation-based security.

1 Introduction

Security protocols employed in practice, such as SSL/TLS, SSH, IPsec, DNSSEC, IEEE 802.11i, and Kerberos, are very complex, and hence, hard to analyze. In order to tame this complexity, a viable approach is composable security analysis based on the framework of simulation-based security, in particular universal composability/reactive simulatability [8,30]: Higher-level components of a protocol are designed and analyzed based on lower-level idealized components, called ideal functionalities. Composition theorems then allow to replace the ideal functionalities by their realizations, altogether resulting in a system without idealized components. Typically, the higher-level components are shown to realize idealized functionalities themselves. Hence, they can be used as low-level idealized components in even more complex systems.

⋆ This work was partially supported by the DFG under Grant KU 1434/5-1 and KU 1434/6-1.

This appealing approach has so far, however, been only rarely applied to real-world protocols (see the related work). One crucial obstacle has been the lack of suitable idealized functionalities and corresponding realizations for the most basic cryptographic primitives. While functionalities for public-key encryption and digital signatures have been proposed early on [8,30,23], only recently a functionality for symmetric encryption, which we denote by $\mathcal{F}_{\text{enc}}$ here, was proposed [25]. This functionality allows parties to generate symmetric and public/private keys and to use these keys for ideal encryption and decryption. The encrypted messages may contain symmetric keys and parties are given the actual ciphertexts, as bit strings. To bootstrap encryption with symmetric keys, $\mathcal{F}_{\text{enc}}$ also enables parties to generate and use pre-shared keys as well as public/private key pairs.

However, by itself $\mathcal{F}_{\text{enc}}$ is still insufficient for the analysis of many real-world protocols. The main goal of our work is therefore to augment this functionality (and its realization) with further primitives employed in real-word protocols and to develop suitable proof techniques in order to be able to carry out manageable, composable, yet faithful analysis of such protocols.

Contribution of this Paper. The first main contribution of this paper is to extend $\mathcal{F}_{\text{enc}}$ by a mechanism for key derivation, which is employed in virtually every real-word security protocol, as well as by MACs, digital signatures, and nonce generation; we call the new functionality $\mathcal{F}_{\text{crypto}}$. We show that, for a reasonable class of environments, $\mathcal{F}_{\text{crypto}}$ can be realized based on standard cryptographic assumptions and constructions: IND-CCA secure or authenticated encryption, UF-CMA secure MACs and digital signatures, and pseudo-random functions for key derivation, which are common also in implementations of real-world protocols. To prove this result, we extend the hybrid argument for $\mathcal{F}_{\text{enc}}$ in [25]. Since $\mathcal{F}_{\text{crypto}}$ is a rather low-level ideal functionality and its realization is based on standard cryptographic assumptions and constructions, it is widely applicable (see below and [25,24]) and allows for a precise modeling of real-word security protocols, including precise modeling of message formats on the bit level.

The second main contribution of our paper are criteria for protocols to provide universally composable key exchange and secure channels. These criteria are based on our ideal functionality $\mathcal{F}_{\text{crypto}}$, and therefore, can be checked merely using information-theoretic arguments, rather than much more involved and harder to manage reduction proofs; often even purely syntactical arguments suffice, without reasoning about probabilities. Indeed, the use of $\mathcal{F}_{\text{crypto}}$ tremendously simplifies proofs in the context of real-world security protocols, as demonstrated by our case study (see below), and in other contexts (see, e.g., [25,24]). Without $\mathcal{F}_{\text{crypto}}$, such proofs quickly become unmanageable.

The third main contribution of this paper is a case study in which we analyze central components of the wireless networking protocol WPA2, which implements the standard IEEE 802.11i [20]. More precisely, we analyze the 4-Way Handshake protocol (4WHS) for key exchange and the CCM Protocol (CCMP) of the pre-shared key mode of WPA2 (WPA2-PSK) for secure channels. Based on $\mathcal{F}_{\text{crypto}}$ and our criteria, we show that 4WHS realizes a universally composable key exchange functionality and that 4WHS with CCMP realizes a universally composable secure channel functionality; we note that 4WHS with TKIP (instead of CCMP) has recently been shown to be insecure [32,29]. Since we use standard cryptographic assumptions and constructions, our modeling of 4WHS and CCMP, including the message formats, is very close to the actual protocol. As to the best of our knowledge, this constitutes the first rigorous cryptographic analysis of these protocols. The framework presented in this paper would also allow us to analyze other real-world security protocols in a similar way, including several modes of Kerberos, SSL/TLS, DNSSEC, and EAP.

Structure of this Paper. In Section 2, we first recall the model for simulation-based security that we use. The functionality $\mathcal{F}_{\text{crypto}}$ and its realization are presented in Section 3. The criteria for secure key exchange and secure channel protocols are established in Section 4. Our case study is presented in Section 5. We conclude with related work in Section 6. Further details and all proofs are provided in our technical report [26].

2 Simulation-Based Security

In this section, we briefly recall the IITM model for simulation-based security (see [22] for details). In this model, security notions and composition theorems are formalized based on a relatively simple, but expressive general computational model in which IITMs (inexhaustible interactive Turing machines) and systems of IITMs are defined. While being in the spirit of Canetti's UC model [10], the IITM model has several advantages over the UC model and avoids some technical problems, as demonstrated and discussed in [22,23,25,19].

2.1 The General Computational Model

The general computational model is defined in terms of systems of IITMs. An *inexhaustible interactive Turing machine (IITM) M* is a probabilistic polynomial-time Turing machine with named input and output tapes. The names determine how different IITMs are connected in a system of IITMs. An IITM runs in one of two modes, CheckAddress and Compute. The CheckAddress mode is used as a generic mechanism for addressing copies of IITMs in a system of IITMs, as explained below. The runtime of an IITM may depend on the length of the input received so far and in *every* activation an IITM may perform a polynomial-time computation; this is why these ITMs are called inexhaustible. However, in this extended abstract we omit the details of the definition of IITMs, as these details are not necessary to be able to follow the rest of the paper.

A *system* $\mathcal{S}$ of IITMs is of the form $\mathcal{S} = M_1 \mid \cdots \mid M_k \mid\ !M'_1 \mid \cdots \mid\ !M'_{k'}$ where the M_i and M'_j are IITMs such that the names of input tapes of different IITMs in the system are disjoint. We say that the machines M'_j are in the scope of a bang operator. This operator indicates that in a run of a system an unbounded number of (fresh) copies of a machine may be generated. Conversely, machines which are not in the scope of a bang operator may not be copied. Systems in which multiple copies of machines may be generated are often needed, e.g., in case of multi-party protocols or in case a system describes the concurrent execution of multiple instances of a protocol.

In a run of a system $\mathcal{S}$ at any time only one IITM is active and all other IITMs wait for new input; the first IITM to be activated in a run of $\mathcal{S}$ is the so-called master

IITM, of which a system has at most one. To illustrate runs of systems, consider, for example, the system $\mathcal{S} = M_1 \mid !M_2$ and assume that M_1 has an output tape named c, M_2 has an input tape named c, and M_1 is the master IITM. (There may be other tapes connecting M_1 and M_2.) Assume that in the run of $\mathcal{S}$ executed so far, one copy of M_2, say M_2', has been generated and that M_1 just sent a message m on tape c. This message is delivered to M_2' (as the first, and, in this case, only copy of M_2). First, M_2' runs in the CheckAddress mode with input m; this is a deterministic computation which outputs "accept" or "reject". If M_2' accepts m, then M_2' gets to process m and could, for example, send a message back to M_1. Otherwise, a new copy M_2'' of M_2 with fresh randomness is generated and M_2'' runs in CheckAddress mode with input m. If M_2'' accepts m, then M_2'' gets to process m. Otherwise, M_2'' is removed again, the message m is dropped, and the master IITM is activated, in this case M_1, and so on. The master IITM is also activated if the currently active IITM does not produce output, i.e., stops in this activation without writing to any output tape. A run stops if the master IITM does not produce output (and hence, does not trigger another machine) or an IITM outputs a message on a tape named decision. Such a message is considered to be the overall output of the system.

We will consider so-called well-formed systems, which satisfy a simple syntactic condition that guarantees polynomial runtime of a system.

Two systems $\mathcal{P}$ and $\mathcal{Q}$ are called *indistinguishable* ($\mathcal{P} \equiv \mathcal{Q}$) iff the difference between the probability that $\mathcal{P}$ outputs 1 (on the decision tape) and the probability that $\mathcal{Q}$ outputs 1 (on the decision tape) is negligible in the security parameter.

2.2 Notions of Simulation-Based Security

We need the following terminology. For a system $\mathcal{S}$, the input/output tapes of IITMs in $\mathcal{S}$ that do not have a matching output/input tape are called *external*. These tapes are grouped into *I/O* and *network tapes*. We consider three different types of systems, modeling i) real and ideal protocols/functionalities, ii) adversaries and simulators, and iii) environments: *Protocol systems* and *environmental systems* are systems which have an I/O and network interface, i.e., they may have I/O and network tapes. *Adversarial systems* only have a network interface. Environmental systems may contain a master IITM. We can now define strong simulatability; other equivalent security notions, such as black-box simulatability and (dummy) UC can be defined in a similar way [22].

Definition 1 ([22]). *Let $\mathcal{P}$ and $\mathcal{F}$ be protocol systems with the same I/O interface, the real and the ideal protocol, respectively. Then, $\mathcal{P}$* realizes *$\mathcal{F}$ ($\mathcal{P} \leq \mathcal{F}$) iff there exists an adversarial system $\mathcal{S}$ (a simulator or ideal adversary) such that the systems $\mathcal{P}$ and $\mathcal{S} \mid \mathcal{F}$ have the same external interface and for all environmental systems $\mathcal{E}$, connecting only to the external interface of $\mathcal{P}$ (and hence, $\mathcal{S} \mid \mathcal{F}$) it holds that $\mathcal{E} \mid \mathcal{P} \equiv \mathcal{E} \mid \mathcal{S} \mid \mathcal{F}$.*

2.3 Composition Theorems

We restate the composition theorems from [22]. The first composition theorem handles concurrent composition of a fixed number of protocol systems. The second one guarantees secure composition of an unbounded number of copies of a protocol system. These theorems can be applied iteratively to construct more and more complex systems.

Theorem 1 **([22]).** *Let $\mathcal{P}_1, \mathcal{P}_2, \mathcal{F}_1, \mathcal{F}_2$ be protocol systems such that $\mathcal{P}_1$ and $\mathcal{P}_2$ as well as $\mathcal{F}_1$ and $\mathcal{F}_2$ only connect via their I/O interfaces, $\mathcal{P}_1 \,|\, \mathcal{P}_2$ and $\mathcal{F}_1 \,|\, \mathcal{F}_2$ are well-formed, and $\mathcal{P}_i \leq \mathcal{F}_i$, for $i \in \{1, 2\}$. Then, $\mathcal{P}_1 \,|\, \mathcal{P}_2 \leq \mathcal{F}_1 \,|\, \mathcal{F}_2$.*

In the following theorem, $\underline{\mathcal{F}}$ and $\underline{\mathcal{P}}$ are the so-called session versions of $\mathcal{F}$ and $\mathcal{P}$, which allow an environment to address different sessions of $\mathcal{F}$ and $\mathcal{P}$, respectively, in the multi-session versions $!\underline{\mathcal{F}}$ and $!\underline{\mathcal{P}}$ of $\mathcal{F}$ and $\mathcal{P}$.

Theorem 2 **([22]).** *Let $\mathcal{P}, \mathcal{F}$ be protocol systems such that $\mathcal{P} \leq \mathcal{F}$. Then, $!\underline{\mathcal{P}} \leq !\underline{\mathcal{F}}$.*

3 Our Crypto Functionality

In this section, we describe our ideal crypto functionality $\mathcal{F}_{\text{crypto}}$ and show that it can be realized under standard cryptographic assumptions (see [26] for details).

As mentioned in the introduction, $\mathcal{F}_{\text{crypto}}$ extends $\mathcal{F}_{\text{enc}}$, proposed in [25], by key derivation, MACs, digital signatures, and ideal nonce generation; also pre-shared keys can now be used just as other symmetric keys. More precisely, parties can use $\mathcal{F}_{\text{crypto}}$ i) to generate symmetric keys, including pre-shared keys, ii) to generate public/private keys, iii) to derive symmetric keys from other symmetric keys, iv) to encrypt and decrypt bit strings (public-key encryption and both unauthenticated and authenticated symmetric encryption is supported), v) to compute and verify MACs and digital signatures, and vi) to generate fresh nonces, where all the above operations are done in an ideal way. All symmetric and public keys can be part of plaintexts to be encrypted under other symmetric and public keys. We emphasize that derived keys can be used just as other symmetric keys. We also note that the functionality can handle an unbounded number of commands for an unbounded number of parties with the messages, ciphertexts, MACs, etc. being arbitrary bit strings of arbitrary length. We leave it up to the protocol that uses $\mathcal{F}_{\text{crypto}}$ how to interpret (parts of) bit strings, e.g., as length fields, nonces, ciphertexts, MACs, non-interactive zero-knowledge proofs etc. Since users of $\mathcal{F}_{\text{crypto}}$ are provided with actual bit strings, $\mathcal{F}_{\text{crypto}}$ can be combined with other functionalities too, including those of interest for real-word protocols, e.g., certification of public keys (see, e.g., [9]).

3.1 The Ideal Crypto Functionality

The ideal crypto functionality $\mathcal{F}_{\text{crypto}}$ is parametrized by what we call a *leakage algorithm L*, a probabilistic polynomial time algorithm which takes as input a security parameter η and a message m, and returns the information that may be leaked about m. Typical examples are i) $L(1^\eta, m) = 0^{|m|}$ and ii) the algorithm that returns a random bit string of length $|m|$. Both leakage algorithms leak exactly the length of m. The functionality $\mathcal{F}_{\text{crypto}}$ is also parameterized by a number n which defines the number of roles in a protocol that uses $\mathcal{F}_{\text{crypto}}$ (e.g., $n = 3$ for protocols with initiator, responder, and key distribution server); $\mathcal{F}_{\text{crypto}}$ has one I/O input and output tape for each role.

In $\mathcal{F}_{\text{crypto}}$, symmetric keys are equipped with types. Keys that may be used for authenticated encryption have type authenc-key, those for unauthenticated encryption have type unauthenc-key. We have the types mac-key for MAC keys and pre-key for keys from which new keys can be derived. All types are disjoint, i.e., a key can only have one type, reflecting common practice that a symmetric key only serves one purpose. For example, a MAC key is not used for encryption and keys from which other keys are derived are typically not used as encryption/MAC keys.

While users of $\mathcal{F}_{\text{crypto}}$, and its realization, are provided with the actual public keys generated within $\mathcal{F}_{\text{crypto}}$ (the corresponding private keys remain in $\mathcal{F}_{\text{crypto}}$), they do not get their hands on the actual symmetric keys stored in the functionality, but only on pointers to these keys, since otherwise no security guarantees could be provided. These pointers may be part of the messages given to $\mathcal{F}_{\text{crypto}}$ for encryption. Before a message is actually encrypted, the pointers are replaced by the keys they refer to. Upon decryption of a ciphertext, keys embedded in the plaintext are first turned into pointers before the plaintext is given to the user. In order to be able to identify pointers/keys, we assume pointers/keys in plaintexts to be tagged according to their types. We speak of *well-tagged* messages. For real-world protocols, including those mentioned in the introduction, it is typically possible to define tagging in such a way that the message formats used in these protocols is captured precisely on the bit level, as demonstrated by our case study in Section 5.

A *user* of $\mathcal{F}_{\text{crypto}}$ is identified, within $\mathcal{F}_{\text{crypto}}$, by the tuple $(p, lsid, r)$, where p is a party name, $r \leq n$ a role, and $lsid$ a local session ID (LSID), which is chosen and managed by the party itself. In particular, on the tape for role r, $\mathcal{F}_{\text{crypto}}$ expects requests to be prefixed by tuples of the form $(p, lsid)$, and conversely $\mathcal{F}_{\text{crypto}}$ prefixes answers with $(p, lsid)$.

The functionality $\mathcal{F}_{\text{crypto}}$ keeps track of which user has access to which symmetric keys (via pointers) and which keys are known to the environment/adversary, i.e., have been corrupted or have been encrypted under a known key, and as a result became known. For this purpose, among others, $\mathcal{F}_{\text{crypto}}$ maintains a set $\mathcal{K}$ of all symmetric keys stored within $\mathcal{F}_{\text{crypto}}$, a set $\mathcal{K}_{\text{known}} \subseteq \mathcal{K}$ of *known* keys, and a set $\mathcal{K}_{\text{unknown}} := \mathcal{K} \setminus \mathcal{K}_{\text{known}}$ of unknown keys.

Before any cryptographic operation can be performed, $\mathcal{F}_{\text{crypto}}$ expects to receive (descriptions of) algorithms from the ideal adversary for symmetric and public-key encryption/decryption as well as the generation and verification of MACs and digital signatures. Also, $\mathcal{F}_{\text{crypto}}$ expects to receive public/private key pairs for encryption/decryption and signing/verification for every party from the adversary. The adversary may decide to statically corrupt a public/private key of a party at the moment she provides it to $\mathcal{F}_{\text{crypto}}$. In this case $\mathcal{F}_{\text{crypto}}$ records the public/private key of this party as corrupted. We do not put any restrictions on these algorithms and keys; all security guarantees that $\mathcal{F}_{\text{crypto}}$ provides are made explicit within $\mathcal{F}_{\text{crypto}}$ without relying on specific properties of these algorithms. As a result, when using $\mathcal{F}_{\text{crypto}}$ in the analysis of systems, one can abstract from these algorithms entirely. We now sketch the operations that $\mathcal{F}_{\text{crypto}}$ provides.

Generating fresh, symmetric keys [(New, t)]**.** A user $(p, lsid, r)$ can ask $\mathcal{F}_{\text{crypto}}$ to generate a new key of type $t \in \{\mathsf{authenc\text{-}key}, \mathsf{unauthenc\text{-}key}, \mathsf{mac\text{-}key}, \mathsf{pre\text{-}key}\}$. The request is forwarded to the adversary who is supposed to provide such a key, say the bit string k. The adversary can decide to corrupt k right away, in which case k is added to $\mathcal{K}_{\text{known}}$, and otherwise k is added to $\mathcal{K}_{\text{unknown}}$. However, if k is uncorrupted, before adding k to $\mathcal{K}_{\text{unknown}}$, $\mathcal{F}_{\text{crypto}}$ verifies that k is fresh, i.e., k does not belong to $\mathcal{K}$. If k is corrupted, before adding k to $\mathcal{K}_{\text{known}}$, $\mathcal{F}_{\text{crypto}}$ verifies that k does not belong to $\mathcal{K}_{\text{unknown}}$. If $\mathcal{F}_{\text{crypto}}$ accepts k, a new pointer ptr to k is created (by increasing a counter) and returned to $(p, lsid, r)$. We emphasize that the difference between $\mathcal{K}_{\text{known}}$ and $\mathcal{K}_{\text{unknown}}$ is not whether or not the adversary knows the value of a key (it provides these values anyway). The point is that operations performed with unknown keys are ideal (see below). In the realization of $\mathcal{F}_{\text{crypto}}$, however, keys in $\mathcal{K}_{\text{unknown}}$ will of course not be known to the adversary.

Establishing pre-shared keys [$(\mathsf{GetPSK}, t, name)$]**.** This request is similar to (New, t). However, if $\mathcal{F}_{\text{crypto}}$ already recorded a key under $(t, name)$, a new pointer to this key is returned. In particular, if different users invoke this command with the same name and type, they are provided with pointers to the same key. This allows users to establish shared keys: For example, for WPA (see Section 5), requests of suppliers (e.g., laptops) and authenticators (e.g., access points) are of the form $(\mathsf{GetPSK}, t, kid)$, where kid is a key ID (instances of) suppliers and authenticators obtain from the environment (e.g., a system administrator) upon initialization.

Key derivation [$(\mathsf{Derive}, ptr, t, s)$]**.** A user $(p, lsid, r)$ can ask to derive a key of type $t \in \{\mathsf{authenc\text{-}key}, \mathsf{unauthenc\text{-}key}, \mathsf{mac\text{-}key}, \mathsf{pre\text{-}key}\}$ from a seed s (an arbitrary bit string) and a key, say k, of type $\mathsf{pre\text{-}key}$ the pointer ptr, which has to belong to the user, points to. If there already exists a key derived from k and s—a fact that $\mathcal{F}_{\text{crypto}}$ keeps track of—, a new pointer to this key is returned. Otherwise, a new key, similarly to the request (New, t) is generated. However, the adversary may not corrupt this key; it is considered to be unknown if and only if k is unknown.

Encryption [(Enc, ptr, x)] **and decryption** [(Dec, ptr, y)]**.** We concentrate on authenticated encryption and decryption (see [26] for unauthenticated and public-key encryption and decryption). A user $(p, lsid, r)$ can ask to encrypt a well-tagged message x using a pointer ptr that has to belong to the user and points to a key, say k, of type $\mathsf{authenc\text{-}key}$. We first consider the case that $k \in \mathcal{K}_{\text{unknown}}$. First, all pointers in x, which again have to belong to the user, are replaced by the actual keys, resulting in a message x'. Then, the *leakage* $\overline{x} = L(1^\eta, x')$ of x' is encrypted under k using the encryption algorithm previously provided by the adversary (see above). The resulting ciphertext y (if any) is returned to the user and (x', y) is stored by $\mathcal{F}_{\text{crypto}}$ for later decryption of y under k. Decryption of a ciphertext y, an arbitrary bit string, under a key k (as above), in fact only succeeds if for y exactly one pair of the form (x', y) is stored in $\mathcal{F}_{\text{crypto}}$. If $k \in \mathcal{K}_{\text{known}}$, the encryption and decryption algorithms provided by the adversary are applied to x' (rather than to $\overline{x} = L(1^\eta, x')$) and y, respectively.

Computing and verifying MACs [(Mac, ptr, x) and $(\mathsf{MacVerify}, ptr, x, \sigma)$]**.** A user $(p, lsid, r)$ can ask $\mathcal{F}_{\text{crypto}}$ to MAC an *arbitrary (uninterpreted) bit string* x using a pointer ptr that has to belong to the user and points to a key, say k, of type $\mathsf{mac\text{-}key}$. Then, $\mathcal{F}_{\text{crypto}}$ computes the MAC of x under k using the MAC algorithm previously provided by the adversary. The resulting MAC σ (if any) is returned to the user. If $k \in \mathcal{K}_{\text{unknown}}$, $\mathcal{F}_{\text{crypto}}$ records x for later verification with k; σ is not recorded since we allow an adversary to derive a new MAC from a given one on the same message.

For verification, $\mathcal{F}_{\text{crypto}}$ runs the MAC verification algorithm previously provided by the adversary on x, σ, and k. If $k \in \mathcal{K}_{\text{known}}$, $\mathcal{F}_{\text{crypto}}$ returns the result of the verification to the user. If $k \in \mathcal{K}_{\text{unknown}}$, this is done too, but success is only returned if x previously has been recorded for k.

Generating fresh nonces [(NewNonce)]. Similarly to generating fresh keys, nonces can be generated by users, where uncorrupted nonces are guaranteed to not collide.

Further operations. For further operations, including computing and verifying digital signatures, requests to obtain public keys, storing and retrieving of symmetric keys, checking the corruption status of keys, and checking whether two pointers point to the same key, we refer the reader to [26].

As illustrated by our case study, $\mathcal{F}_{\text{crypto}}$ is a convenient and easy to use tool for analyzing (real-world) security protocols. We note that, as explained above, corruption is modeled on a per key basis. This allows to model many types of corruption, including corruption of single sessions and of complete parties (see Section 5 for an example).

3.2 Realizing the Ideal Crypto Functionality

Let $\Sigma_{\text{unauthenc}}$, Σ_{authenc}, Σ_{pub} be schemes for symmetric and public-key encryption, respectively, Σ_{mac} be a MAC scheme, Σ_{sig} be a digital signature scheme, and $F = \{F_\eta\}_{\eta\in\mathbb{N}}$ be a family of pseudo-random functions with $F_\eta\colon \{0,1\}^\eta \times \{0,1\}^* \to \{0,1\}^\eta$ for all $\eta \in \mathbb{N}$. For simplicity of presentation, we assume keys to be chosen uniformly at random from $\{0,1\}^\eta$.

These schemes induce a realization $\mathcal{P}_{\text{crypto}}$ of $\mathcal{F}_{\text{crypto}}$ in the obvious way: The realization $\mathcal{P}_{\text{crypto}}$ maintains keys and pointers to keys in the same way as $\mathcal{F}_{\text{crypto}}$ does, but it does not maintain the sets $\mathcal{K}_{\text{known}}$ and $\mathcal{K}_{\text{unknown}}$. However, it is recorded whether a key is corrupted. Uncorrupted keys are honestly generated within $\mathcal{P}_{\text{crypto}}$ whereas corrupted keys are provided by the adversary. All ideal operations are replaced by their real counterparts in the natural way. Key derivation for a key k and a seed s is realized by computing F_η on k and s.

One cannot prove that $\mathcal{P}_{\text{crypto}}$ realizes $\mathcal{F}_{\text{crypto}}$ for standard assumptions about the symmetric encryption schemes $\Sigma_{\text{unauthenc}}$ and Σ_{authenc}, namely IND-CCA security and authenticated encryption (IND-CPA and INT-CTXT security), respectively, because it is easy to see that such a theorem does not hold in the presence of environments that may produce so-called key cycles (see, e.g., [6,2]) or cause the so-called commitment problem (see, e.g., [2]). Therefore, similar to [25] and [2], we restrict the class of environments that we consider basically to those environments that do not produce key cycles or cause the commitment problem. More precisely, to formulate such a class of environments that captures what is typically encountered in applications, we observe, as was first pointed in [2], that once a key has been used in a protocol to encrypt a message, this key is typically not encrypted anymore in the rest of the protocol. Let us call these protocols *standard*; for example, WPA can trivially be seen to be standard (see Section 5). This observation can be generalized to *used-order respecting environments*, which we formulate based on $\mathcal{F}_{\text{crypto}}$: An environment $\mathcal{E}$ (for $\mathcal{F}_{\text{crypto}}$) is called *used-order respecting* if it happens only with negligible probability that, in a run of

$\mathcal{E} \,|\, \mathcal{F}_{\text{crypto}}$, an unknown key k (i.e., k is marked unknown in $\mathcal{F}_{\text{crypto}}$) which has been used at some point (for encryption or key derivation, in case of keys of type unauthenc-key also for decryption) is encrypted itself by an unknown key k' used for the first time later than k. Clearly, such environments do not produce key cycles among unknown keys, with overwhelming probability. (We do not need to prevent key cycles among known keys.) We say that an environment $\mathcal{E}$ *does not cause the commitment problem (is non-committing)*, if it happens only with negligible probability that, in a run of $\mathcal{E} \,|\, \mathcal{F}_{\text{crypto}}$, after an unknown key k has been used to encrypt a message or to derive a new key, k becomes known later on in the run, i.e., is marked known by $\mathcal{F}_{\text{crypto}}$. It is easy to see that for standard protocols, as introduced above, the commitment problem does not occur.

We can now state the theorem, which shows that $\mathcal{F}_{\text{crypto}}$ *exactly* captures IND-CCA security, authenticated encryption, and UF-CMA security. In the theorem, instead of explicitly restricting the class of environments introduced above, we use a functionality $\mathcal{F}^*$ that provides exactly the same I/O interface as $\mathcal{F}_{\text{crypto}}$ (and hence, $\mathcal{P}_{\text{crypto}}$), but before forwarding requests to $\mathcal{F}_{\text{crypto}}/\mathcal{P}_{\text{crypto}}$ checks whether the used-order is still respected and the commitment problem is not caused. Otherwise, $\mathcal{F}^*$ raises an error flag and from then on blocks all messages, i.e., effectively stops the run.

Theorem 3. *Let $\Sigma_{\text{unauthenc}}, \Sigma_{\text{authenc}}, \Sigma_{\text{pub}}$ be encryption schemes as above, where the domain of plaintexts is the set of well-tagged bit strings. Let Σ_{mac} be a MAC scheme, Σ_{sig} be a digital signature scheme, and F be a pseudo-random function family as above. Let L be a leakage algorithm which leaks exactly the length of a message. Then, $\mathcal{F}^* \,|\, \mathcal{P}_{\text{crypto}} \leq \mathcal{F}^* \,|\, \mathcal{F}_{\text{crypto}}$ if and only if $\Sigma_{\text{unauthenc}}$ and Σ_{pub} are IND-CCA, Σ_{authenc} is IND-CPA and INT-CTXT, and Σ_{mac} and Σ_{sig} are UF-CMA secure. (The direction from right to left holds for any plaintext domains of the encryption schemes.)*

Since derived keys can be encrypted and used as encryption keys, the security of encryption depends on the security of key derivation and vice versa. Therefore, in the proof of the above theorem we need to carry out a single hybrid argument, intertwining both encryption and key derivation (see [26] for details).

The following corollary shows that if a protocol system $\mathcal{P}$ that uses $\mathcal{F}_{\text{crypto}}$ is *non-committing* and *used-order respecting*, i.e., $\mathcal{E} \,|\, \mathcal{P}$ is a *non-committing, used-order respecting* environment for all environment systems $\mathcal{E}$, then $\mathcal{F}^*$ can be omitted. As mentioned above, most protocols, including standard protocols, have this property and this can typically be easily checked by inspection of the protocol (see Section 5 for an example).

Corollary 1. *Let $\Sigma_{\text{unauthenc}}$, Σ_{authenc}, Σ_{pub}, Σ_{mac}, Σ_{sig}, F, and L be given as in Theorem 3. Let $\mathcal{P}$ be a non-committing, used-order respecting protocol system. Then, $\mathcal{P} \,|\, \mathcal{P}_{\text{crypto}} \leq \mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$ if $\Sigma_{\text{unauthenc}}$ and Σ_{pub} are IND-CCA, Σ_{authenc} is IND-CPA and INT-CTXT, and Σ_{mac} and Σ_{sig} are UF-CMA secure.*

As demonstrated in the following sections, using Theorem 3 and Corollary 1 protocols can first be analyzed based on $\mathcal{F}_{\text{crypto}}$ and then $\mathcal{F}_{\text{crypto}}$ can be replaced by its realization $\mathcal{P}_{\text{crypto}}$. We note that the joint state composition theorems for public-key encryption and symmetric encryption under pre-shared keys in [25] carry over to $\mathcal{F}_{\text{crypto}}$. That is, we can prove that a—so called—*joint state realization* of $\mathcal{F}_{\text{crypto}}$ realizes the multi-session version of $\mathcal{F}_{\text{crypto}}$. However, as explained in Section 4, we do not use composition with joint state in this paper.

4 Applications to Key Exchange and Secure Channels

In this section, we consider a general class of key exchange and secure channel protocols which use the functionality $\mathcal{F}_{\text{crypto}}$ (or its realization $\mathcal{P}_{\text{crypto}}$) and develop criteria to prove universally composable security for such protocols. Since our criteria are based on $\mathcal{F}_{\text{crypto}}$, proving the criteria merely requires information-theoretic arguments or purely syntactical arguments (without reasoning about probabilities), rather than involved cryptographic reduction proofs.

Our criteria are formulated w.r.t. multiple protocol sessions. Alternatively, we could formulate them for single sessions and then extend them to the multi-session case by joint state theorems [13,23,25]. However, in order for our models to be very close to the actual (real-world) protocols, in this paper, we avoid these theorems: First, they rely on the setup assumption that the parties participating in a session already agreed upon a unique session identifier (SID). Real-world protocols do not rely on this assumption. Second, in joint state realizations, SIDs are explicitly added to messages before encryption, signing, and MACing, i.e., in a session with SID *sid*, instead of the actual message, say m, the message (sid, m) is encrypted, signed, or MACed. While this is a good design principle, it modifies the actual protocols.

4.1 A Criterion for Universally Composable Key Exchange

We define an ideal functionality for (multi-session) key exchange $\mathcal{F}_{\text{ke}}$, formulate a general class of key exchange protocols that use $\mathcal{F}_{\text{crypto}}$ for cryptographic operations, and present a criterion which allows us to prove that a key exchange protocol in this class realizes $\mathcal{F}_{\text{ke}}$.

The Ideal Key Exchange Functionality. The basic idea of an ideal functionality for key exchange $\mathcal{F}_{\text{ke}}$, see, e.g., [10], is that parties can send requests to $\mathcal{F}_{\text{ke}}$ to exchange a key with other parties and then, in response, receive a session key which is generated by $\mathcal{F}_{\text{ke}}$ and guaranteed to be i) the same for every party in the same session and ii) only known to these parties. As mentioned above and unlike other formulations, our functionality directly allows to handle an unbounded number of sessions between arbitrary parties.

More precisely, similarly to $\mathcal{F}_{\text{crypto}}$, our ideal key exchange functionality $\mathcal{F}_{\text{ke}}$ is parametrized by a number n which specifies the number of roles, e.g., $n = 2$ in case of a two-party key exchange protocol. To address multiple sessions of a party, the parties identify themselves to $\mathcal{F}_{\text{ke}}$ as a *user* (similarly to $\mathcal{F}_{\text{crypto}}$), represented by a tuple $(p, lsid, r)$, where p is the PID of the party, *lsid* a local session ID chosen and managed by the party itself, and the role $r \in \{1, \ldots, n\}$. For every user a corresponding *local session* is managed in $\mathcal{F}_{\text{ke}}$, which contains the state of the key exchange for this user. To initiate a key exchange, a user, say $(p, lsid, r)$, can send a *session-start* message of the form $(\mathsf{Start}, p_1, \ldots, p_n)$, with $p = p_r$, to $\mathcal{F}_{\text{ke}}$, where the PIDs $p_1, \ldots, p_n$ are the desired partners of p in the n roles for the key exchange. Upon such a request, $\mathcal{F}_{\text{ke}}$

records this *session-start* message as a local session for user $(p, lsid, r)$ and informs the (ideal) adversary about this request by forwarding it to her. The adversary determines (at some point) to which *global* session local sessions belong, by sending a *session-create* message of the form $(\mathsf{Create}, (p_1, lsid_1, 1), \ldots, (p_n, lsid_n, n))$ to $\mathcal{F}_{\text{ke}}$, containing one local session for every role. The functionality $\mathcal{F}_{\text{ke}}$ only accepts such a message if it is consistent with the local sessions: The mentioned local sessions all exist, are uncorrupted (see below) and are not already part of another global session, and the desired partners in the local sessions correspond to each other. For a global session, $\mathcal{F}_{\text{ke}}$ creates a fresh key—called the *session key*—according to some probability distribution. For a local session $(p, lsid, r)$ which is part of a global session in $\mathcal{F}_{\text{ke}}$, the adversary can send a *session-finish* message of the form $(\mathsf{Finish}, (p, lsid, r))$ to $\mathcal{F}_{\text{ke}}$, upon which $\mathcal{F}_{\text{ke}}$ sends a *session-key-output* message of the form $(\mathsf{SessionKey}, k)$ to the user $(p, lsid, r)$, which contains the session key k for this session.

The adversary can corrupt a local session $(p, lsid, r)$ which is not already part of a global session by sending a *corrupt* message of the form $(\mathsf{Corrupt}, (p, lsid, r))$ to $\mathcal{F}_{\text{ke}}$. For a corrupted local session, the adversary may determine the session key by sending a *session-finish* message of the form $(\mathsf{Finish}, (p, lsid, r), k)$ to $\mathcal{F}_{\text{ke}}$, upon which $\mathcal{F}_{\text{ke}}$ sends a *session-key-output* message of the form $(\mathsf{SessionKey}, k)$ to the user $(p, lsid, r)$, which contains the session key k chosen by the adversary. As usual, the environment can ask whether a local session is corrupted or not.

Key Exchange Protocols. An $\mathcal{F}_{\text{crypto}}$*-key exchange protocol* ($\mathcal{F}_{\text{crypto}}$*-KE protocol)*, which is meant to realize $\mathcal{F}_{\text{ke}}$, is a protocol system $\mathcal{P}$ which connects to the I/O interface of $\mathcal{F}_{\text{crypto}}$ such that $\mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$ has the same I/O interface as $\mathcal{F}_{\text{ke}}$. The system $\mathcal{P}$ is of the form $\mathcal{P} = !M_1 \,|\, \ldots \,|\, !M_n$ for some n and machines (IITMs) $M_1, \ldots, M_n$. For every user $(p, lsid, r)$, there is one instance of M_r; intuitively, such an instance is meant to realize a local session in $\mathcal{F}_{\text{ke}}$. Every instance of M_r may arbitrarily communicate with the adversary (the network) and may use $\mathcal{F}_{\text{crypto}}$ in the name of the corresponding user.[1] Analogously to $\mathcal{F}_{\text{ke}}$, a user $(p, lsid, r)$ initiates a key exchange by sending a *session-start* message to (its instance of) M_r. At some point, every instance of M_r may return a *session-key-pointer-output* message of the form $(\mathsf{SessionKeyPointer}, ptr)$ to its user which contains a pointer ptr, called the *session key pointer*, to the actual session key stored in $\mathcal{F}_{\text{crypto}}$; so, unlike $\mathcal{F}_{\text{ke}}$, only a pointer to the session key, rather than the actual key, is output (see below for a variant of $\mathcal{P}$ in which, similar to $\mathcal{F}_{\text{ke}}$, the actual session key is given to the user). This instance then provides its user with an interface to $\mathcal{F}_{\text{crypto}}$ where initially only the session key pointer ptr may be used (but subsequently other pointers can be generated). More precisely, the user $(p, lsid, r)$ may send any request for $\mathcal{F}_{\text{crypto}}$ to M_r, such as encryption, decryption, and key derivation requests. Upon such a request, M_r forwards this request to $\mathcal{F}_{\text{crypto}}$ and waits for receiving an answer from $\mathcal{F}_{\text{crypto}}$, which is then forwarded to the user $(p, lsid, r)$. However, we require that all pointers in such a request have been output by M_r to this user before and that the session key pointer is never encrypted or explicitly revealed by a retrieve command (see below for an example). Before forwarding requests to $\mathcal{F}_{\text{crypto}}$, M_r checks whether this requirement is satisfied; if the check fails, M_r returns an error message to the user $(p, lsid, r)$.

[1] We note that an environment of $\mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$ cannot directly access the I/O interface of $\mathcal{F}_{\text{crypto}}$, but only via the IITMs $M_1, \ldots, M_n$.

For example, after having received ($\mathsf{SessionKeyPointer}, ptr$) from M_r, the user $(p, lsid, r)$ might send the request (New, t) to M_r upon which M_r will forward it to $\mathcal{F}_{\text{crypto}}$. Then, $\mathcal{F}_{\text{crypto}}$ will return a new pointer ptr' to M_r which is forwarded by M_r to the user $(p, lsid, r)$. To encrypt a message m which contains the pointer ptr' (and no other pointer, say) under the session key pointer ptr, $(p, lsid, r)$ sends the request (Enc, ptr, m) to M_r. Then, M_r will forward this message to $\mathcal{F}_{\text{crypto}}$ because all pointers in this request, i.e., ptr and ptr', have been output to this user before. Finally, the ciphertext returned by $\mathcal{F}_{\text{crypto}}$ is forwarded to the user $(p, lsid, r)$.

We do not fix a special form of corruption but leave the modeling of corruption to the definition of the protocol $\mathcal{P}$, up to the following conditions: i) the environment can ask about the corruption status of instances of M_r (this corresponds to the environment asking $\mathcal{F}_{\text{ke}}$ whether a local session is corrupted), ii) once an instance of M_r is corrupted, it stays corrupted, and iii) an instance of M_r cannot be corrupted after it has returned a *session-key-pointer-output* message. (See our case study in Section 5 for an example.)

We also consider a variant $\widehat{\mathcal{P}}$ of an $\mathcal{F}_{\text{crypto}}$-KE protocol $\mathcal{P}$ defined as follows: Instead of sending *session-key-pointer-output* messages, $\widehat{\mathcal{P}}$ sends *session-key-output* messages (as $\mathcal{F}_{\text{ke}}$) which contain the actual key the session key pointer refers to. This key is obtained using the retrieve command ($\mathsf{Retrieve}, ptr$) of $\mathcal{F}_{\text{crypto}}$. Furthermore, in contrast to $\mathcal{P}$, $\widehat{\mathcal{P}}$ does not provide the environment with an interface to $\mathcal{F}_{\text{crypto}}$, i.e., $\widehat{\mathcal{P}}$ does not forward requests to $\mathcal{F}_{\text{crypto}}$. We note that the protocol $\widehat{\mathcal{P}}$ is meant to realize $\mathcal{F}_{\text{ke}}$ (see below). The advantage of $\mathcal{P}$ over $\widehat{\mathcal{P}}$ is that a session key pointer can still be used for *ideal* cryptographic operations, e.g., ideal encryption or even to establish an ideal secure channel (see below).

We note that in [26] we consider a more general form of $\mathcal{F}_{\text{crypto}}$-KE protocols: We allow $\mathcal{P}$ and $\widehat{\mathcal{P}}$ to use (arbitrary) ideal functionalities $\mathcal{F}_1 \,|\, \dots \,|\, \mathcal{F}_l$ in addition to $\mathcal{F}_{\text{crypto}}$. These functionalities can provide additional cryptographic operations, such as public-key certification. As shown in [26], our criteria and all results obtained in this paper remain unchanged and carry over to these generalized $\mathcal{F}_{\text{crypto}}$-KE protocols.

Criterion for Secure Key Exchange Protocols. We now present a sufficient criterion for an $\mathcal{F}_{\text{crypto}}$-KE protocol to realize $\mathcal{F}_{\text{ke}}$, and hence, to provide universally composable key exchange. The criterion is based on partnering functions.[2]

A *partnering function* τ for an $\mathcal{F}_{\text{crypto}}$-KE protocol $\mathcal{P}$ is a polynomial-time computable function that maps a sequence of configurations of $\mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$ to a set of tuples of the form $(s_1, \dots, s_n)$, where s_r is of the form $(p, lsid, r)$, i.e., s_r refers to an instance of M_r, for all $r \leq n$. We say that the instances $s_1, \dots, s_n$ *form a (global) session according to* τ. We call τ *valid for* $\mathcal{P}$ if for any environment $\mathcal{E}$ for $\mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$ and any run of $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$ the following holds, where τ operates on the projection of the runs to configurations of $\mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$: i) All instances occur in at most one session (according to τ). ii) Instances in one session agree on the PIDs of the desired partners. iii) τ is monotonic, i.e., once a session has been established according to τ, it continues to exist. Now, we are ready to state our criterion.

[2] We note that partnering functions have been used in game-based security definitions (e.g., [4]). However, their use has been criticized in subsequent work (e.g., [3,21]). We emphasize that here, partnering functions are only part of our criterion but not part of the security definition.

Definition 2. *We say that an $\mathcal{F}_{\text{crypto}}$-KE protocol $\mathcal{P}$ is* strongly $\mathcal{F}_{\text{crypto}}$-secure (with type t_0 of a key) *if there exists a valid partnering function τ for $\mathcal{P}$ such that for every environment $\mathcal{E}$ for $\mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$ the following holds with overwhelming probability, where the probability is over runs of $\mathcal{E} \,|\, \mathcal{P} \,|\, \mathcal{F}_{\text{crypto}}$: For every uncorrupted instance of M_r, say $(p, lsid, r)$, which has output a session key pointer to say the key k in $\mathcal{F}_{\text{crypto}}$ it holds that:*

i) The local session $(p, lsid, r)$ belongs to some global session (according to τ) which contains only uncorrupted local sessions.
ii) The key k is of type t_0 and marked unknown in $\mathcal{F}_{\text{crypto}}$.
iii) The key k has never been used in $\mathcal{F}_{\text{crypto}}$ as a key for encryption, key derivation, or to compute a MAC by any user, except through the interface to $\mathcal{F}_{\text{crypto}}$ provided to the environment after a session-key-pointer-output message.
iv) Session key pointers (if any) of other instances point to the same key k if and only if they belong to the same session as $(p, lsid, r)$ (according to τ).

The following theorem states that this criterion is indeed sufficient for an $\mathcal{F}_{\text{crypto}}$-KE protocol to realize the ideal key exchange functionality $\mathcal{F}_{\text{ke}}$.

Theorem 4. *Let $\mathcal{P}$ be an $\mathcal{F}_{\text{crypto}}$-KE protocol. If $\mathcal{P}$ is strongly $\mathcal{F}_{\text{crypto}}$-secure and $\widehat{\mathcal{P}}$ is used-order respecting and non-committing, then $\widehat{\mathcal{P}} \,|\, \mathcal{P}_{\text{crypto}} \leq \mathcal{F}_{\text{ke}}$.*

4.2 Applications to Secure Channels

A secure channel, see, e.g., [12], between two parties provides confidentiality and authenticity of the messages sent over the channel and prevents rearrangement and replay of messages. Some secure channels also prevent message loss. In this section, we only briefly sketch our results; see [26] for details.

We define two ideal functionalities for secure channels $\mathcal{F}_{\text{sc}}$ and $\mathcal{F}_{\text{sc}}^{+}$, where, unlike $\mathcal{F}_{\text{sc}}$, $\mathcal{F}_{\text{sc}}^{+}$ prevents message loss. Just as $\mathcal{F}_{\text{ke}}$ and in contrast to previous formulations, our functionalities directly allow to handle an unbounded number of sessions between arbitrary parties.

We consider two generic realizations of $\mathcal{F}_{\text{sc}}$ and $\mathcal{F}_{\text{sc}}^{+}$, namely $\mathcal{P}_{\text{sc}}$ and $\mathcal{P}_{\text{sc}}^{+}$, respectively, which use an $\mathcal{F}_{\text{crypto}}$-key exchange protocol $\mathcal{P}$ as a sub-protocol. Every session of $\mathcal{P}_{\text{sc}}$ (analogously for $\mathcal{P}_{\text{sc}}^{+}$) runs a session of $\mathcal{P}$ to exchange a session key. This session key is then used to establish secure channels between the parties of the session, one channel for each pair of parties in that session. For this purpose, before a message is encrypted (using authenticated encryption) under the session key, the PIDs of the sender and receiver are added to the plaintexts as well as a counter.

We provide a criterion for $\mathcal{F}_{\text{crypto}}$-KE protocols and show that $\mathcal{P}_{\text{sc}}$ and $\mathcal{P}_{\text{sc}}^{+}$ realize $\mathcal{F}_{\text{sc}}$ and $\mathcal{F}_{\text{sc}}^{+}$, respectively, if the underlying $\mathcal{F}_{\text{crypto}}$-KE protocol $\mathcal{P}$ satisfies this criterion. While we could use "strongly $\mathcal{F}_{\text{crypto}}$-secure" as our criterion, a weaker criterion in fact suffices, which we call *α-$\mathcal{F}_{\text{crypto}}$-secure*. Unlike strong $\mathcal{F}_{\text{crypto}}$-security, α-$\mathcal{F}_{\text{crypto}}$-security allows that session keys are used in the key exchange protocol (e.g., for key confirmation), i.e., condition iii) in Definition 2 is dropped. But then, messages encrypted under these keys in the key exchange protocol should not interfere with messages sent over the secure channel. Instead of condition iii), we therefore consider a set α of messages and require that only messages in α are encrypted under the session key in the key exchange protocol. We note that strongly $\mathcal{F}_{\text{crypto}}$-secure protocols are $\emptyset$-$\mathcal{F}_{\text{crypto}}$-secure.

The following theorem states that α-$\mathcal{F}_{\text{crypto}}$-security is a sufficient condition for the generic secure channels protocols to realize the ideal secure channel functionalities, provided that plaintexts sent over the secure channel do not belong to α. Usually, the key exchange and the secure channel protocol use different message formats such that the messages cannot be confused, e.g., because of tagging with different protocol identifiers. In this case, an appropriate α can easily be defined.

Theorem 5. *Let $\mathcal{P}$ be an $\mathcal{F}_{\text{crypto}}$-KE protocol and α be a set of messages as above such that it does not contain any plaintext that is potentially encrypted by $\mathcal{P}_{\text{sc}}$ (or $\mathcal{P}^+_{\text{sc}}$). If $\mathcal{P}$ is α-$\mathcal{F}_{\text{crypto}}$-secure, then $\mathcal{P}_{\text{sc}} \,|\, \mathcal{P} \,|\, \mathcal{F}_{\text{crypto}} \leq \mathcal{F}_{\text{sc}}$ and $\mathcal{P}^+_{\text{sc}} \,|\, \mathcal{P} \,|\, \mathcal{F}_{\text{crypto}} \leq \mathcal{F}^+_{\text{sc}}$.*

5 Security Analysis of IEEE 802.11i

Using our results and methods developed in the previous sections, we now analyze two central protocols of WPA2-PSK (IEEE 802.11i) [20], namely the 4-Way Handshake (4WHS) protocol and the CCM Protocol (CCMP), with more details provided in [26]. We prove that 4WHS provides universally composable key exchange and that 4WHS with CCMP provides universally composable secure channels. Without $\mathcal{F}_{\text{crypto}}$, our modular approach, and our criteria, the proof would be considerably more complex and would involve non-trivial reduction proofs. In particular, due to $\mathcal{F}_{\text{crypto}}$, our proofs only require syntactic arguments and they illustrate that $\mathcal{F}_{\text{crypto}}$ can be used in an intuitive and easy way for the analysis of real-world security protocols.

5.1 The 4-Way Handshake Protocol

Description of the 4WHS Protocol. The 4-Way Handshake (4WHS) protocol consists of two roles, an authenticator A, e.g., an access point, and a supplicant S, e.g., a laptop, which share a Pairwise Master Key (PMK). The authenticator may communicate with several supplicants using the same PMK, which in WPA2-PSK is a pre-shared key (PSK). On an abstract level, the message exchange between an authenticator A and a supplicant S is shown in Figure 1, where p_A and p_S are the names (Media Access Control (MAC) addresses) of A and S, respectively, n_A and n_S are nonces generated by A and S, respectively, and $c_1, \ldots, c_4$ are pairwise distinct constants used to indicate different messages. From the PMK, A and S derive a Pairwise Transient Key PTK by computing PTK $= F($PMK, "Pairwise key expansion" $\|$ $\min(p_A, p_S)$ $\|$ $\max(p_A, p_S)$ $\|$ $\min(n_A, n_S)$ $\|$ $\max(n_A, n_S)))$, where F is an HMAC, which according to the IEEE 802.11i standard is assumed to be pseudo-random. The PTK is then split into the Key Confirmation Key (KCK), the Key Encryption Key (KEK), and the Temporary Key (TK), where TK is used in CCMP to establish a secure channel between A and S (see below).

Modeling the 4WHS Protocol. Modeling the 4WHS protocol as an $\mathcal{F}_{\text{crypto}}$-KE protocol is straightforward. We emphasize that, since $\mathcal{F}_{\text{crypto}}$ provides a low-level interface to basic cryptographic primitives with a very liberal use of tagging, our modeling of

1. $A \rightarrow S$: p_A, n_A, c_1
2. $S \rightarrow A$: $p_S, n_S, c_2, \mathrm{MAC}_{\mathrm{KCK}}(n_S, c_2)$
3. $A \rightarrow S$: $p_A, n_A, c_3, \mathrm{MAC}_{\mathrm{KCK}}(n_A, c_3)$
4. $S \rightarrow A$: $p_S, c_4, \mathrm{MAC}_{\mathrm{KCK}}(c_4)$

Fig. 1. The 4-Way Handshake Protocol of IEEE 802.11i

the 4WHS protocol, including message formats, the use of cryptographic primitives, and cryptographic assumptions, is quite close to the actual standard. We note that in our modeling of 4WHS parties may *not* play both the role of an authenticator and a supplicant with the same pre-shared key. Otherwise, 4WHS would be insecure. Indeed, a reflection attack would be possible [17], and our security proofs would fail.

The adversary can (statically) corrupt an instance of A or S, i.e., a local session, by sending a special corrupt message to it. This has to be the first message this instance receives from the adversary. A corrupted instance grants the adversary full control over its interface, including the interface it has to $\mathcal{F}_{\text{crypto}}$. If the instance is corrupted, all keys it has should be corrupted as well. We therefore require that the adversary corrupts all keys a corrupted instance creates using $\mathcal{F}_{\text{crypto}}$. A corrupted instance always checks (by asking $\mathcal{F}_{\text{crypto}}$) if its keys created in $\mathcal{F}_{\text{crypto}}$ indeed have been corrupted by the adversary and terminates if they have not been corrupted. Note that since keys in $\mathcal{F}_{\text{crypto}}$ of a corrupted instance are known, it is not a problem if the adversary generates key cycles or causes the commit problem with those keys. Conversely, uncorrupted instances always check that the key, PSK, and the nonce, n_A or n_S, they have created using $\mathcal{F}_{\text{crypto}}$ are uncorrupted at the time of their creation.

In the literature, (static) corruption is often modeled on a per party basis, i.e., if a party is corrupted, then all its keys are corrupted and the adversary is in full control of that party. We note that this is a special case of our modeling of corruption because the adversary can decide to corrupt all keys and local sessions of a corrupted party.

Security Analysis. We first show that 4WHS is strongly $\mathcal{F}_{\text{crypto}}$-secure.

Theorem 6. *The protocol* 4WHS *is strongly* $\mathcal{F}_{\text{crypto}}$*-secure with type* *authenc-key.*

Proof. First, we define a partnering function τ for 4WHS: Two instances are defined to form a session if a) they have different roles, namely A and S, respectively, b) they are both uncorrupted, c) the party names of the desired partner correspond to each other, d) they use the same pre-shared key, e) the values of the nonces correspond to each other, and f) one of them has already output a session key pointer. Because $\mathcal{F}_{\text{crypto}}$ guarantees that (uncorrupted) nonces are unique for every instance, there are at most two such instances, and hence, it is easy to see that τ is a valid partnering function for 4WHS.

It remains to show that 4WHS is strongly $\mathcal{F}_{\text{crypto}}$-secure w.r.t. τ and every environment $\mathcal{E}$ of 4WHS $|\, \mathcal{F}_{\text{crypto}}$: Let ρ be a run of $\mathcal{E} \,|\,$ 4WHS $|\, \mathcal{F}_{\text{crypto}}$ and let (p, lsid, r) be some uncorrupted instance (i.e., an instance of M_r) in ρ which has output a session key pointer to a key, say k, in $\mathcal{F}_{\text{crypto}}$, and which established the pre-shared key PSK and derived KCK and TK from it in $\mathcal{F}_{\text{crypto}}$.

First, we observe that, by our corruption model, since (p, lsid, r) is uncorrupted, PSK is uncorrupted (in $\mathcal{F}_{\text{crypto}}$). Also, every other instance that established PSK must be uncorrupted as well since keys created by corrupted instances are required to be corrupted. In uncorrupted instances, PSK is only used to derive keys, hence, PSK is always marked unknown in $\mathcal{F}_{\text{crypto}}$. In particular, no corrupted local session has a pointer to PSK. Now, by definition of $\mathcal{F}_{\text{crypto}}$, KCK and TK can only be derived by instances that have a pointer to PSK, leaving only uncorrupted instances. Moreover, again by $\mathcal{F}_{\text{crypto}}$, these uncorrupted instances have to use the same seed s as $(p, lsid, r)$, which contains the party names, p and p' say, and two nonces. Since uncorrupted nonces generated by $\mathcal{F}_{\text{crypto}}$ are guaranteed to be unique, by the construction of s, it follows that besides $(p, lsid, r)$ at most one other (uncorrupted) instance $(p', lsid', r')$, for some p', $lsid'$, and r', uses s, and hence, has a pointer to KCK and TK by derivation. By the definition of the protocol, uncorrupted instances only use KCK for MACing and TK is at most used after being output in a *session-key-pointer-output* message, but then TK may not be encrypted or retrieved. By definition of $\mathcal{F}_{\text{crypto}}$, it follows that KCK and TK are always marked unknown in $\mathcal{F}_{\text{crypto}}$ and only $(p, lsid, r)$ and, if present, $(p', lsid', r')$ have pointers to KCK and TK.

We now show that $(p', lsid', r')$ exists and that $(p, lsid, r)$ and $(p', lsid', r')$ belong to the same session (according to τ), which implies i) of Definition 2: We assume that $r = A$; the proof for $r = S$ is similar. The instance $(p, lsid, r)$ verified a MAC in a message of the form $p', n'', c_2, \text{MAC}_{\text{KCK}}(n'', c_2)$. Since $r = A$ and the constants c_2 and c_3 are distinct, $(p, lsid, r)$ has not created such a MAC. By definition of $\mathcal{F}_{\text{crypto}}$, $\text{MAC}_{\text{KCK}}(n'', c_2)$ can only have been created by some instance that has a pointer to KCK, which must be the (uncorrupted) instance $(p', lsid', r')$ from above. It follows that $r' = S$ since an uncorrupted instance with $r' = A$ would not create a MAC of such a form. By our assumption that a party does not play both the role of A and S with the same pre-shared key, it follows that $p' \neq p$. (Our assumption, and the implied fact, $p' \neq p$, is crucial; without it the proof would fail and in fact a reflection attack would be possible [17].) We can now show that $(p, lsid, r)$ and $(p', lsid', r')$ belong to the same session according to τ: We already know that conditions a), b), d), and f) for τ (as defined above) are satisfied. Since $p \neq p'$, it follows that the intended partner of $(p', lsid', r')$ is p, since, by definition of $\mathcal{F}_{\text{crypto}}$ and KCK, otherwise $(p', lsid', r')$ could not have derived KCK. So c) is satisfied. (Without our assumption mentioned above, this could not be concluded.) Similarly, condition e) is satisfied since otherwise the two instances would not have derived the same KCK.

We already know that TK ($= k$) is marked unknown in $\mathcal{F}_{\text{crypto}}$. This key is of type authenc-key because, by definition of the protocol, it has been derived as a key of this type. So ii) of Definition 2 follows.

We also know that only $(p, lsid, r)$ and $(p', lsid', r')$ have a pointer to TK in $\mathcal{F}_{\text{crypto}}$. Hence, iv) of Definition 2 follows. Since both instances are uncorrupted, by the definition of the protocol, iii) follows as well. $\square$

Trivially, $4\widehat{\text{WHS}}$ (recall that $4\widehat{\text{WHS}}$ outputs the session key instead of a pointer to it) is a standard protocol (as defined in Section 3), hence, it is used-order respecting and non-committing. Using Theorem 4 and 6, we immediately obtain that $4\widehat{\text{WHS}} \mid \mathcal{P}_{\text{crypto}}$ is a universally composable secure key exchange protocol.

Corollary 2. $4\widehat{\text{WHS}} \mid \mathcal{P}_{\text{crypto}} \leq \mathcal{F}_{\text{ke}}$.

5.2 The CCM Protocol

WPA2-PSK uses CCMP with the Temporal Key (TK), exchanged by running the 4WHS protocol, to establish a secure channel between the authenticator and the supplicant. CCMP can be modeled faithfully by $\mathcal{P}_{\mathrm{sc}}$ (see Section 4.2). By Theorem 5 and 6 we obtain that CCMP using 4WHS and $\mathcal{F}_{\mathrm{crypto}}$ is a universally composable secure channel protocol. Moreover, it is easy to see that CCMP | 4WHS is a standard protocol (as defined in Section 3), and hence, it is used-order respecting and non-committing. By Corollary 1, we then obtain:

Corollary 3. CCMP | 4WHS | $\mathcal{F}_{\mathrm{crypto}} \leq \mathcal{F}_{\mathrm{sc}}$ *and* CCMP | 4WHS | $\mathcal{P}_{\mathrm{crypto}} \leq \mathcal{F}_{\mathrm{sc}}$.

6 Related Work

Backes et al. (see, e.g., [2]) proposed a Dolev-Yao style cryptographic library. The main purpose of the library is to provide a Dolev-Yao style abstraction to the user, in the spirit of computational soundness results [27,15,1,24]. In contrast, our functionality provides a much lower-level idealization, aiming at wide applicability and faithful treatment of cryptographic primitives. More specifically, unlike $\mathcal{F}_{\mathrm{crypto}}$, based on the Dolev-Yao library only those protocols can be analyzed which merely use operations provided by the library (since the user, except for payload data, only gets his/her hands on pointers to Dolev-Yao terms in the library, rather than on the actual bit strings, internally everything is represented as terms too) and these protocols can only be shown to be secure w.r.t. non-standard encryption schemes (since, e.g., extra randomness and tagging with key identifiers is assumed for encryption schemes) and assuming specific message formats (all types of messages—nonces, ciphertexts, pairs of messages etc.—, are tagged in the realization). While the Dolev-Yao library considers symmetric encryption (key derivation is not considered at all) [2], it is an open problem whether there is a reasonable realization; the original proof of the realization of the crypto library in [2] is flawed, as examples presented in [14] illustrate (see also [25]).

Our criteria for secure key exchange and secure channel protocols presented in Section 4 are related to the concept of secretive protocols proposed by Roy et al. [31] (see also [25]). However, unlike our criteria, which can be checked based on information-theoretic/syntactical arguments, checking whether a protocol is secretive requires involved cryptographic reduction proofs. Also, Roy et al. do not prove implications for *composable* security and they do not consider secure channels.

The only work we are aware of that attempts to perform a cryptographic analysis of the 4-Way Handshake protocol of IEEE 802.11i is [33]; secure channels are not considered. However, this work is quite preliminary: The security assumptions and theorems are not formulated precisely and no security proofs or proof sketches are available. In He et al. [18], the first *symbolic* analysis of IEEE 802.11i has been presented, based on their Protocol Composition Logic (PCL). There are only a few other papers on the analysis of real-world protocols that involve key derivation: The Internet Key-Exchange (IKE) protocol (which is part of IPsec) was analyzed in [11]. (Fragments of) TLS were analyzed in [16,28,5], assuming session identifiers in ciphertexts [16] or the random oracle for key derivation [28,5]. Cryptographic analysis of Kerberos was carried out for example in [7], where key derivation is modeled by pseudo-random functions within CryptoVerif. However, this analysis considers more abstract message formats and does not yield composable security guarantees.

References

1. Backes, M., Dürmuth, M., Küsters, R.: On Simulatability Soundness and Mapping Soundness of Symbolic Cryptography. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 108–120. Springer, Heidelberg (2007)
2. Backes, M., Pfitzmann, B.: Symmetric Encryption in a Simulatable Dolev-Yao Style Cryptographic Library. In: CSFW-17 2004, pp. 204–218. IEEE Computer Society, Los Alamitos (2004)
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
4. Bellare, M., Rogaway, P.: Provably Secure Session Key Distribution: The Three Party Case. In: STOC 1995, pp. 57–66. ACM, New York (1995)
5. Bhargavan, K., Fournet, C., Corin, R., Zalinescu, E.: Cryptographically Verified Implementations for TLS. In: CCS 2008, pp. 459–468. ACM, New York (2008)
6. Black, J., Rogaway, P., Shrimpton, T.: Encryption-Scheme Security in the Presence of Key-Dependent Messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)
7. Blanchet, B., Jaggard, A.D., Scedrov, A., Tsay, J.-K.: Computationally Sound Mechanized Proofs for Basic and Public-key Kerberos. In: ASIACCS 2008, pp. 87–99. ACM, New York (2008)
8. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS 2001, pp. 136–145. IEEE Computer Society, Los Alamitos (2001)
9. Canetti, R.: Universally Composable Signature, Certification, and Authentication. In: CSFW-17 2004, pp. 219–233. IEEE Computer Society, Los Alamitos (2004)
10. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols, Technical Report 2000/067, Cryptology ePrint Archive (December 2005), http://eprint.iacr.org/2000/067/
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)
12. Canetti, R., Krawczyk, H.: Universally Composable Notions of Key Exchange and Secure Channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
13. Canetti, R., Rabin, T.: Universal Composition with Joint State. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
14. Comon-Lundh, H., Cortier, V.: Computational soundness of observational equivalence. Technical Report INRIA Research Report RR-6508, INRIA (2008), http://www.loria.fr/~cortier/Papiers/CCS08-report.pdf
15. Cortier, V., Kremer, S., Küsters, R., Warinschi, B.: Computationally Sound Symbolic Secrecy in the Presence of Hash Functions. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 176–187. Springer, Heidelberg (2006)
16. Gajek, S., Manulis, M., Pereira, O., Sadeghi, A., 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)

17. He, C., Mitchell, J.C.: Security Analysis and Improvements for IEEE 802.11i. In: NDSS 2005, The Internet Society (2005)
18. He, C., Sundararajan, M., Datta, A., Derek, A., Mitchell, J.C.: A Modular Correctness Proof of IEEE 802.11i and TLS. In: CCS 2005, pp. 2–15. ACM, New York (2005)
19. Hofheinz, D., Unruh, D., Müller-Quade, J.: Polynomial Runtime and Composability. Technical Report 2009/023, Cryptology ePrint Archive (2009), http://eprint.iacr.org/2009/023/
20. IEEE Standard 802.11-2007. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Part 11 of IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements (June 2007)
21. Kobara, K., Shin, S., Strefler, M.: Partnership in key exchange protocols. In: ASIACCS 2009, pp. 161–170. ACM, New York (2009)
22. Küsters, R.: Simulation-Based Security with Inexhaustible Interactive Turing Machines. In: CSFW-19 2006, pp. 309–320. IEEE Computer Society, Los Alamitos (2006)
23. Küsters, R., Tuengerthal, M.: Joint State Theorems for Public-Key Encryption and Digitial Signature Functionalities with Local Computation. In: CSF 2008, pp. 270–284. IEEE Computer Society, Los Alamitos (2008)
24. Küsters, R., Tuengerthal, M.: Computational Soundness for Key Exchange Protocols with Symmetric Encryption. In: CCS 2009, pp. 91–100. ACM Press, New York (2009)
25. Küsters, R., Tuengerthal, M.: Universally Composable Symmetric Encryption. In: CSF 2009, pp. 293–307. IEEE Computer Society, Los Alamitos (2009)
26. Küsters, R., Tuengerthal, M.: Ideal Key Derivation and Encryption in Simulation-based Security. Technical Report 2010/295, Cryptology ePrint Archive (2010), http://eprint.iacr.org/2010/295/
27. Micciancio, D., Warinschi, B.: Soundness of Formal Encryption in the Presence of Active Adversaries. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 133–151. Springer, Heidelberg (2004)
28. 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)
29. Ohigashi, T., Morii, M.: A Practical Message Falsification Attack on WPA. In: JWIS 2009 (2009)
30. Pfitzmann, B., Waidner, M.: A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In: S&P 2001, pp. 184–201. IEEE Computer Society, Los Alamitos (2001)
31. Roy, A., Datta, A., Derek, A., Mitchell, J.C.: Inductive Proofs of Computational Secrecy. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 219–234. Springer, Heidelberg (2007)
32. Tews, E., Beck, M.: Practical Attacks against WEP and WPA. In: WISEC 2009, pp. 79–86. ACM, New York (2009)
33. Zhang, F., Ma, J., Moon, S.: The Security Proof of a 4-Way Handshake Protocol in IEEE 802.11i. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3802, pp. 488–493. Springer, Heidelberg (2005)

Beyond Provable Security Verifiable IND-CCA Security of OAEP

Gilles Barthe[1], Benjamin Grégoire[2], Yassine Lakhnech[3], and Santiago Zanella Béguelin[1]

[1] IMDEA Software
[2] INRIA Sophia Antipolis-Méditerranée
[3] Université Grenoble 1, CNRS, Verimag

Abstract. OAEP is a widely used public-key encryption scheme based on trapdoor permutations. Its security proof has been scrutinized and amended repeatedly. Fifteen years after the introduction of OAEP, we present a machine-checked proof of its security against adaptive chosen-ciphertext attacks under the assumption that the underlying permutation is partial-domain one-way. The proof can be independently verified by running a small and trustworthy proof checker and fixes minor glitches that have subsisted in published proofs. We provide an overview of the proof, highlight the differences with earlier works, and explain in some detail a crucial step in the reduction: the elimination of indirect queries made by the adversary to random oracles via the decryption oracle. We also provide—within the limits of a conference paper—a broader perspective on independently verifiable security proofs.

1 Introduction

Optimal Asymmetric Encryption Padding (OAEP) [9] is a prominent public-key encryption scheme based on trapdoor permutations, most commonly used in combination with the RSA [29] and Rabin [28] functions. OAEP is widely deployed; many variants of OAEP are recommended by several standards, including IEEE P1363, PKCS, ISO 18033-2, ANSI X9, CRYPTREC and SET. Yet, the history of OAEP security is fraught with difficulties. The original 1994 paper of Bellare and Rogaway [9] proves that, under the hypothesis that the underlying trapdoor permutation family is one-way, OAEP is semantically secure under chosen-ciphertext attacks. Shoup [30] subsequently discovered in 2000 that this proof only established the security of OAEP against non-adaptive chosen-ciphertext attacks, and not (as was believed at that time) against the stronger version of IND-CCA that allows the adversary to adaptively obtain the decryption of ciphertexts of its choice. Shoup suggested a modified scheme, OAEP+, secure against adaptive attacks under the one-wayness of the underlying permutation, and gave a proof of the adaptive IND-CCA security of the original scheme when it is used in combination with RSA with public exponent $e = 3$. Simultaneously, Fujisaki, Okamoto, Pointcheval and Stern [15] proved that OAEP in its original formulation is indeed secure against adaptive attacks, but under the assumption that the underlying permutation family is partial-domain one-way. Since for the particular case of RSA this latter assumption is no stronger than (full-domain) one-wayness, this finally established the adaptive IND-CCA security of RSA-OAEP. In 2004, Pointcheval [27] gave a different proof of the same result; this new proof fills several gaps in the reduction of [15], which results in a weaker bound than originally stated. Nonetheless, the inaccurate bound of [15] remains the reference bound used in practical analyses of OAEP, see e.g. [13]. Finally, Bellare, Hofheinz and Kiltz [8], recently pointed out some ambiguities in the definition of IND-CCA, leading to four possible formulations (all of them used in the literature), and question which definition is used in the statements and proofs of OAEP.

This paper reports on a machine-checked proof that OAEP is IND-CCA secure against adaptive attacks. For the sake of definitional clarity, we identify IND-CCA with the strongest definition in the taxonomy of [8], IND-CCA-SE. Let us first give a formal definition of OAEP:

Definition 1 (OAEP encryption scheme). *Let $(\mathcal{K}_f, f, f^{-1})$ be a family of trapdoor permutations on $\{0,1\}^k$, and*

$$G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0} \qquad H : \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$$

two hash functions, with $k = n + k_0 + k_1$. The Optimal Asymmetric Encryption Padding (OAEP) scheme is composed of the following triple of algorithms:

$$\begin{array}{lll}
\mathcal{K}(\eta) & \stackrel{def}{=} & (pk, sk) \leftarrow \mathcal{K}_f(\eta);\ \textit{return}\ (pk, sk) \\
\mathcal{E}(pk, m) & \stackrel{def}{=} & r \xleftarrow{\$} \{0,1\}^{k_0};\ s \leftarrow G(r) \oplus (m \,\|\, 0^{k_1});\ t \leftarrow H(s) \oplus r; \\
 & & \textit{return}\ f(pk, s \,\|\, t) \\
\mathcal{D}(sk, c) & \stackrel{def}{=} & (s \,\|\, t) \leftarrow f^{-1}(sk, c);\ r \leftarrow t \oplus H(s);\ m \leftarrow s \oplus G(r); \\
 & & \text{if } [m]_{k_1} = 0^{k_1} \text{ then } \textit{return}\ [m]^n \text{ else } \textit{return}\ \bot
\end{array}$$

where $[x]_n$ (resp. $[x]^n$) denotes the n least (resp. most) significant bits of x.

Our main result is:

Theorem 1 (IND-CCA security of OAEP). *Let $\mathcal{A}$ be an adversary against the adaptive IND-CCA security of OAEP that makes at most q_G and q_H queries to the hash oracles G and H, respectively, and at most $q_{\mathcal{D}}$ queries to the decryption oracle $\mathcal{D}$. Suppose this adversary achieves an IND-CCA advantage ϵ within time t. Then, there exists an inverter $\mathcal{I}$ that finds a partial preimage (the most significant $k - k_0$ bits) of an element uniformly drawn from the domain of the underlying permutation f with probability ϵ' within time t', where*

$$\epsilon' \geq \frac{1}{q_H}\left(\frac{\epsilon}{2} - \frac{3q_{\mathcal{D}}q_G + q_{\mathcal{D}}^2 + 4q_{\mathcal{D}} + q_G}{2^{k_0}} - \frac{2q_{\mathcal{D}}}{2^{k_1}}\right)$$
$$t' \leq t + q_{\mathcal{D}}\ q_G\ q_H\ (T_f + O(1))$$

and where T_f is an upper bound on the time needed to compute the image of a bitstring under f. Moreover, if the underlying permutation family is partial-domain one-way and adversary $\mathcal{A}$ runs in probabilistic polynomial-time (on some *security parameter η), then the advantage of $\mathcal{A}$ is negligible, provided parameters k_0, k_1 are at least linear on η.*

The formal statement is given in Fig. 1. The proof is built using CertiCrypt [6], a general framework for building game-based cryptographic proofs in the Coq proof assistant [32], and yields an independently verifiable certificate. Said otherwise, an external verifier can examine the statement to convince herself that it faithfully captures the definitions of OAEP and IND-CCA security and can delegate the verification of the proof to an automated checker. Our exact security bound unveils minor glitches in the proof of [27], and marginally improves on its exact security bound by performing an aggressive analysis of oracle queries earlier in the sequence of games. Beyond its individual merits, the proof is highly emblematic and provides tangible evidence of the onset of tools to build and verify cryptographic proofs.

2 A Primer on Formal Proofs

Proof assistants are programs designed to support interactive construction and automatic verification of mathematical statements (understood in a broad sense). Initially developed by logicians to experiment with the expressive power of their foundational formalisms, proof assistants are now emerging as a mature technology that can be used effectively for verifying intricate mathematical proofs, such as the Four Color theorem [16] or the Kepler conjecture [18,19], or complex software systems, such as operating systems [21], virtual machines [22] and optimizing compilers [24]. In the realm of cryptography, proof assistants have been used to formally verify secrecy and authenticity properties of protocols [26].

Proof assistants rely on expressive specification languages that allow formalizing arbitrary mathematical notions, and that provide a formal representation of proofs as proof objects. Their architecture is organized into two layers: a kernel, and a proof engine.

- The kernel is the cornerstone for correctness. Its central component is a checker for verifying the consistency of formal theories, including definitions and proofs. In particular, the checker guarantees that definitions and proofs are well-typed, that there are no missing cases or undefined notions in definitions, and that all proofs are built from valid elementary logical steps and make a correct use of assumptions.
- In contrast, the proof engine helps proof construction. The proof engine embraces a variety of tools. The primary tools are a set of pre-defined tactics, and a language for writing user-defined tactics. Tactics allow to reduce a proof goal to simpler ones. When invoked on a proof goal A, a tactic will compute a new set of goals $A_1 \dots A_n$, and a proof that $A_1 \wedge \dots \wedge A_n \implies A$. At the end of each demonstration, the proof engine outputs a proof object.

Proof objects are independently checked by the kernel. Therefore, the proof engine need not be trusted, and the validity of a formal proof—beyond the accuracy of the statement itself—only depends on the correctness of the kernel.

Pleasingly, kernels are extremely reliable programs with restricted functionalities and solid logical foundations.

As with any other mathematical activity, formal proofs strive for elegance and conciseness. In our experience, they also provide a natural setting for improving proofs—in the case of cryptography, improvement can be measured by comparing exact security bounds. Yet, what matters most about a formal proof is that it provides a nearly absolute degree of assurance, without requiring expensive human verification.

3 The Statement

The formal statement of the exact IND-CCA security of OAEP is displayed in Figure 1; it comprises the definition of the IND-CCA game and the simulation that reduces security to the partial-domain one-wayness of the trapdoor permutation. The security result is expressed as a lower bound on the success probability of the reduction in terms of the success probability of an IND-CCA adversary. Both probabilities are captured formally by expressions of the form $\Pr[\mathsf{G} : E]$, where G is a game and E an event. The definition of probabilities is taken from Audebaud and Paulin's library [2], whereas the definition of games and events is taken from the CertiCrypt framework [6]. In essence, games are probabilistic programs with calls to adversaries; formally, a game is given by a main command and an environment that provides the code of algorithms and oracles—in contrast, adversaries are formalized as procedures with unknown code. Games have a probabilistic semantics: given an interpretation of adversaries as probabilistic programs, a game G is interpreted as a function $[\![\mathsf{G}]\!]$ from initial states to distributions of final states. The semantics of games is taken from [6]. Events are merely predicates over final states, and $\Pr[\mathsf{G} : E]$ is simply the probability of E in the distribution induced by $[\![\mathsf{G}]\!]$ starting from an empty initial state.

The IND-CCA game involves an adversary $\mathcal{A}$ (modeled by procedures $\mathcal{A}_1$ and $\mathcal{A}_2$), defines algorithms $\mathcal{K}$ for key generation and $\mathcal{E}$ for encryption, and gives the adversary access to a decryption oracle $\mathcal{D}$ and to random oracles G and H. We follow the convention of typesetting global variables in boldface. The first line of the main command initializes oracle memories; the lists $\boldsymbol{L}_G$ and $\boldsymbol{L}_H$ are used to simulate the random oracles G and H, whereas the list $\boldsymbol{L}_\mathcal{D}$ is a ghost variable used to track decryption queries and exclude invalid adversaries that query the decryption oracle with the challenge ciphertext during the second phase of the game. The remainder of the game is standard; note that we set a flag $\hat{\boldsymbol{c}}_{\mathsf{def}}$ just before giving the challenge ciphertext to the adversary in order to distinguish decryption queries made in the second phase of the game from those made in the first phase. The code of the decryption oracle and the encryption and key generation algorithms is a direct transcription of the informal definitions and is omitted.

The code of the game is complemented by a variable policy that declares which variables are accessible to adversaries: $\mathcal{A}$ cannot read nor modify the values of $\boldsymbol{sk}$, $\boldsymbol{L}_\mathcal{D}$, $\boldsymbol{L}_G$, $\boldsymbol{L}_H$, $\hat{\boldsymbol{c}}_{\mathsf{def}}$, and $\hat{\boldsymbol{c}}$, and cannot modify the value of $\boldsymbol{pk}$; on the other hand, the procedures representing the two phases of the adversary can communicate

```
Game $\mathsf{G}_{\mathsf{IND\text{-}CCA}}$ :
  $\boldsymbol{L}_G, \boldsymbol{L}_H, \boldsymbol{L}_{\mathcal{D}} \leftarrow \mathsf{nil}$;
  $(\boldsymbol{pk}, \boldsymbol{sk}) \leftarrow \mathcal{K}(\eta)$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(\boldsymbol{pk})$;
  $b \overset{\$}{\leftarrow} \{0,1\}$;
  $\hat{\boldsymbol{c}} \leftarrow \mathcal{E}(m_b)$;
  $\hat{\boldsymbol{c}}_{\mathsf{def}} \leftarrow \mathsf{true}$;
  $\overline{b} \leftarrow \mathcal{A}_2(\boldsymbol{pk}, \hat{\boldsymbol{c}})$

Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(\boldsymbol{L}_G)$ then
    $g \overset{\$}{\leftarrow} \{0,1\}^{n+k_1}$;
    $\boldsymbol{L}_G[r] \leftarrow g$
  else $g \leftarrow \boldsymbol{L}_G[r]$
  return $g$

Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(\boldsymbol{L}_H)$ then
    $h \overset{\$}{\leftarrow} \{0,1\}^{k_0}$;
    $\boldsymbol{L}_H[s] \leftarrow h$
  else $h \leftarrow \boldsymbol{L}_H[s]$
  return $h$

Oracle $\mathcal{D}(c)$ :
  $\boldsymbol{L}_{\mathcal{D}} \leftarrow (\hat{\boldsymbol{c}}_{\mathsf{def}}, c) :: \boldsymbol{L}_{\mathcal{D}}$;
  $(s, t) \leftarrow f^{-1}(\boldsymbol{sk}, c)$;
  $h \leftarrow H(s)$;
  $r \leftarrow t \oplus h$;
  $g \leftarrow G(r)$;
  if $[s \oplus g]_{k_1} = 0^{k_1}$ then
    return $[s \oplus g]^n$
  else return $\perp$
```

```
Game $\mathsf{G}_{\mathsf{set\text{-}PD\text{-}OW}}$ :
  $(pk, sk) \leftarrow \mathcal{K}_f(\eta)$;
  $s \overset{\$}{\leftarrow} \{0,1\}^{n+k_1}$;
  $t \overset{\$}{\leftarrow} \{0,1\}^{k_0}$;
  $S \leftarrow \mathcal{I}(pk, f(pk, s \,\|\, t))$

Adversary $\mathcal{I}(pk, y)$ :
  $\boldsymbol{L}_G, \boldsymbol{L}_H \leftarrow \mathsf{nil}$;
  $\boldsymbol{pk} \leftarrow pk$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(\boldsymbol{pk})$;
  $\hat{\boldsymbol{c}} \leftarrow y$;
  $\hat{\boldsymbol{c}}_{\mathsf{def}} \leftarrow \mathsf{true}$;
  $\overline{b} \leftarrow \mathcal{A}_2(\boldsymbol{pk}, \hat{\boldsymbol{c}})$;
  return $\mathsf{dom}(\boldsymbol{L}_H)$

Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(\boldsymbol{L}_G)$ then
    $g \overset{\$}{\leftarrow} \{0,1\}^{n+k_1}$;
    $\boldsymbol{L}_G[r] \leftarrow g$
  else $g \leftarrow \boldsymbol{L}_G[r]$
  return $g$

Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(\boldsymbol{L}_H)$ then
    $h \overset{\$}{\leftarrow} \{0,1\}^{k_0}$;
    $\boldsymbol{L}_H[s] \leftarrow h$
  else $h \leftarrow \boldsymbol{L}_H[s]$
  return $h$

Oracle $\mathcal{D}(c)$ :
  if $\exists (s, h) \in \boldsymbol{L}_H, (r, g) \in \boldsymbol{L}_G$.
    $c = f(\boldsymbol{pk}, s \,\|\, (r \oplus h)) \wedge$
    $[s \oplus g]_{k_1} = 0^{k_1}$
  then return $[s \oplus g]^n$
  else return $\perp$
```

$$\mathsf{WF}(\mathcal{A}) \wedge \Pr[\mathsf{G}_{\mathsf{IND\text{-}CCA}} : |\boldsymbol{L}_G| \leq q_G + q_{\mathcal{D}} + 1 \wedge |\boldsymbol{L}_{\mathcal{D}}| \leq q_{\mathcal{D}} \wedge (\mathsf{true}, \hat{\boldsymbol{c}}) \notin \boldsymbol{L}_{\mathcal{D}}] = 1 \implies$$
$$\Pr\left[\mathsf{G}_{\mathsf{IND\text{-}CCA}} : \overline{b} = b\right] - \frac{1}{2} \leq \Pr[\mathsf{G}_{\mathsf{set\text{-}PD\text{-}OW}} : s \in S] + \frac{3q_{\mathcal{D}}q_G + q_{\mathcal{D}}^2 + 4q_{\mathcal{D}} + q_G}{2^{k_0}} + \frac{2q_{\mathcal{D}}}{2^{k_1}}$$

Fig. 1. Formal statement of IND-CCA security of OAEP

through shared variables. An adversary $\mathcal{A}$ respecting the variable policy is said to be well-formed; this is noted as $\mathsf{WF}(\mathcal{A})$.

The security statement itself takes the form of an implication, whose premise fixes the class of adversaries considered. The statement considers well-formed adversaries that make at most $q_{\mathcal{D}}$ and q_G queries to the decryption and G oracles respectively[1], and that do not query the decryption oracle with the challenge ciphertext in the second phase of the game. Given an IND-CCA adversary $\mathcal{A}$, we show how to construct an inverter $\mathcal{I}$ that uses $\mathcal{A}$ as a subroutine to partially invert the underlying trapdoor permutation. The success probability of the inverter is given by $\Pr[\mathsf{G}_{\mathsf{set\text{-}PD\text{-}OW}} : s \in S]$, and is lower bounded by:

$$\frac{1}{2}\,\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CCA}} - \frac{3q_{\mathcal{D}}q_G + q_{\mathcal{D}}^2 + 4q_{\mathcal{D}} + q_G}{2^{k_0}} - \frac{2q_{\mathcal{D}}}{2^{k_1}}$$

where the IND-CCA advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CCA}}$ of $\mathcal{A}$ is defined as usual as

$$2\,\Pr\left[\mathsf{G}_{\mathsf{IND\text{-}CCA}} : \overline{b} = b\right] - 1$$

[1] The formal statement slightly relaxes this condition; it requires the length of $\boldsymbol{L}_G$ be at most $q_G + q_{\mathcal{D}} + 1$ (the 1 accounting for the call to G needed to compute the challenge ciphertext), so that the adversary could trade calls to $\mathcal{D}$ for calls to G.

One additional remark is needed to relate the formal statement to the statement of Theorem [1]. Strictly, the formal statement reduces the security of OAEP not to the partial-domain one-wayness of the permutation, but to its *set* partial-domain one-wayness. Both notions are closely related (cf. [15]). We could have formally proven the reduction to the former problem using basically the same argument, but making the inverter return a value uniformly chosen from the domain of $\boldsymbol{L}_H$ instead; this accounts for the multiplicative factor q_H^{-1} in Theorem [1]. The reduction from partial-domain one-wayness to set partial-domain one-wayness is inessential to the presentation and can be proven independently and generically for any inverter $\mathcal{I}$.

4 The Proof

One claimed virtue of verifiable security is that there is no need to understand its proof (only its statement) to trust the correctness of a result. Obviously, it remains of interest to understand the thrust of the proof, and if one intends to reproduce the proof—perhaps in a slightly different setting, or for a different scheme, or with a different framework—its ultimate details. This section provides an overview of the techniques used to conduct the proof and delves into the details of one significant proof step, namely eliminating fresh oracle calls to G in the decryption oracle. The code of the proof and all the infrastructure needed to independently verify may be obtained from the authors upon simple request.

Tools. The proof makes an extensive use of the techniques provided by the CertiCrypt framework, as reported in [6], and the additional techniques described in [7]. The unifying formalism used by CertiCrypt to justify transitions between games is a Relational Hoare Logic, whose judgments are of the form $\vdash \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi$, relating two games G_1 and G_2 w.r.t. two relations Ψ and Φ on states. Such a judgment means that for any initial memories m_1 and m_2 satisfying the precondition $m_1\ \Psi\ m_2$, the distributions $[\![\mathsf{G}_1]\!]\ m_1$ and $[\![\mathsf{G}_2]\!]\ m_2$ are related by the lifting of Φ to distributions[2]. Relational Hoare Logic subsumes observational equivalence $\vdash \mathsf{G}_1 \sim_Y^X \mathsf{G}_2$, which is obtained by setting Ψ and Φ to $=_X$ and $=_Y$, where X (resp. Y) is a set of variables and $=_X$ (resp. $=_Y$) relates memories that coincide on all variables in X (resp. Y).

Both Relational Hoare Logic and observational equivalence statements allow to express that two games perfectly simulate each other. Proofs can be conducted using proof rules à la Hoare Logic—i.e., there is a rule for each construction of the programming language and structural rules—or certified tactics that automate program transformations such as dead code elimination, constant folding and propagation, or procedure call inlining.

[2] In the general case, we adopt the definition of lifting from probabilistic process algebra, which is formulated in terms of a max-flow min-cut problem and involves an existential quantification over distributions. For partial equivalence relations, the definition coincides with the usual approach that requires the probability of equivalence classes be the same.

We use the logic of *swapping statements* of [7] to prove independence of values from adversary's view. We say that a value is independent from adversary's view at some point in a game if it can be resampled without modifying the meaning of the game. The logic for swapping statements deals with Relational Hoare judgments of the form $\vdash S; \mathsf{G}_1 \sim_Y^X \mathsf{G}_2; S$, where the games $S; \mathsf{G}_1$ and $\mathsf{G}_2; S$ are respectively obtained from games G_1 and G_2 by prefixing and postfixing some code fragment S. Typically, S just resamples part of the state of the game; moreover, the code of oracles in G_1 and G_2 may also differ in the random samplings they perform. In general, the logic of swapping statements can be used to justify eager and lazy sampling transformations—overcoming limitations in [6]. An example of its application is given below.

In addition to Relational Hoare Logic, CertiCrypt formalizes the Fundamental Lemma of Game-Playing [20,10,31], which is used to justify "lossy" steps where two consecutive games in a proof structured as a sequence of games only diverge when a failure event occurs. The Failure Event Lemma of [7] complements the Fundamental Lemma of Game-Playing and allows to bound the probability of a failure event triggered inside an oracle by a function of the number of calls made to the oracle. There exist several specialized versions of this lemma; the simplest instance focuses on games in which the failure event F is triggered by an oracle $\mathcal{O}$ with a probability bounded by a constant ϵ, independent from the argument with which it is called and of any previous calls. In this case, the Failure Event Lemma bounds the probability of event F by $q_{\mathcal{O}}\ \epsilon$, where $q_{\mathcal{O}}$ is a bound on the number of calls to $\mathcal{O}$. While this instance of the Failure Event Lemma suffices to justify most lossy transformations in the proof of OAEP, we also needed to resort to the full generality of the lemma on two occasions; one of them is outlined below.

Proof outline. Figure 2 outlines the structure of the proof; the first step from $\mathsf{G}_{\mathsf{IND\text{-}CCA}}$ to G_1 and the final step from G_5 to $\mathsf{G}_{\mathsf{set\text{-}PD\text{-}OW}}$ are not displayed. The reduction successively eliminates all situations in which the plaintext extractor used by the inverter to simulate decryption may fail.

Starting from game $\mathsf{G}_{\mathsf{IND\text{-}CCA}}$, we use the logic of swapping statements to fix the hash $\hat{\boldsymbol{g}}$ that G gives in response to the random seed in the challenge ciphertext; the computation of the challenge ciphertext unfolds to:

$$\hat{\boldsymbol{r}} \xleftarrow{\$} \{0,1\}^{k_0};\ \hat{\boldsymbol{s}} \leftarrow \hat{\boldsymbol{g}} \oplus (m_b \,\|\, 0^{k_1});\ \hat{\boldsymbol{h}} \leftarrow H(\hat{\boldsymbol{s}});\ \hat{\boldsymbol{t}} \leftarrow \hat{\boldsymbol{h}} \oplus \hat{\boldsymbol{r}};\ \hat{\boldsymbol{c}} \leftarrow f(\boldsymbol{pk}, \hat{\boldsymbol{s}} \,\|\, \hat{\boldsymbol{t}})$$

where $\hat{\boldsymbol{g}}$ is sampled from $\{0,1\}^{k-k_0}$ before the first call to $\mathcal{A}$. We then make G respond to an adversary query $\hat{\boldsymbol{r}}$ with a freshly sampled value instead of $\hat{\boldsymbol{g}}$; this only makes a difference if flag **bad** is set in game G_1. Since at this point $\hat{\boldsymbol{g}}$ is uniformly distributed and independent from the adversary's view, the value $\hat{\boldsymbol{s}}$ computed as $\hat{\boldsymbol{g}} \oplus (m_b \,\|\, 0^{k_1})$ is as well uniformly distributed and independent from the adversary's view. This removes the dependence of the adversary output on the hidden bit b, and thus the probability of a correct guess is exactly $1/2$. Using the Fundamental Lemma we obtain the bound:

$$\Pr\left[\mathsf{G}_{\mathsf{IND\text{-}CCA}} : \bar{b} = b\right] - \Pr\left[\mathsf{G}_1 : \bar{b} = b\right] = \Pr\left[\mathsf{G}_{\mathsf{IND\text{-}CCA}} : \bar{b} = b\right] - \frac{1}{2} \qquad (1)$$

$$\leq \Pr[\mathsf{G}_1 : \mathbf{bad}] \qquad (2)$$

```
Game $\mathsf{G}_1$ :
  $\boldsymbol{L}_G, \boldsymbol{L}_H, \boldsymbol{L}_\mathcal{D} \leftarrow$ nil;
  $(\boldsymbol{pk}, \boldsymbol{sk}) \leftarrow \mathcal{K}_f()$;
  $\hat{\boldsymbol{r}} \xleftarrow{\$} \{0,1\}^{k_0}$;
  $\hat{\boldsymbol{s}} \xleftarrow{\$} \{0,1\}^{k-k_0}$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(\boldsymbol{pk})$;
  $b \xleftarrow{\$} \{0,1\}$;
  $\hat{\boldsymbol{h}} \leftarrow H(\hat{\boldsymbol{s}})$;
  $\hat{\boldsymbol{t}} \leftarrow \hat{\boldsymbol{h}} \oplus \hat{\boldsymbol{r}}$;
  $\hat{\boldsymbol{c}} \leftarrow f(\boldsymbol{pk}, \hat{\boldsymbol{s}} \,\|\, \hat{\boldsymbol{t}})$;
  $\hat{\boldsymbol{c}}_{\mathsf{def}} \leftarrow$ true;
  $\bar{b} \leftarrow \mathcal{A}_2(\boldsymbol{pk}, \hat{\boldsymbol{c}})$

Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(\boldsymbol{L}_G)$ then
    if $r = \hat{\boldsymbol{r}}$ then
      bad $\leftarrow$ true;
    $g \xleftarrow{\$} \{0,1\}^{k-k_0}$;
    $\boldsymbol{L}_G[r] \leftarrow g$
  else $g \leftarrow \boldsymbol{L}_G[r]$
  return $g$

Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(\boldsymbol{L}_H)$ then
    $h \xleftarrow{\$} \{0,1\}^{k_0}$;
    $\boldsymbol{L}_H[s] \leftarrow h$
  else $h \leftarrow \boldsymbol{L}_H[s]$
  return $h$

Oracle $\mathcal{D}(c)$ :
  if $(\hat{\boldsymbol{c}}_{\mathsf{def}} \wedge \hat{\boldsymbol{c}} = c) \vee q_\mathcal{D} < |\boldsymbol{L}_\mathcal{D}| \vee q_\mathcal{D} + q_G < |\boldsymbol{L}_G|$
  then return $\perp$
  else
    $\boldsymbol{L}_\mathcal{D} \leftarrow (\hat{\boldsymbol{c}}_{\mathsf{def}}, c) :: \boldsymbol{L}_\mathcal{D}$;
    $(s, t) \leftarrow f^{-1}(\boldsymbol{sk}, c)$;
    $r \leftarrow t \oplus H(s)$;
    $g \leftarrow G(r)$;
    if $[s \oplus g]_{k_1} = 0^{k_1}$ then return $[s \oplus g]^n$ else return $\perp$
```

$\Pr[\mathsf{G}_1 : \mathbf{bad}] \leq \Pr[\mathsf{G}_2 : \mathbf{bad}] + \dfrac{q_\mathcal{D}^2 + q_\mathcal{D} q_G + q_\mathcal{D}}{2^{k_0}} + \dfrac{q_\mathcal{D}}{2^{k_1}}$

Inline G and case analysis on whether $s \in \mathsf{dom}(\boldsymbol{L}_H)$ in $\mathcal{D}$.
Reject ciphertexts with a fresh g or h

```
Game $\mathsf{G}_2$ :
  $\boldsymbol{L}_G, \boldsymbol{L}_H, \boldsymbol{L}_\mathcal{D} \leftarrow$ nil;
  $(\boldsymbol{pk}, \boldsymbol{sk}) \leftarrow \mathcal{K}_f()$;
  $\hat{\boldsymbol{r}} \xleftarrow{\$} \{0,1\}^{k_0}$;
  $\hat{\boldsymbol{s}} \xleftarrow{\$} \{0,1\}^{k-k_0}$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(\boldsymbol{pk})$;
  $b \xleftarrow{\$} \{0,1\}$;
  $\hat{\boldsymbol{h}} \leftarrow H(\hat{\boldsymbol{s}})$;
  $\hat{\boldsymbol{t}} \leftarrow \hat{\boldsymbol{h}} \oplus \hat{\boldsymbol{r}}$;
  $\hat{\boldsymbol{c}} \leftarrow f(\boldsymbol{pk}, \hat{\boldsymbol{s}} \,\|\, \hat{\boldsymbol{t}})$;
  $\hat{\boldsymbol{c}}_{\mathsf{def}} \leftarrow$ true;
  $\bar{b} \leftarrow \mathcal{A}_2(\boldsymbol{pk}, \hat{\boldsymbol{c}})$

Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(\boldsymbol{L}_G)$ then
    if $r = \hat{\boldsymbol{r}}$ then
      bad $\leftarrow$ true;
    $g \xleftarrow{\$} \{0,1\}^{k-k_0}$;
    $\boldsymbol{L}_G[r] \leftarrow g$
  else $g \leftarrow \boldsymbol{L}_G[r]$
  return $g$

Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(\boldsymbol{L}_H)$ then
    $h \xleftarrow{\$} \{0,1\}^{k_0}$;
    $\boldsymbol{L}_H[s] \leftarrow h$
  else $h \leftarrow \boldsymbol{L}_H[s]$
  return $h$

Oracle $\mathcal{D}(c)$ :
  if $(\hat{\boldsymbol{c}}_{\mathsf{def}} \wedge \hat{\boldsymbol{c}} = c) \vee q_\mathcal{D} < |\boldsymbol{L}_\mathcal{D}| \vee q_\mathcal{D} + q_G < |\boldsymbol{L}_G|$
  then return $\perp$
  else
    $\boldsymbol{L}_\mathcal{D} \leftarrow (\hat{\boldsymbol{c}}_{\mathsf{def}}, c) :: \boldsymbol{L}_\mathcal{D}$;
    $(s, t) \leftarrow f^{-1}(\boldsymbol{sk}, c)$;
    if $s \in \mathsf{dom}(\boldsymbol{L}_H)$ then
      $r \leftarrow t \oplus H(s)$;
      if $r \in \mathsf{dom}(\boldsymbol{L}_G)$ then
        $g \leftarrow \boldsymbol{L}_G[r]$;
        if $[s \oplus g]_{k_1} = 0^{k_1}$ then return $[s \oplus g]^n$
        else return $\perp$
      else
        if $r = \hat{\boldsymbol{r}}$ then bad $\leftarrow$ true;
        $g \xleftarrow{\$} \{0,1\}^{k-k_0}$; $\boldsymbol{L}_G[r] \leftarrow g$; return $\perp$
    else
      $r \leftarrow t \oplus H(s)$;
      if $r \notin \mathsf{dom}(\boldsymbol{L}_G)$ then
        $g \xleftarrow{\$} \{0,1\}^{k-k_0}$; $\boldsymbol{L}_G[r] \leftarrow g$
      return $\perp$
```

$\Pr[\mathsf{G}_2 : \mathbf{bad}] \leq \Pr[\mathsf{G}_3 : \mathbf{bad}] + \dfrac{q_\mathcal{D}}{2^{k_1}}$

Eliminate assignments to $\boldsymbol{L}_G$ in $\mathcal{D}$
Update $\mathcal{D}$ to enforce new bound on $\boldsymbol{L}_G$

```
Game $\mathsf{G}_3$ :
  $\boldsymbol{L}_G, \boldsymbol{L}_H, \boldsymbol{L}_\mathcal{D} \leftarrow$ nil;
  $(\boldsymbol{pk}, \boldsymbol{sk}) \leftarrow \mathcal{K}_f()$;
  $\hat{\boldsymbol{r}} \xleftarrow{\$} \{0,1\}^{k_0}$;
  $\hat{\boldsymbol{s}} \xleftarrow{\$} \{0,1\}^{k-k_0}$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(\boldsymbol{pk})$;
  $b \xleftarrow{\$} \{0,1\}$;
  $\hat{\boldsymbol{h}} \leftarrow H(\hat{\boldsymbol{s}})$;
  $\hat{\boldsymbol{t}} \leftarrow \hat{\boldsymbol{h}} \oplus \hat{\boldsymbol{r}}$;
  $\hat{\boldsymbol{c}} \leftarrow f(\boldsymbol{pk}, \hat{\boldsymbol{s}} \,\|\, \hat{\boldsymbol{t}})$;
  $\hat{\boldsymbol{c}}_{\mathsf{def}} \leftarrow$ true;
  $\bar{b} \leftarrow \mathcal{A}_2(\boldsymbol{pk}, \hat{\boldsymbol{c}})$

Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(\boldsymbol{L}_G)$ then
    if $r = \hat{\boldsymbol{r}}$ then
      bad $\leftarrow$ true;
    $g \xleftarrow{\$} \{0,1\}^{k-k_0}$;
    $\boldsymbol{L}_G[r] \leftarrow g$
  else $g \leftarrow \boldsymbol{L}_G[r]$
  return $g$

Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(\boldsymbol{L}_H)$ then
    $h \xleftarrow{\$} \{0,1\}^{k_0}$;
    $\boldsymbol{L}_H[s] \leftarrow h$
  else $h \leftarrow \boldsymbol{L}_H[s]$
  return $h$

Oracle $\mathcal{D}(c)$ :
  if $(\hat{\boldsymbol{c}}_{\mathsf{def}} \wedge \hat{\boldsymbol{c}} = c) \vee q_\mathcal{D} < |\boldsymbol{L}_\mathcal{D}| \vee q_G < |\boldsymbol{L}_G|$
  then return $\perp$
  else
    $\boldsymbol{L}_\mathcal{D} \leftarrow (\hat{\boldsymbol{c}}_{\mathsf{def}}, c) :: \boldsymbol{L}_\mathcal{D}$;
    $(s, t) \leftarrow f^{-1}(\boldsymbol{sk}, c)$;
    if $s \in \mathsf{dom}(\boldsymbol{L}_H)$ then
      $r \leftarrow t \oplus H(s)$;
      if $r \in \mathsf{dom}(\boldsymbol{L}_G)$ then
        $g \leftarrow \boldsymbol{L}_G[r]$;
        if $[s \oplus g]_{k_1} = 0^{k_1}$ then return $[s \oplus g]^n$
        else return $\perp$
      else
        if $r = \hat{\boldsymbol{r}}$ then bad $\leftarrow$ true;
        return $\perp$
    else
      $r \leftarrow t \oplus H(s)$; return $\perp$
```

Fig. 2. Outline of the reduction showing the *lossy* transitions. Fragments of code that change between games are highlighted on a gray background.

$\Pr[\mathsf{G}_3 : \mathsf{bad}] \leq \Pr[\mathsf{G}_4 : \mathsf{bad}] + \frac{q_{\mathcal{D}} q_G + q_{\mathcal{D}}}{2^{k_0}}$

Inline calls to H in $\mathcal{D}$
Eliminate assignments to $\boldsymbol{L}_H$ in $\mathcal{D}$

```
Game G4 :
  $L_G, L_H, L_D \leftarrow$ nil;
  $(pk, sk) \leftarrow \mathcal{K}_f()$;
  $\hat{r} \xleftarrow{\$} \{0,1\}^{k_0}$;
  $\hat{s} \xleftarrow{\$} \{0,1\}^{k-k_0}$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
  $b \xleftarrow{\$} \{0,1\}$;
  $\hat{h} \leftarrow H(\hat{s})$;
  $\hat{t} \leftarrow \hat{h} \oplus \hat{r}$;
  $\hat{c} \leftarrow f(pk, \hat{s} \,\|\, \hat{t})$;
  $\hat{c}_{\mathsf{def}} \leftarrow$ true;
  $\overline{b} \leftarrow \mathcal{A}_2(pk, \hat{c})$
```

```
Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(L_G)$ then
    if $r = \hat{r}$ then
      bad $\leftarrow$ true;
    $g \xleftarrow{\$} \{0,1\}^{k-k_0}$;
    $L_G[r] \leftarrow g$
  else $g \leftarrow L_G[r]$
  return $g$

Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(L_H)$ then
    $h \xleftarrow{\$} \{0,1\}^{k_0}$;
    $L_H[s] \leftarrow h$
  else $h \leftarrow L_H[s]$
  return $h$
```

```
Oracle $\mathcal{D}(c)$ :
  if $(\hat{c}_{\mathsf{def}} \wedge \hat{c} = c) \vee q_{\mathcal{D}} < |L_{\mathcal{D}}| \vee q_G < |L_G|$
  then return $\perp$
  else
    $L_{\mathcal{D}} \leftarrow (\hat{c}_{\mathsf{def}}, c) :: L_{\mathcal{D}}$;
    $(s, t) \leftarrow f^{-1}(sk, c)$;
    if $s \in \mathsf{dom}(L_H)$ then
      $h \leftarrow L_H[s]$; $r \leftarrow t \oplus h$;
      if $r \in \mathsf{dom}(L_G)$ then
        $g \leftarrow L_G[r]$;
        if $[s \oplus g]_{k_1} = 0^{k_1}$ then return $[s \oplus g]^n$
        else return $\perp$
      else
        if $r = \hat{r}$ then bad $\leftarrow$ true;
        return $\perp$
    else return $\perp$
```

$\Pr[\mathsf{G}_4 : \mathsf{bad}] \leq \Pr[\mathsf{G}_5 : \mathsf{bad}_H] + \frac{q_{\mathcal{D}} q_G + 2q_{\mathcal{D}} + q_G}{2^{k_0}}$

Eagerly sample the value of $\hat{\boldsymbol{h}}$
Introduce bad_H in H
Bound bad in terms of bad_H

```
Game G5 :
  $L_G, L_H, L_D \leftarrow$ nil;
  $(pk, sk) \leftarrow \mathcal{K}_f()$;
  $\hat{r} \xleftarrow{\$} \{0,1\}^{k_0}$;
  $\hat{s} \xleftarrow{\$} \{0,1\}^{k-k_0}$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
  $b \xleftarrow{\$} \{0,1\}$;
  $\hat{h} \xleftarrow{\$} \{0,1\}^{k_0}$;
  $\hat{t} \leftarrow \hat{h} \oplus \hat{r}$;
  $\hat{c} \leftarrow f(pk, \hat{s} \,\|\, \hat{t})$;
  $\hat{c}_{\mathsf{def}} \leftarrow$ true;
  $\overline{b} \leftarrow \mathcal{A}_2(pk, \hat{c})$
```

```
Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(L_G)$ then
    if $r = \hat{r}$ then
      bad $\leftarrow$ true;
    $g \xleftarrow{\$} \{0,1\}^{k-k_0}$;
    $L_G[r] \leftarrow g$
  else $g \leftarrow L_G[r]$
  return $g$

Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(L_H)$ then
    if $s = \hat{s}$ then
      $\mathsf{bad}_H \leftarrow$ true;
    $h \xleftarrow{\$} \{0,1\}^{k_0}$;
    $L_H[s] \leftarrow h$
  else $h \leftarrow L_H[s]$
  return $h$
```

```
Oracle $\mathcal{D}(c)$ :
  if $(\hat{c}_{\mathsf{def}} \wedge \hat{c} = c) \vee q_{\mathcal{D}} < |L_{\mathcal{D}}| \vee q_G < |L_G|$
  then return $\perp$
  else
    $L_{\mathcal{D}} \leftarrow (\hat{c}_{\mathsf{def}}, c) :: L_{\mathcal{D}}$;
    $(s, t) \leftarrow f^{-1}(sk, c)$;
    if $s \in \mathsf{dom}(L_H)$ then
      $h \leftarrow L_H[s]$; $r \leftarrow t \oplus h$;
      if $r \in \mathsf{dom}(L_G)$ then
        $g \leftarrow L_G[r]$;
        if $[s \oplus g]_{k_1} = 0^{k_1}$ then return $[s \oplus g]^n$
        else return $\perp$
      else return $\perp$
    else return $\perp$
```

Fig. 2. Outline of the reduction showing the *lossy* transitions. Fragments of code that change between games are highlighted on a gray background.

The transition from G_1 to G_2 modifies the decryption oracle successively by inlining the call to G, and by applying the Fundamental and Failure Event lemmas to reject the ciphertext when there is a small chance it matches the padding. Overall, we prove:

$$\Pr[\mathsf{G}_1 : \mathsf{bad}] \leq \Pr[\mathsf{G}_2 : \mathsf{bad}] + \frac{q_{\mathcal{D}}^2 + q_{\mathcal{D}} q_G + q_{\mathcal{D}}}{2^{k_0}} + \frac{q_{\mathcal{D}}}{2^{k_1}} \tag{3}$$

Next, we eliminate fresh calls to G in the decryption oracle. These calls correspond to the two assignments $\boldsymbol{L}_G[r] \leftarrow g$, since calls to G have been inlined previously. We perform an aggressive elimination and remove both calls. As a result, in game G_3 the length of list $\boldsymbol{L}_G$ (i.e. the number of calls to G) is bounded by q_G rather than $q_{\mathcal{D}} + q_G$. This is the key to improve on the security bound of Pointcheval [27], who only removes the second call. The proof relies on the logic of swapping statements to show that values of discarded calls are "uniformly distributed and independent from the adversary's view". Details appear in next paragraph. Overall, we prove:

$$\Pr[\mathsf{G}_2 : \textbf{bad}] \leq \Pr[\mathsf{G}_3 : \textbf{bad}] + \frac{q_{\mathcal{D}}}{2^{k_1}} \tag{4}$$

Likewise, we eliminate calls to H in $\mathcal{D}$, yielding a new game G_4 in which the decryption oracle does not add any new values to the memories of G and H. Using the Fundamental and Failure Event lemmas, we obtain:

$$\Pr[\mathsf{G}_3 : \textbf{bad}] \leq \Pr[\mathsf{G}_4 : \textbf{bad}] + \frac{q_{\mathcal{D}} q_G + q_{\mathcal{D}}}{2^{k_0}} \tag{5}$$

We next fix the value $\hat{\boldsymbol{h}}$ that oracle H gives in response to $\hat{\boldsymbol{s}}$, and then make H return a freshly sampled value instead of $\hat{\boldsymbol{h}}$. This allows us to bound the probability of **bad** in terms of the probability of a newly introduced event $\textbf{bad}_H$, that indicates whether the adversary queried the value of $H(\hat{\boldsymbol{s}})$. The proof uses the hypothesis that $\mathcal{A}_2$ cannot query the decryption oracle with the challenge ciphertext, and yields:

$$\Pr[\mathsf{G}_4 : \textbf{bad}] \leq \Pr[\mathsf{G}_5 : \textbf{bad}_H] + \frac{q_{\mathcal{D}} q_G + 2q_{\mathcal{D}} + q_G}{2^{k_0}} \tag{6}$$

Finally, we prove that the probability of $\textbf{bad}_H$ in G_5 is upper bounded by the probability that the inverter $\mathcal{I}$ in Figure 1 succeeds in partially inverting the permutation f. The proof uses the (standard, non-relational) invariant on G_5:

$$\textbf{bad}_H \implies \hat{\boldsymbol{s}} \in \mathsf{dom}(\boldsymbol{L}_H)$$

The inverter $\mathcal{I}$ that we build (shown in Fig. 1) gives its own challenge y as the challenge ciphertext to the IND-CCA adversary $\mathcal{A}$. The task of the inverter is to return a list of values containing the partial preimage of its challenge which, stated in terms of the variables of game G_5, is $\hat{\boldsymbol{s}}$. Thus:

$$\Pr[\mathsf{G}_5 : \textbf{bad}_H] \leq \Pr[\mathsf{G}_5 : \hat{\boldsymbol{s}} \in \mathsf{dom}(\boldsymbol{L}_H)] = \Pr[\mathsf{G}_{\textsf{set-PD-OW}} : s \in S] \tag{7}$$

Where the last equality follows from an algebraic equivalence that we prove as a lemma:

$$\hat{\boldsymbol{h}} \overset{\$}{\leftarrow} \{0,1\}^{k_0};\ \hat{\boldsymbol{t}} \leftarrow \hat{\boldsymbol{h}} \oplus \hat{\boldsymbol{r}} \sim^{\{\hat{\boldsymbol{r}}\}}_{\{\hat{\boldsymbol{h}},\hat{\boldsymbol{t}},\hat{\boldsymbol{r}}\}} \hat{\boldsymbol{t}} \overset{\$}{\leftarrow} \{0,1\}^{k_0};\ \hat{\boldsymbol{h}} \leftarrow \hat{\boldsymbol{t}} \oplus \hat{\boldsymbol{r}}$$

Putting together Equations (1)–(7) concludes the proof of the statement in Figure 1.

Detailed proof of the transition from G_2 *to* G_3*.* We use the five intermediate games shown in Figure 3. The first transition from G_2 to G_2^1 consists in adding

```
Game $\fbox{G_2^1}$ $\boxed{G_2^2}$ :
  $L_G, L_H, L_\mathcal{D} \leftarrow$ nil;
  $(pk, sk) \leftarrow \mathcal{K}_f()$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
  $b \overset{\$}{\leftarrow} \{0,1\}$;
  $\hat{r} \overset{\$}{\leftarrow} \{0,1\}^{k_0}$;
  $\hat{s} \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;
  $\hat{h} \leftarrow H(\hat{s})$;
  $\hat{t} \leftarrow \hat{h} \oplus \hat{r}$;
  $\hat{c} \leftarrow f(pk, \hat{s} \,\|\, \hat{t})$;
  $\hat{c}_{\mathsf{def}} \leftarrow$ true;
  $\bar{b} \leftarrow \mathcal{A}_2(pk, \hat{c})$
```

```
Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(L_G)$ then
    if $r = \hat{r}$ then
      bad $\leftarrow$ true
    $g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;
    $L_G[r] \leftarrow$ (false, $g$)
  else
    $(d, g) \leftarrow L_G[r]$;
    $L_G[r] \leftarrow$ (false, $g$)
  return $g$
```

```
Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(L_H)$ then
    $h \overset{\$}{\leftarrow} \{0,1\}^{k_0}$;
    $L_H[s] \leftarrow h$
  else $h \leftarrow L_H[s]$
  return $h$
```

```
Oracle $\mathcal{D}(c)$ :
  if $(\hat{c}_{\mathsf{def}} \wedge \hat{c} = c) \vee q_\mathcal{D} < |L_\mathcal{D}| \vee q_\mathcal{D} + q_G < |L_G|$
  then return $\perp$
  else
    $L_\mathcal{D} \leftarrow (\hat{c}_{\mathsf{def}}, c) :: L_\mathcal{D}$;  $(s, t) \leftarrow f^{-1}(sk, c)$;
    if $s \in \mathsf{dom}(L_H)$ then
      $r \leftarrow t \oplus H(s)$;
      if $r \in \mathsf{dom}(L_G)$ then
        $(d, g) \leftarrow L_G[r]$;
        if $d$ = true then
          if $[s \oplus g]_{k_1} = 0^{k_1}$ then
            bad$_1$ $\leftarrow$ true;
            $\fbox{return $[s \oplus g]^n$}$ $\boxed{\text{return } \perp}$
          else return $\perp$
        else
          if $[s \oplus g]_{k_1} = 0^{k_1}$ then return $[s \oplus g]^n$
          else return $\perp$
      else
        if $r = \hat{r}$ then bad $\leftarrow$ true;
        $g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;  $L_G[r] \leftarrow$ (true, $g$);
        return $\perp$
    else
      $r \leftarrow t \oplus H(s)$;
      if $r \notin \mathsf{dom}(L_G)$ then
        $g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;  $L_G[r] \leftarrow$ (true, $g$);
      return $\perp$
```

```
Game $G_2^3$ $\fbox{G_2^4}$ $\boxed{G_2^5}$ :
  $L_G, L_H, L_\mathcal{D} \leftarrow$ nil;
  $(pk, sk) \leftarrow \mathcal{K}_f()$;
  $(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
  $b \overset{\$}{\leftarrow} \{0,1\}$;
  $\hat{r} \overset{\$}{\leftarrow} \{0,1\}^{k_0}$;
  $\hat{s} \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;
  $\hat{h} \leftarrow H(\hat{s})$;
  $\hat{t} \leftarrow \hat{h} \oplus \hat{r}$;
  $\hat{c} \leftarrow f(pk, \hat{s} \,\|\, \hat{t})$;
  $\hat{c}_{\mathsf{def}} \leftarrow$ true;
  $\bar{b} \leftarrow \mathcal{A}_2(pk, \hat{c})$
  ┌ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
  $L \leftarrow L_G$;
  while $L \neq$ nil do
    $(r, (b, g)) \leftarrow$ head$(L)$;
    if $b$ = true then
      $g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;
      $L_G[r] \leftarrow$ (true, $g$)
    $L \leftarrow$ tail$(L)$
  └ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
```

```
Oracle $G(r)$ :
  if $r \notin \mathsf{dom}(L_G)$ then
    if $r = \hat{r}$ then
      bad $\leftarrow$ true
    $g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;
    $L_G[r] \leftarrow$ (false, $g$)
  else
    $(d, g) \leftarrow L_G[r]$;
    if $d$ = true then
      $\fbox{$g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;}$
      $\boxed{g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0};}$
      $L_G[r] \leftarrow$ (false, $g$);
      bad$_2$ $\leftarrow P(g, r)$
  return $g$
```

```
Oracle $H(s)$ :
  if $s \notin \mathsf{dom}(L_H)$ then
    $h \overset{\$}{\leftarrow} \{0,1\}^{k_0}$;
    $L_H[s] \leftarrow h$
  else $h \leftarrow L_H[s]$
  return $h$
```

```
Oracle $\mathcal{D}(c)$ :
  if $(\hat{c}_{\mathsf{def}} \wedge \hat{c} = c) \vee q_\mathcal{D} < |L_\mathcal{D}| \vee q_\mathcal{D} + q_G < |L_G|$
  then return $\perp$
  else
    $L_\mathcal{D} \leftarrow (\hat{c}_{\mathsf{def}}, c) :: L_\mathcal{D}$;  $(s, t) \leftarrow f^{-1}(sk, c)$;
    if $s \in \mathsf{dom}(L_H)$ then
      $r \leftarrow t \oplus H(s)$;
      if $r \in \mathsf{dom}(L_G)$ then
        $(d, g) \leftarrow L_G[r]$;
        if $d$ = true then return $\perp$
        else
          if $[s \oplus g]_{k_1} = 0^{k_1}$ then return $[s \oplus g]^n$
          else return $\perp$
      else
        if $r = \hat{r}$ then bad $\leftarrow$ true;
        $g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;  $L_G[r] \leftarrow$ (true, $g$);
        return $\perp$
    else
      $r \leftarrow t \oplus H(s)$;
      if $r \notin \mathsf{dom}(L_G)$ then
        $g \overset{\$}{\leftarrow} \{0,1\}^{k-k_0}$;  $L_G[r] \leftarrow$ (true, $g$);
      return $\perp$
```

$P(g, r) \overset{\text{def}}{=} \exists (d, c) \in L_\mathcal{D}.\ \mathsf{let}\ (s, t) = f^{-1}(sk, c)\ \mathsf{in}\ s \in \mathsf{dom}(L_H)\ \wedge\ r = t \oplus L_H[s]\ \wedge\ [s \oplus g]_{k_1} = 0^{k_1}$

Fig. 3. Games in the transition from G_2 to G_3. Fragments of code inside a box appear only in the game whose name is surrounded by the matching box.

a Boolean flag in the memory of G that will be used to record whether a query originated directly from the adversary or from the decryption oracle. The decryption oracle tests this tag when accessing the memory of G: if the ciphertext queried is valid and its random seed appeared in a previous decryption query, but not yet in a direct query to G, the decryption oracle raises a flag **bad**$_1$.

We show that this can happen with probability 2^{-k_1} for any single query, since the random seed is uniformly distributed and independent from the adversary's view. In this case, the decryption oracle can safely reject the ciphertext, as done in game G_2^2. The proof proceeds in two steps. We first show that game G_2 is observationally equivalent to game G_2^1 using the relational invariant

$$\boldsymbol{L}_G\langle 1\rangle = \mathsf{map}\ (\lambda(r,(b,g)).(r,g))\ \boldsymbol{L}_G\langle 2\rangle$$

where $e\langle 1\rangle$ (resp. $e\langle 2\rangle$) denotes the value that an expression e takes in the left hand side (resp. right-hand side) program in an equivalence. Therefore,

$$\Pr[\mathsf{G}_2\ :\ \mathsf{bad}] = \Pr\left[\mathsf{G}_2^1\ :\ \mathsf{bad}\right]$$

Game G_2^2 is identical to G_2^1, except that it rejects ciphertexts that raise the bad_1 flag. Applying the Fundamental Lemma, we show that

$$\Pr\left[\mathsf{G}_2^1\ :\ \mathsf{bad}\right] \le \Pr\left[\mathsf{G}_2^2\ :\ \mathsf{bad}\right] + \Pr\left[\mathsf{G}_2^2\ :\ \mathsf{bad}_1\right]$$

Our next goal is to show that answers to queries tagged as true can be resampled. However, one cannot directly apply the logic of swapping statements at this stage to resample these answers in G because flag bad_1 is set on $\mathcal{D}$ and depends on them. The solution is to introduce a new game G_2^3 that sets another flag bad_2 in the code of G instead of setting bad_1 in the decryption oracle[3]. Flag bad_2 is raised whenever the adversary queries G with the random seed of a valid ciphertext previously submitted to the decryption oracle. We prove that games G_2^2 and G_2^3 satisfy the relational invariant:

$$\mathsf{bad}_1\langle 1\rangle \implies (\mathsf{bad}_2 \vee \phi)\langle 2\rangle$$

where the predicate ϕ is defined as

$$\begin{array}{l}\exists (d,c) \in \boldsymbol{L}_{\mathcal{D}}.\ \mathsf{let}\ (s,t) = f^{-1}(\boldsymbol{sk}, c),\ r = t \oplus \boldsymbol{L}_H[s]\ \mathsf{in} \\ \quad r \in \mathsf{dom}(\boldsymbol{L}_G) \wedge s \in \mathsf{dom}(\boldsymbol{L}_H) \wedge \mathsf{fst}(\boldsymbol{L}_G[r]) = \mathsf{false} \wedge [s \oplus \mathsf{snd}(\boldsymbol{L}_G[r])]_{k_1} = 0^{k_1}\end{array}$$

Therefore:

$$\Pr\left[\mathsf{G}_2^2\ :\ \mathsf{bad}\right] + \Pr\left[\mathsf{G}_2^2\ :\ \mathsf{bad}_1\right] \le \Pr\left[\mathsf{G}_2^3\ :\ \mathsf{bad}\right] + \Pr\left[\mathsf{G}_2^3\ :\ \mathsf{bad}_2 \vee \phi\right]$$

We now consider game G_2^4 where oracle G resamples the answers to queries previously sampled in the decryption oracle. As such answers are uniformly distributed and independent from the adversary's view, the logic for swapping statements can be used to establish that this transformation preserves semantics. Hence:

$$\Pr\left[\mathsf{G}_2^3\ :\ \mathsf{bad}\right] + \Pr\left[\mathsf{G}_2^3\ :\ \mathsf{bad}_2 \vee \phi\right] = \Pr\left[\mathsf{G}_2^4\ :\ \mathsf{bad}\right] + \Pr\left[\mathsf{G}_2^4\ :\ \mathsf{bad}_2 \vee \phi\right]$$

[3] As bad_1 is not set anymore, we simplify the code of $\mathcal{D}$ by coalescing branches in the innermost conditional.

Note that in order to prove semantic equivalence we need to resample the values in $\boldsymbol{L}_G$ associated to queries tagged as true—made by the $\mathcal{D}$—at the end of the game. Using the Failure Event Lemma of [7], we upper bound the probability of $\textbf{bad}_2 \vee \phi$ in G_2^4:

$$\Pr\left[\mathsf{G}_2^4 \,:\, \textbf{bad}_2 \vee \phi\right] \leq \frac{q_{\mathcal{D}}}{2^{k_1}}$$

We are now only interested in bounding $\textbf{bad}$, so we can remove as dead code the fragment of code at the end of G_2^4 that resamples values in $\boldsymbol{L_G}$, obtaining G_2^5, and prove that

$$\Pr\left[\mathsf{G}_2^4 \,:\, \textbf{bad}\right] = \Pr\left[\mathsf{G}_2^5 \,:\, \textbf{bad}\right]$$

We finally prove that game G_2^5 is observationally equivalent to G_3, in which the code for the oracle G is reverted to its original form and the decryption oracle no longer tampers with the memory of G. Thus,

$$\Pr[\mathsf{G}_2 \,:\, \textbf{bad}] \leq \Pr\left[\mathsf{G}_2^5 \,:\, \textbf{bad}\right] + \frac{q_{\mathcal{D}}}{2^{k_1}} = \Pr[\mathsf{G}_3 \,:\, \textbf{bad}] + \frac{q_{\mathcal{D}}}{2^{k_1}} \qquad \square$$

Comparison with the security bound in [27]. Pointcheval obtains a slightly different bound:

$$\epsilon' \geq \left(\frac{\epsilon}{2} - \frac{4q_{\mathcal{D}}q_G + 2q_{\mathcal{D}}^2 + 4q_{\mathcal{D}} + 8q_G}{2^{k_0}} - \frac{3q_{\mathcal{D}}}{2^{k_1}}\right)$$

We marginally improve on this bound by reducing the coefficients. As previously mentioned, the improvement stems from the transition from G_2 to G_3, where we eliminate both calls to G, whereas only the second call is eliminated in [27]. In fact, eliminating both calls is not only useful to give a better bound, but also essential for the correctness of the proof. Indeed, the transition from G_3 to G_4 would not be possible if $\mathcal{D}$ modified the memory of G. Concretely, the justification of Equation (27) in [27] contains two minor glitches: firstly, the remark "which just cancels r' from $\boldsymbol{L}_G$" oversees the possibility of this removal having an impact on future queries. Secondly, "the probability for r' to be in $\boldsymbol{L}_G$ is less than $q_G/2^{k_0}$" oversees that the length of $\boldsymbol{L}_G$ is upper bounded by $q_G + q_{\mathcal{D}}$ rather than q_G, as the decryption oracle still adds values to $\boldsymbol{L}_G$; a correct bound for this probability in [27] is $(q_G + q_{\mathcal{D}})/2^{k_0}$.

5 Perspectives

The CertiCrypt framework consists of over 30,000 lines of Coq. Less than 5% of the development is part of the trusted base, covering the definition of the semantics, of well-formed adversaries, and of probabilistic polynomial-time programs. The remaining 95% consist of proof tools, including the mechanization of common program transformations, of observational equivalence and Relational Hoare Logic, and of the Fundamental Lemma of Game-Playing. The logic of swapping statements, and the Failure Event Lemma, that have been developed specifically for the purpose of this proof, account for about 1,300 and 500 lines of Coq, respectively.

The verifiable proof is over 10,000 lines of Coq scripts, and can be checked fully automatically using Coq version 8.2pl1, the latest, as yet unreleased, version of Audebaud and Paulin's library of probabilities, and the current implementation of the CertiCrypt framework. Most importantly, less than 1% of the verifiable proof needs to be trusted, namely the formal statement of Figure 1.

The structure of the formal proof is more fine grained than the outline of Figure 2, and contains about 30 games. For example, just the transition from $\mathsf{G}_{\mathsf{IND\text{-}CCA}}$ to G_1 overviewed in Section 4 accounts for 10 games. Decomposing transitions into intermediate games is mostly a matter of taste, but common wisdom in formal proofs is to introduce many intermediate lemmas with short proofs rather than a few lemmas with intricate proofs.

The overall proof was completed within about 6 man-months. While substantial, and perhaps even a bit discouraging for a scientist without experience in formal proofs, the effort required to complete the proof is reasonable in comparison with other large-scale formalization projects. Moreover, a significant amount of work was devoted to pinpoint the details of the proof, and to find a means to capture formally "independence from the adversary's view". We expect that formalizing related proofs in the line of [4,13] would now be significantly faster.

Still, the time and expertise required for developing formal proofs currently make verifiable security an exclusive option that might be considered for proving standards, but that is otherwise too costly for cryptographers to use in their own research. In an attempt to make verifiable security a reasonable (and we believe profitable) alternative for the working cryptographer, we are building dedicated proof engines to which most of the construction of a verifiable proof could be delegated. Preliminary experiments suggest that most formalizations in CertiCrypt, including our proof of OAEP and the proofs in [6,33], rely on relational invariants that fall in a decidable fragment of predicate logic, and that can be established through simple heuristics. We are currently developing a front-end to CertiCrypt that extracts verifiable proofs from a proof sketch submitted by the user consisting of a sequence of games and statements that justify transitions, including relational invariants.

6 Related Work

The motivations behind verifiable security appear in Bellare and Rogaway's seminal article on code-based game-playing proofs [10], and in Halevi's manifesto for computer-aided cryptographic proofs [20]. However, the most extensive realization of verifiable security to date is CertiCrypt [6], which has been used previously to build verifiable security proofs of the existential unforgeability of FDH signatures (both for the conventional and optimal bounds) and of semantic security of OAEP. CertiCrypt is particularly suitable for formalizing proofs involving algebraic and number-theoretic reasoning, since in addition to automating common techniques used in game-based cryptographic proofs, it gives access to the full expressive power of the logic of Coq and to the many available libraries and theories developed using it. There is a leap in complexity between the proof of IND-CPA security of OAEP and the proof of IND-CCA security presented here. Specifically, tools such as the Failure Event Lemma and the logic of swapping statements were developed to tackle difficulties arising in some transitions in the latter proof. In another attempt to build a system that supports verifiable security, Backes, Berg and Unruh [3] formalize a language for games in the Isabelle proof assistant, and prove the Fundamental Lemma; however, no specific example is reported. Nowak [25], and Affeldt, Marti and Tanaka [1] also report on preliminary experiments with machine-checked proofs.

CryptoVerif [11] is a prover for exact security of cryptographic schemes and protocols in the computational model; it has been used to verify Kerberos [12] and the conventional bound of FDH [11]. CryptoVerif trades off generality for automation, and consequently adopts a non-standard axiomatization of cryptographic primitives based on term rewriting. As a result, sequences of games can sometimes be inferred automatically; yet, at the same time, the connection between CryptoVerif proofs and standard cryptographic proofs is not as strong as one would desire. Finally, CryptoVerif in its current form acts more like a proof engine than a proof checker, and thus does not comply with the objective of verifiable security—see however [17] for preliminary work on certifying successful runs of CryptoVerif. Courant et al. [14] have also developed an automated prover for proving asymptotic security of encryption schemes based on one-way functions. Their prover is able to handle many schemes from the literature, but it cannot handle OAEP. As CryptoVerif, their tool is a proof engine and does not generate verifiable proofs. More recently, Barthe et al. [5] propose a computationally sound logic to reason about cryptographic primitives. Their logic captures many common reasoning steps in cryptographic proofs and has been used to prove the exact security of PSS. There is no tool support for this logic.

Somehow surprisingly, Koblitz [23] recently published an article that vehemently dismisses relying on computer-assisted proof building and proof checking. While Koblitz rightfully points to some weaknesses of existing tools—e.g. lack of automation and unduly verbosity—a closer look at the article reveals a fragmentary knowledge of the state-of-the-art in machine-checked proofs, and a profound misconception on the role of formal verification.

7 Conclusion

Verifiable security goes beyond provable security by providing independently verifiable evidence that proofs are correct. We used the Coq proof assistant to build the first verifiable proof of IND-CCA security of OAEP. Our proof is a strong indicator that proof assistants are mature enough to support the construction of cryptographic proofs, and gives strong empirical evidence that dedicated tactics could improve automation and reduce the length and development time of formal proofs. Making verifiable security an appealing alternative for working cryptographers is the next objective.

References

1. Affeldt, R., Tanaka, M., Marti, N.: Formal proof of provable security by game-playing in a proof assistant. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 151–168. Springer, Heidelberg (2007)
2. Audebaud, P., Paulin-Mohring, C.: Proofs of randomized algorithms in Coq. Sci. Comput. Program. 74(8), 568–589 (2009)
3. Backes, M., Berg, M., Unruh, D.: A formal language for cryptographic pseudocode. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 353–376. Springer, Heidelberg (2008)
4. Backes, M., Dürmuth, M., Unruh, D.: OAEP is secure under key-dependent messages. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 506–523. Springer, Heidelberg (2008)
5. Barthe, G., Daubignard, M., Kapron, B., Lakhnech, Y.: Computational indistinguishability logic. In: 17th ACM Conference on Computer and Communications Security, CCS 2010. ACM, New York (2010)
6. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Formal certification of code-based cryptographic proofs. In: 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, pp. 90–101. ACM, New York (2009)
7. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Programming language techniques for cryptographic proofs. In: Kaufmann, M., Paulson, L.C. (eds.) ITP 2010. LNCS, vol. 6172, pp. 115–130. Springer, Heidelberg (2010)
8. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: When and how should challenge-decryption be disallowed? Cryptology ePrint Archive, Report 2009/418 (2009)
9. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
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. Blanchet, B.: A computationally sound mechanized prover for security protocols. IEEE Trans. Dependable Sec. Comput. 5(4), 193–207 (2008)
12. Blanchet, B., Jaggard, A.D., Scedrov, A., Tsay, J.-K.: Computationally sound mechanized proofs for basic and public-key Kerberos. In: 15th ACM Conference on Computer and Communications Security, CCS 2008, pp. 87–99. ACM, New York (2008)
13. Boldyreva, A.: Strengthening security of RSA-OAEP. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 399–413. Springer, Heidelberg (2009)
14. Courant, J., Daubignard, M., Ene, C., Lafourcade, P., Lakhnech, Y.: Towards automated proofs for asymmetric encryption schemes in the random oracle model. In: 15th ACM Conference on Computer and Communications Security, CCS 2008, pp. 371–380. ACM, New York (2008)
15. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. J. Cryptology 17(2), 81–104 (2004)
16. Gonthier, G.: Formal Proof — The Four Colour Theorem. Notices of the AMS 55(11), 1382–1393 (2008)
17. Goubault-Larrecq, J.: Towards producing formally checkable security proofs, automatically. In: 21st IEEE Computer Security Foundations Symposium, CSF 2008, pp. 224–238. IEEE Computer Society, Los Alamitos (2008)
18. Hales, T.: Formal Proof. Notices of the AMS 55(11), 1370–1380 (2008)

19. Hales, T., Harrison, J., McLaughlin, S., Nipkow, T., Obua, S., Zumkeller, R.: A revision of the proof of the Kepler conjecture. Discrete and Computational Geometry 44(1), 1–34 (2010)
20. Halevi, S.: A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Report 2005/181 (2005)
21. Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: formal verification of an OS kernel. In: 22nd ACM Symposium on Operating Systems Principles, SOSP 2009, pp. 207–220. ACM Press, New York (2009)
22. Klein, G., Nipkow, T.: A machine-checked model for a Java-like language, virtual machine and compiler. ACM Trans. Program. Lang. Syst. 28(4), 619–695 (2006)
23. Koblitz, N.: Another look at automated theorem-proving. J. Math. Cryptol. 1(4), 385–403 (2008)
24. Leroy, X.: Formal certification of a compiler back-end, or: programming a compiler with a proof assistant. In: 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006, pp. 42–54. ACM, New York (2006)
25. Nowak, D.: A framework for game-based security proofs. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 319–333. Springer, Heidelberg (2007)
26. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. J. of Comput. Secur. 6(1-2), 85–128 (1998)
27. Pointcheval, D.: Provable security for public key schemes. In: Advanced Courses on Contemporary Cryptology, ch. D, pp. 133–189. Birkhäuser, Basel (2005)
28. Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA (1979)
29. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978)
30. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 239–259. Springer, Heidelberg (2001)
31. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004)
32. The Coq development team. The Coq Proof Assistant Reference Manual Version 8.2. Online (2009), http://coq.inria.fr
33. Zanella Béguelin, S., Grégoire, B., Barthe, G., Olmedo, F.: Formally certifying the security of digital signature schemes. In: 30th IEEE Symposium on Security and Privacy, S&P 2009, pp. 237–250. IEEE Computer Society, Los Alamitos (2009)

(Second) Preimage Attacks on Step-Reduced RIPEMD/RIPEMD-128 with a New Local-Collision Approach

Lei Wang[1], Yu Sasaki[2], Wataru Komatsubara[1], Kazuo Ohta[1], and Kazuo Sakiyama[1]

[1] The University of Electro-Communications
{wanglei,wkoma,ota,saki}@ice.uec.ac.jp
[2] NTT Corporation
sasaki.yu@lab.ntt.co.jp

Abstract. This paper uses new types of local collisions named *one-message-word local collisions* to construct meet-in-the-middle preimage attacks on two double-branch hash functions RIPEMD and RIPEMD-128, and obtains the following results.

1) A pseudo-preimage and second preimage attacks on the first 47 steps of RIPEMD (full version: 48 steps) are proposed with complexities of 2^{119} and $2^{124.5}$ compression function computations, respectively. The number of the attacked steps is greatly increased from previous preimage attacks on the first 33 steps and intermediate 35 steps.
2) A pseudo-preimage and preimage attacks on intermediate 36 steps of RIPEMD-128 (full version: 64 steps) are proposed with complexities of 2^{123} and $2^{126.5}$ compression function computations, respectively, while previous attacks can work at most intermediate 35 steps.

Keywords: RIPEMD, RIPEMD-128, Meet-in-the-Middle, One-Message-Word Local Collision.

1 Introduction

Cryptographic hash function is one of the most basic primitives in modern cryptography which supports the security of various systems. Traditionally, hash functions producing n-bit digests are required to have (second) preimage resistance up to 2^n computations.

Recently cryptanalysts are paying a lot of attentions to evaluating preimage resistance of dedicated hash functions. Several hash functions such as MD4, MD5 and Tiger have been broken in the sense of preimage resistance [10,13,7]. Step-reduced variants of other hash functions such as SHA-0, SHA-1 and SHA-2 have been found weakness in the sense of preimage resistance [4,2]. Among these preimage attack results, most follow a framework named *Meet-in-the-Middle Preimage Attack*, which is devised by Aoki and Sasaki [3] and works efficiently on narrow-pipe Merkle-Damgård hash functions such as MD4-family.

However, this framework seems to have a limitation on its applicability to a double-branch Merkle-Damgård hash function, whose compression function computes two branches of transformations and generates the output by mixing the results of the two branches. Because the internal state size of such a compression function becomes $2n$ bits, a trivial meet-in-the-middle pseudo-preimage attack on it costs at least 2^n computations, which has no advantage compared to the brute force attack. Thus it is difficult to apply the meet-in-the-middle preimage attack framework on double-branch hash functions.

This paper will deal with preimage resistance of two famous double-branch hash functions RIPEMD [6] and RIPEMD-128 [5]. RIPEMD-128 has been standardized by ISO/IEC. Using several new observations found by us, we will show how to make it more feasible to tailor the meet-in-the-middle preimage attack framework to attack these two hash functions.

Related works. Wang *et al.* proposed the first preimage attack on RIPEMD at ISPEC2009 [15], and claimed preimage attacks on the first 26 steps and intermediate 29 steps of RIPEMD[1]. Later at ACISP2009, Sasaki *et al.* firstly tried to apply the meet-in-the-middle preimage attack framework on RIPEMD [14], and found improved preimage attacks on the first 33 steps and intermediate 35 steps of RIPEMD with complexities of $2^{125.5}$ and 2^{113} compression function computations, respectively.

Independently from our work, recently Ohtahara *et al.* published preimage attacks on RIPEMD-128 at INSCRYPT2010 [11]. They proposed preimage attacks on the first 33 steps and intermediate 35 steps of RIPEMD-128 with complexities of $2^{124.5}$ and 2^{121} compression function computations, respectively.

We notice that a preimage attack on a 4-branch hash function named FORK-256 was published [12] at INDOCRYPT2007. This attack is based on a particular weakness of the message schedule algorithm of FORK-256. It seems hard to apply the same attack approach to RIPEMD or RIPEMD-128.

Our contributions. This paper will propose improved preimage attacks on RIPEMD and RIPEMD-128. First of all, we will revisit Sasaki *et al.*'s strategies on applying the meet-in-the-middle preimage attack framework to double-branch hash functions [14], and reveal two underlying restrictions which limit the number of the attacked steps. Then we will present new observations to relax these two restrictions. More precisely, our new observations are constructing local collisions by using only one message word, which are named *one-message-word local collisions* in this paper. We will use two types of one-message-word local collisions: 1) one-message-word local collisions in a single branch; and 2) one-message-word local collisions spanning the first several steps of the two branches. Finally we succeed to increase the number of the attacked steps. The results of our attacks and a comparison with previous attacks are summarized in Table 1.

[1] Their attacks seem to contain a small flaw: they used a message-word order exactly the same as that of MD4, whereas RIPEMD uses different one. If the correct message-word order is used, their attack seems to work on more steps of RIPEMD; the first 31 steps instead of 26 steps.

Table 1. A Comparison with Previous Preimage Attacks

Hash Function	#Steps	Pseudo-Preimage Attacks	(Second) Preimage Attacks	Memory	Reference
RIPEMD	26	2^{110}	$2^{115.2}$	2^{23}	[15]
	29 (*)				
	33	2^{121}	$2^{125.5}$	2^{10}	[14]
	35 (*)	2^{96}	2^{113}	2^{35}	
	47 (s)	2^{119}	$2^{124.5}$	$2^{10.5}$	Ours
RIPEMD-128	33	2^{119}	$2^{124.5}$	2^{12}	[11]
	35 (*)	2^{112}	2^{121}	2^{16}	
	36 (*)	2^{123}	$2^{126.5}$	$2^{6.5}$	Ours

'*' means that the attacked steps start from some intermediate step.
's' means that the attack can only be applied to find second preimages.

2 Specifications and Related Works

2.1 RIPEMD Hash Function

RIPEMD [6] is a Merkle-Damgård hash function using a double-branch compression function. We will give a brief description. For the completed specification, refer to [6].

To produce a RIPEMD digest for a message M, first pad M to be a multiple of 512 bits long following the standard padding method of the Merkle-Damgård mode. Then divide the padded message into 512-bit blocks $M_1||M_2||\cdots||M_t$. Finally hash these blocks by iterating a compression function (CF): $h_i \longleftarrow CF(h_{i-1}, M_i)$, where h_0 is a public constant. h_t will be a RIPEMD digest of M. All h_is ($0 \leq i \leq t$) are 128 bits long.

RIPEMD compression function. First divide h_{i-1} and M_i into 32-bit variables $Q_{-3}||Q_0||Q_{-1}||Q_{-2}$ and $X_0||X_1||\cdots||X_{15}$ respectively. Then process them through two branches of MD4-compression-function-like transformations. Each branch updates $Q_{-3}||Q_0||Q_{-1}||Q_{-2}$ by three round functions, each round function iterates a step function sixteen times, and each step function updates one internal state word Q_i by using one message word X_j. The step function is

$$Q_i = (Q_{i-4} + F(Q_{i-1}, Q_{i-2}, Q_{i-3}) + X_j + K_i) \lll S_i,$$

where K_i and S_i are public constants, $\lll S_i$ is a cyclic rotation to left by S_i bits, and F is a public Boolean function. The two branches differ in the values of the constants $\{K_i\}$. In the rest of this paper, we will denote by K_i and K'_i the constants in the left branch and in the right branch respectively. Similarly we denote by Q_i and Q'_i the internal state words in the left branch and in the right branch respectively.[2] Finally mix $Q_{-3}||Q_0||Q_{-1}||Q_{-2}$, $Q_{45}||Q_{48}||Q_{47}||Q_{46}$ and $Q'_{45}||Q'_{48}||Q'_{47}||Q'_{46}$ to produce h_i as follows.

[2] In order to show that the two branches share the same initial state words, we will still use the notation $Q_{-3}||Q_0||Q_{-1}||Q_{-2}$ as the initial state in the right branch.

Table 2. Parameters of RIPEMD Compression Function

First Round	j of X_j	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 , 13 , 14, 15
	K_i (K'_i)	`0x00000000` (`0x50a28be6`)
	F	$(Q_{i-1} \wedge Q_{i-2}) \vee (\neg Q_{i-1} \wedge Q_{i-3})$
	S_i	11, 14, 15, 12, 5, 8, 7, 9, 11, 13, 14, 15, 6, 7, 9, 8
Second Round	j of X_j	7, 4, 13, 1, 10, 6, 15, 3, 12, 0, 9, 5, 14, 2, 11, 8
	K_i (K'_i)	`0x5a827999` (`0x00000000`)
	F	$(Q_{i-1} \wedge Q_{i-2}) \vee (Q_{i-2} \wedge Q_{i-3}) \vee (Q_{i-1} \wedge Q_{i-3})$
	S_i	7, 6, 8, 13, 11, 9, 7, 15, , 7, 12, 15, 9, 7, 11, 13, 12
Third Round	j of X_j	3, 10, 2, 4, 9, 15, 8, 1, 14, 7, 0, 6, 11, 13, 5, 12
	K_i (K'_i)	`0x6ed9eba1` (`0x5c4dd124`)
	F	$Q_{i-1} \oplus Q_{i-2} \oplus Q_{i-3}$
	S_i	11, 13, 14, 7, 14, 9, 13, 15, 6, 8, 13, 6, 12, 5, 7, 5

$$h_i = (Q_0+Q_{47}+Q'_{46})||(Q_{-1}+Q_{46}+Q'_{45})||(Q_{-2}+Q_{45}+Q'_{48})||(Q_{-3}+Q_{48}+Q'_{47}).$$

The details of K_i, S_i, F and X_j are shown in Table 2.

2.2 RIPEMD-128 Hash Function

RIPEMD-128 [5] follows the same design framework with RIPEMD. The differences between RIPEMD-128 and RIPEMD exist in the number of rounds and the instantiation of the parameters in every round. The parameters in every round of RIPEMD-128 are detailed in Appendix A.

2.3 Meet-in-the-Middle Preimage Attack Framework

We briefly describe the framework of the meet-in-the-middle preimage attack. For a completed description, refer to [3]. First the attacker builds a pseudo-preimage attack on compression function, and then extends it to a preimage attack on hash function.

Pseudo-preimage attack on a compression function. The attacker divides the computation of a compression function into two independent computations, and then searches a pseudo-preimage by matching the outputs of the two independent computations. Such a strategy transforms the problem of finding a pseudo-preimage to another problem of finding a collision between the two independent computations. Thus the complexity can be reduced, and the attack is faster than the brute force attack. Suppose that each independent computation has l free bits. The complexity of finding a pseudo-preimage will become 2^{n-l}.

Usually the attacker will separate a compression function into two parts so that one part will not involve a message word X_i, and the other part will not involve a message word X_j $(i \neq j)$. X_i and X_j are named *neutral message words*. These two parts can be independently computed as follows: fix all the message words except X_i and X_j, and compute the former part by randomly choosing the value of X_j and the latter part by randomly choosing the value of X_i.

Preimage attacks on a hash function. The attacker first generates several pseudo-preimages $\{(h^i, M^i)\}$ of the target digest, and then links h_0 to the values of $\{h^i\}$ exhaustively. If a message linking h_0 to one h^i is successfully found, a preimage of the target digest has been generated. If finding a pseudo-preimage needs 2^{n-l} computations, the complexity of preimage attack will be $2^{n-\frac{l}{2}+1}$.

Overall, the existence of at least 2 well-positioned message words is essential for applying the meet-in-the-middle preimage attack framework. When $l > 2$ holds, the meet-in-the-middle preimage attack will be faster than the brute force attack.

2.4 Revisit Sasaki *et al.*'s Meet-in-the-Middle Preimage Attack Strategies on Double-Branch Hash Functions [14]

This section will mainly revisit one of Sasaki *et al.*'s attack strategies, which is illustrated in Figure 1. They used the technique *Local Collision (LC)*.[3]

Loosely speaking, a local collision between steps i and j means that the values of the message words, which are used between steps i and j, can be adaptively modified without influencing the values of the internal states right before step i and at step j. As shown in Figure 1, Sasaki *et al.* use two message words X_i and X_t to construct a local collision in a single branch, e.g. the left branch. Thanks to this local collision, the value of the initial state h will be a constant. Thus the two computations are independent.

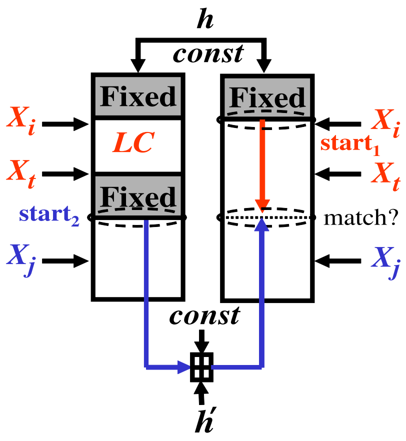


Fig. 1. Sasaki *et al.*'s Attack Strategy [14]

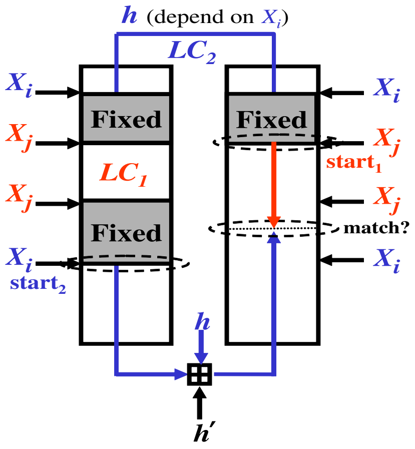


Fig. 2. Our Attack Strategy

[3] *LC* is the terminology named by [14]. The technique *Tunnel* proposed by Klima [9] can play the same role.

There are two restrictions on the power of their attack strategy, which are detailed below.

Restriction I. Using two-message-word local collisions limits the candidate space of the potential neutral message words. As we can see, three message words (X_i, X_t) and X_j are used as the neutral message words. Thus the attacker must find three well-positioned message words in order to apply the meet-in-the-middle preimage attack framework.

Restriction II. Typically a message word will be used in every round in each branch. So it is obvious that the number of the attacked steps cannot bypass two rounds if the attack target is from the first step. Otherwise, the message words X_i and X_t will surely appear in the computation related to X_j.

3 New Observations

This section will use several new observations to relax the two restrictions on Sasaki *et al.*'s attack strategy detailed in Section 2.4.

3.1 Relax Restriction I: One-Message-Word Local Collision in a Single Branch

In order to enlarge the candidate space of the potential neutral message words, we introduce a new type of local collisions: *one-message-word local collision in a single branch.*

Suppose that a message word X_i is used at steps i_i and i_2 $(i_1 < i_2)$ in a branch. If the value of X_i can be adaptively modified without influencing the values of the internal states right before step i_1 and at step i_2, we will call that X_i constructs a one-message-word local collision between steps i_1 and i_2 in that branch (Refer to Section 4.1 for a concrete example).

As we can see, using such one-message-word local collisions, the attacker only needs to find two well-positioned neutral message words, which is not covered by previous Sasaki *et al.*'s attack strategy. Therefore we can enlarge the candidate space of the potential neutral message words.

Efficiency of such local collisions. We will use a toy example to show both the feasibility and the freedom degree of such a one-message-word local collision. We stress that a general case can be easily derived from this example. Suppose that a message word X_j is used at steps i and $i+4$ in the left branch of RIPEMD. How to construct a one-message-word local collision using X_j is detailed as below.

Step $i-1$. Set the value of the internal state at this step as a constant satisfying $Q_{i-4} + F(Q_{i-1}, Q_{i-2}, Q_{i-3}) + K_i =$ `0x00000000`.

Step i. The internal state word Q_i is computed as $(Q_{i-4}+F(Q_{i-1}, Q_{i-2}, Q_{i-3})+K_i + X_j) \lll S_i = X_j \lll S_i$.

Step $i+1$**.** Use the absorption properties of F to keep the internal state word Q_{i+1} as a constant, namely independent from the value of Q_i.

Step $i+2$**.** Similarly to step $i+1$, Q_{i+2} is a constant.

Step $i+3$**.** Similarly to step $i+1$, Q_{i+3} is a constant.

Step $i+4$**.** From $Q_{i+4} = (Q_i + X_j + F(Q_{i+1}, Q_{i+2}, Q_{i+3}) + K_{i+4}) \lll S_{i+4}$, if the value of X_j can be adaptively modified without changing the value of $Q_i + X_j$, Q_{i+4} will be a constant, which means the internal state at step $i+4$ $Q_{i+1}||Q_{i+4}||Q_{i+3}||Q_{i+2}$ is a constant. Thus a local collision between steps i and $i+4$ is constructed.

Another concern is about the freedom degree of X_j on keeping $X_j + Q_i$, more precisely $X_j + (X_j \lll S_i)$, as a constant. We point out that the freedom degree of X_j is related to the value of $gcd(32, S_i)$. Denote $gcd(32, S_i)$ *mod* 32 by g. The freedom degree of X_j should be 2^g to make the value of $X_j + (X_j \lll S_i)$ be the constant `0xffffffff`. This can be easily verified. Note that g may be 1, 2, 4, 8 or 16. Pick $S_i = 24$ and $g = 8$ as an example.[4] Divide X_j into four bytes $X_{j,3}||X_{j,2}||X_{j,1}||X_{j,0}$. From $X_j + (X_j \lll 24) = (X_{j,3}||X_{j,2}||X_{j,1}||X_{j,0}) + (X_{j,0}||X_{j,3}||X_{j,2}||X_{j,1})$ = `0xffffffff`, we can derive the following relations: $X_{j,1} = \texttt{0xff} - X_{j,0}$; $X_{j,2} = \texttt{0xff} - X_{j,1} = X_{j,0}$; and $X_{j,3} = \texttt{0xff} - X_{j,2} = \texttt{0xff} - X_{j,0}$. If we adaptively choose the values of $X_{j,3}$, $X_{j,2}$ and $X_{j,1}$ with the value of $X_{j,0}$ following these relations, $(X_{j,3}||X_{j,2}||X_{j,1}||X_{j,0}) + (X_{j,0}||X_{j,3}||X_{j,2}||X_{j,1})$ will always be `0xffffffff` no matter what the value of $X_{j,0}$ is. So the freedom degree of X_j for the local collision is at least 2^8. We stress that the other cases of g can also be easily verified. Due to the limited space, we will omit the details.

3.2 Relax Restriction II: One-Message-Word Local Collisions Spanning the Two Branches

This section will propose another type of one-message-word local collisions which span the first several steps of the two branches.

Suppose that a message word X_i is used at step t in both branches. If adaptively modifying the value of X_i will not influence the values of the internal states at step t in both branches, we will call that X_i constructs a one-message-word local collision spanning the first t steps of the two branches.

Efficiency of such local collisions. We also use an example to show both the feasibility and freedom degree of such a local collision. In RIPEMD, X_0 is used at step 1 in both the left and the right branches. The computation is as follows.

$$Q_1 = (X_0 + Q_{-3} + F(Q_0, Q_{-1}, Q_{-2}) + K_0) \lll 11;$$
$$Q'_1 = (X_0 + Q_{-3} + F(Q_0, Q_{-1}, Q_{-2}) + K'_0) \lll 11;$$

As long as the values of $X_0 + Q_{-3}$, Q_0, Q_{-1} and Q_{-2} are kept as a constant, the values of the internal states after step 1 in both branches will be constants. So

[4] This example is actually used in our attack on RIPEMD, which is described in Section 4.1.

we will adaptively modify the values of X_0 and Q_{-3} to construct a one-message-word local collision spanning the first step of the two branches. The freedom degree of X_0 in this local collision is obviously 2^{32}.

In next section, we will show how exactly to use such a one-message-word local collision to make the number of the attacked steps bypass two rounds even when the target is from the first step.

3.3 Our Attack Strategy

This section describes our attack strategy using the two types of one-message-word local collisions.

As shown in Figure 2, X_j and X_i are chosen as the neutral message words. Denote the two steps, where X_j is used, as steps j_1 and j_2 ($j_1 < j_2$). Similarly denote the two steps, where X_i is used, as steps i_1 and i_2 ($i_1 < i_2$).

X_j will construct a one-message-word local collision LC_1 between steps j_1 and j_2 in a single branch, e.g. the left branch, following the technique in Section 3.1. X_i will construct another one-message-word local collision LC_2 spanning the first i_1 steps of the two branches, following the technique in Section 3.2.

We are able to carry out two independent computations: 1) one computation starts from step j_1 until step $i_2 - 1$ in the right branch, which depends on X_j but is independent from X_i; 2) the other computation starts with a forward computation from step i_2 in the left branch, then goes through the feed-forward operation, and finally ends with a backward computation until step i_2 in the right branch, which depends on X_i but is independent from X_j. As we can see, these two computations are independent, and can be matched in the right branch.

We stress that each of X_i and X_j appears at least twice in either the left or the right branches. Thus the number of the attacked steps in our attack strategy will bypass two rounds, even when the target is from the first step.

4 Our Attacks on 47-Step RIPEMD

This section presents our pseudo-preimage attack on 47-step RIPEMD compression function, which can be converted to a second preimage attack on 47-step RIPEMD hash function. The neutral message words we use are X_0 and X_{12}. The overview of our pseudo-preimage attack is given in Figure 3. Note that our attack target is the first 47 steps, so the last step is erased. X_{12} constructs LC_1 (the same notation as Figure 2) in the left branch, which is detailed in Section 4.1. X_0 constructs LC_2 (the same notation as Figure 2), which is detailed in Section 4.2.

4.1 The Details of LC_1

X_{12} locates at steps 13 and 25 in each branch of RIPEMD. In order to construct LC_1, we will use the absorption properties of the Boolean function to control the influence of modifying X_{12} to the internal state words between steps 13 and

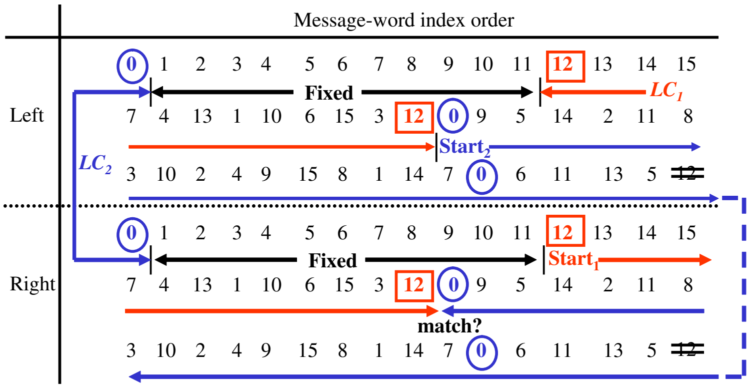


Fig. 3. Overview of Our Pseudo-Preimage Attack on RIPEMD

Table 3. All the Conditions for LC_1

Step Index	i	$i+1$	$i+2$
$i = 14$	$Q_{12} = Q_{11}$;	$Q_{14} =$ `0x00000000`;	$Q_{15} =$ `0xffffffff`;
$i = 18$	$Q_{16} = Q_{15}$;	$Q_{18} = Q_{16}$;	$Q_{19} = Q_{18}$;
$i = 22$	$Q_{20} = Q_{19}$;	$Q_{22} = Q_{20}$;	$Q_{23} = Q_{22}$;

25, which is shown in Figure 4. More precisely, we will set conditions on the internal state words, which are independent from X_{12} between steps 13 and 25. All the conditions are listed in Table 3. Here we pick step 14 as an example to show how we set conditions. All the other conditions can be similarly and easily derived. At step 14, Q_{13} will change with X_{12}, and we want to keep Q_{14} as a constant, namely independent from X_{12}. The Boolean function F at this step is $(Q_{13} \wedge Q_{12}) \vee (\neg Q_{13} \wedge Q_{11})$. We can see that by setting a condition $Q_{12} = Q_{11}$, the output of F will always be equal to Q_{12}. Thus Q_{14} will be a constant and independent from X_{12}.

Set up LC_1. We will explain how to exactly set up the local collision LC_1 for our attack. Set the related internal state words satisfy the conditions in Table 3. More precisely, Q_{14} is fixed as `0x00000000`, and all the state words $\{Q_{15}, Q_{16}, Q_{18}, Q_{19}, Q_{20}, Q_{22}, Q_{23}\}$ are fixed as `0xffffffff`. These conditions will at the same time fix the values of several message words X_4, X_{13}, X_1, X_6 and X_{15} at steps 18, 19, 20, 22 and 23, respectively. As an example, X_{13} can be computed at step 19 as $(Q_{19} \ggg 8) - F(Q_{18}, Q_{17}, Q_{16}) - Q_{15} - K_{19}$. Moreover, two message words X_{13} and X_{15} are used twice in this local collision, which have been fixed at steps 19 and 23 respectively. Then at step 16, Q_{12} will be computed. At step 15, because of the condition $Q_{11} = Q_{12}$, X_{14} will deterministically be computed. At step 14, Q_{10} will also be computed.

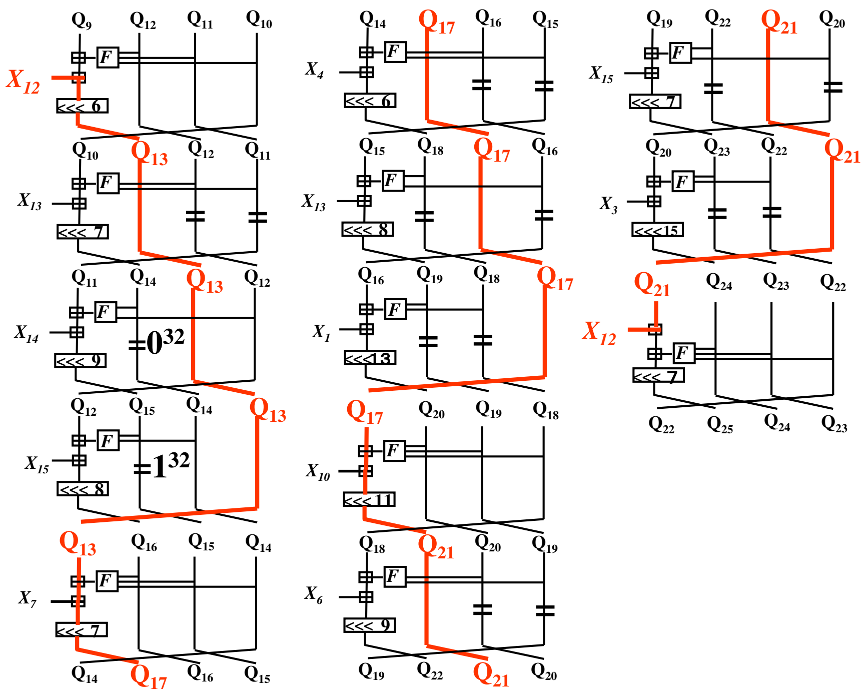


Fig. 4. One-Message-Word Local Collision LC_1
0^{32} and 1^{32} are `0x00000000` and `0xffffffff` respectively

So far, we have successfully controlled the influence of modifying X_{12} to the internal state words exactly the same as shown in Figure 4. The next question is that how much freedom degree X_{12} has in LC_1.

This paper will present a method to make the freedom degree of X_{12} be 2^8 in LC_1. More precisely, set conditions on Q_9, X_7 and X_{10} at steps 13, 17 and 21, respectively. We will explain the details one step by one step. At step 13, choose the value of Q_9 to make $Q_9 + F(Q_{12}, Q_{11}, Q_{10}) + K_{13}$ be `0x00000000`. So Q_{13} will be equal to $X_{12} \lll 6$. Then at step 17, choose the value of X_7 to make $X_7 + F(Q_{16}, Q_{15}, Q_{14}) + K_{17}$ be `0x00000000`. So the value of Q_{17} will be equal to $Q_{13} \lll 7$ $(= X_{12} \lll 13)$. Then at step 21, choose the value of X_{10} to make $X_{10} + F(Q_{20}, Q_{19}, Q_{18}) + K_{21}$ be `0x00000000`. So the value of Q_{21} will be equal to $Q_{21} \lll 11$ $(= X_{12} \lll 24)$. Finally at step 25, $Q_{21} + X_{12}$ becomes $(X_{12} \lll 24) + X_{12}$. Following the example in Section 3.1, when fixing the value of $X_{12} + (X_{12} \lll 24)$ as `0xffffffff`, the freedom degree of X_{12} will be 2^8. We recall that our method will fix the values of Q_9, X_7 and X_{10}. Moreover, at step 25 since the value of $Q_{21} + X_{12}$ is fixed as `0xffffffff`, the values of both Q_{24} and Q_{25} will only depend on the value of X_3.

Remark. Each of the message words X_{13} and X_{15} are used twice in LC_1. We stress that it will not cause a contradiction in LC_1. X_{15} is used at steps 16 and

23. At step 16, the condition on X_{15} is that the value of $Q_{12}+X_{15}$ is fixed. But neither the value of Q_{12} nor the value of X_{15} are fixed at this step so far. Thus for any value of X_{15}, we can adaptively choose a value for Q_{12} without causing a contradiction at step 16. So we will first compute the value of X_{15} fixed at step 23, and then adaptively choose the value of Q_{12} at step 16. No contradiction will occur. Similarly for X_{13}, we will first compute its value fixed at step 19 and then adaptively choose the value of Q_{10} at step 14.

Summarization. Setting up LC_1 determines the values of the message words X_1, X_4, X_6, X_7, X_{10}, X_{13}, X_{14} and X_{15}, and the internal state words from Q_9 until Q_{23}. The values of Q_{24} and Q_{25} depend on the value of X_3. The freedom degree of X_{12} in LC_1 is 2^8.

4.2 The Details of LC_2

We will use X_0 to construct LC_2. This is exactly the example we explained in Section 3.2. We will only recall that the freedom degree of X_0 is 2^{32}.

4.3 Our Pseudo-preimage Attack Procedure

This section describes the procedure of our pseudo-preimage attacks on 47-step RIPEMD compression function.

1. Set up LC_1 in the left branch, which has been illustrated in Section 4.1. Set a randomly chosen value to X_3 and compute the values of Q_{24} and Q_{25}. The internal states $Q_9||Q_{12}||Q_{11}||Q_{10}$ at step 12 and $Q_{22}||Q_{25}||Q_{24}||Q_{23}$ at step 25 have been fixed.
2. Set randomly chosen values to X_2, X_5, X_8 and X_9.
3. Compute from step 12 in the backward direction in the left branch until the initial state. The values of Q_0, Q_{-1} and Q_{-2} are computed. And the value of $(Q_1 \ggg 11) - K_1 - F(Q_0, Q_{-1}, Q_{-2})$, denoted as C, is also computed.
4. Compute the value of Q'_1 as $(C + F(Q_0, Q_{-1}, Q_{-2}) + K'_1) \lll 11$ in the right branch . So the internal state at step 1 in the right branch is obtained.
5. Compute from step 2 in the forward direction in the right branch until step 12 to obtain the value of the internal state $Q'_9||Q'_{12}||Q'_{11}||Q'_{10}$.
6. For all the 2^8 candidate values of X_{12}, compute from step 13 until step 25 in the right branch. Memorize the values of $(X_{12}, Q'_{22}||Q'_{25}||Q'_{24}||Q'_{23})$ in a Table $\mathcal{T}$.
7. For all the 2^{32} candidate values of X_0,
   (a) Compute the value of Q_{-3}; $Q_{-3} = C - X_0$.
   (b) Compute from step 26 until step 47 in the left branch to obtain the value of $Q_{44}||Q_{47}||Q_{46}||Q_{45}$.
   (c) Compute the corresponding value of $Q'_{44}||Q'_{47}||Q'_{46}||Q'_{45}$ according to the given target digest.
   (d) Compute from step 47 until step 25 in the backward direction in the right branch to obtain the value of $Q'_{22}||Q'_{25}||Q'_{24}||Q'_{23}$.

(e) Match the obtained value of $Q'_{22}||Q'_{25}||Q'_{24}||Q'_{23}$ at STEP 7(d)[5] with the elements in Table $\mathcal{T}$. If it matches, a pseudo-preimage has been found.

8. If no pseudo-preimage is found, change the values of the message words in STEP 2, and repeat STEPs 3 – 7.

Recall the freedom degree of X_0 and X_{12} is 2^{32} and 2^8 respectively. So one execution of STEPs 6 and 7 will contribute to $2^{40(=8\times32)}$ pairs. In total 2^{128} pairs are necessary, therefore the execution from STEP 2 until STEP 7 will be repeated $2^{88(=128-40)}$ times. We stress that the freedom degree at STEP 2 is 2^{128}, which guarantees the success of the attack.

4.4 Complexity Evaluation

We count one 47-step compression function computation as a unit. Correspondingly one step function computation is regarded as $\frac{1}{94(=47\times2)}$ computation.

STEP 1. $\frac{13}{94}$ computation.
STEP 2. Negligible.
STEP 3. $2^{88} \times \frac{12}{94}$ computations.
STEP 4. $2^{88} \times \frac{1}{94}$ computations.
STEP 5. $2^{88} \times \frac{12}{94}$ computations.
STEP 6. $2^{88} \times 2^{8} \times \frac{13}{94}$ computations.
STEP 7(a). $2^{88} \times 2^{32} \times \frac{1}{94}$ computations.
STEP 7(b). $2^{88} \times 2^{32} \times \frac{22}{94}$ computations.
STEP 7(c). $2^{88} \times 2^{32} \times \frac{1}{94}$ computations.
STEP 7(d). $2^{88} \times 2^{32} \times \frac{22}{94}$ computations.
STEP 8. Negligible.

The overall complexity is less than 2^{119} computations. The dominant memory requirement is STEP 6. The memory requirement is less than $2^{10.5}$ words.

4.5 Impact on 47-Step RIPEMD Hash Function

Our pseudo-preimage attack on 47-step RIPEMD compression function can be converted to a second preimage attack on 47-step RIPEMD hash function following the method in Section 2.3. The complexity is $2^{124.5}$ compression function computations and the memory requirement is $2^{10.5}$ words. Our second preimage attack can succeed as long as the original message has more than 2 blocks. The complexity is fixed, and not related to the block length of the original message.

Comparison with generic attacks. Several generic second-preimage attacks have been published on the Merkle-Damgård hash functions [8,1]. The complexities of these generic attacks are related to the block length of the original message. Suppose that the original message has t blocks. Loosely speaking, the

[5] This paper refers to "STEP" as the steps of the attack procedure and refers to "step" as the step functions of hash functions.

complexity of generic attacks is around $2^{128}/(t - log_2 t)$. Thus as long as $t < 16$, our attack will be faster than these generic attacks.

We will present another generic second-preimage attack, which works efficiently on short original messages. We will construct an expandable message with variable block length from 1 until $t-1$ as follows:

1. Choose a random message block M_0, and compute $h_1 = CF(h_0, M_0)$.
2. For $i = 2, \ldots, t-1$, generate a pair of messages M_{i-1} and M'_{i-1} such that $h_i = CF(h_{i-1}, M_{i-1})$ $=CF(h_0, M'_{i-1})$ by the birthday attack.

This expandable message can produce messages, linking h_0 to h_{t-1}, with blocks varying from 1 until $t-1$. A 1-block message M'_{t-2} links h_0 to h_{t-1}. A $(t-1)$-block message $M_0||M_1||\cdots||M_{t-2}$ links h_0 to h_{t-1}. A s-block message $(1 < s < t-1)$ $M'_{t-1-s}||M_{t-s}\ ||\cdots||M_{t-2}$ links h_0 to h_{t-1}.

Finally we will exhaustively link h_{t-1} to any one of $(t-1)$ intermediate hash values of the original message for producing a second preimage. The total complexity will be $2^{128}/(t-1)$, when t is small. Compared with our generic attack, our second preimage attack is faster when the original message is no longer than 12 blocks.

5 Our Attacks on Intermediate 36-Step RIPEMD-128

This section will apply our attack strategy to evaluate preimage resistance of RIPEMD-128, and propose a preimage attack on intermediate 36-step RIPEMD-128, more precisely from steps 23 until 58.

We will use X_2 and X_{13} as the neutral message words. X_2 will construct a one-message-word local collision from step 32 until step 44 in the right branch, following the technique in Section 3.1. The overview of our pseudo-preimage attack is detailed in Figure 5.

We will briefly describe the one-message-word local collision in the right branch. We will set conditions on the internal state words to control the influence of modifying X_2. All the conditions are listed in Table 4. The influenced internal state words are only Q'_{32}, Q'_{36} and Q'_{40}. Similarly to our attack on RIPEMD, we will adjust the values of the message words X_2, X_3 and X_9 at steps 32, 36 and 40 to obtain the following relations: $Q'_{32} = X_2 \lll 11$ (at step 32); $Q'_{36} = Q'_{32} \lll 11$ (at step 36); and $Q'_{40} = Q'_{36} \lll 14$ (at step 40). Finally at step 44, the value of $X_2 + Q'_{40}$ is $X_2 + (X_2 \lll 36))$. As explained in Section 3.2, the freedom degree of X_2 to make $X_2 + (X_2 \lll 36)$ be `0xffffffff` should be 2^4. Luckily, each message word involved in this local collision is used only once, so we can trivially set up the local collision.

There are another two issues we will address. Firstly, we stress that there are enough freedom degree to carry out the pseudo-preimage attacks. More precisely the values of X_0, X_4, X_{10} and X_{12} can be freely chosen. Secondly, even though X_{15} is involved in the local collision, but its value is not fixed and can be chosen by the attacker. Thus the pseudo-preimage attack can be converted to a preimage attack on the hash function.

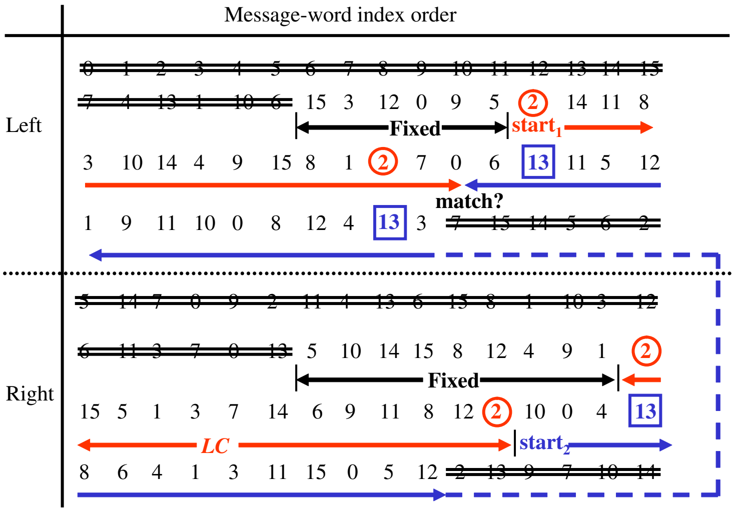


Fig. 5. Overview of Our Pseudo-Preimage Attacks on RIPEMD-128

Table 4. The Conditions for the Local Collision of Our Attack on RIPEMD-128

Step Index	i	$i+1$	$i+2$
$i = 33$	$Q'_{31} = Q'_{30}$	$Q'_{33} =$ `0x00000000`	$Q'_{34} =$ `0xffffffff`
$i = 37$	$Q'_{35} = Q'_{34}$	$Q'_{37} =$ `0x00000000`	$Q'_{38} =$ `0xffffffff`
$i = 41$	$Q'_{39} = Q'_{38}$	$Q'_{41} =$ `0x00000000`	$Q'_{42} =$ `0xffffffff`

The complexity of the pseudo-preimage attack is 2^{123} computations. The memory requirement is $2^{6.5}$ words. The pseudo-preimage attack can be converted into a preimage attack on step-reduced RIPEMD-128 following the method in Section 2.3. The complexity of preimage attack is $2^{126.5}$ computations.

6 Conclusion

This paper have proposed a pseudo-preimage and second preimage attacks on 47-step RIPEMD with complexities of 2^{119} and $2^{124.5}$ compression function computations, respectively. Moreover we also have proposed a pseudo-preimage and preimage attacks on intermediate 36-step RIPEMD-128 with complexities of 2^{123} and $2^{126.5}$ compression function computations, respectively.

References

1. Andreeva, E., Bouillaguet, C., Fouque, P.-A., Hoch, J.J., Kelsey, J., Shamir, A., Zimmer, S.: Second preimage attacks on dithered hash functions. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 270–288. Springer, Heidelberg (2008)

2. 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)
3. 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)
4. 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)
5. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A strengthened version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
6. Goos, G., Hartmanis, J., van Leeuwen, J. (eds.): Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040. LNCS, ch. RIPEMD, vol. 1007, pp. 69–111. Springer, Heidelberg (1995)
7. 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)
8. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
9. Klima, V.: Tunnels in hash functions: MD5 collisions within a minute. Cryptology ePrint Archive, Report 2006/105 (2006), http://eprint.iacr.org/2006/105
10. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
11. Ohtahara, C., Sasaki, Y., Shimoyama, T.: Preimage attacks on step-reduced RIPEMD-128 and RIPEMD-160. In: INSCRYPT (2010)
12. Saarinen, M.-J.O.: A meet-in-the-middle collision attack against the new FORK-256. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 10–17. Springer, Heidelberg (2007)
13. 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)
14. Sasaki, Y., Aoki, K.: Meet-in-the-middle preimage attacks on double-branch hash functions: Application to RIPEMD and others. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 214–231. Springer, Heidelberg (2009)
15. Wang, G., Wang, S.: Preimage attack on hash function RIPEMD. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 274–284. Springer, Heidelberg (2009)

A Parameters of RIPEMD-128

Table 5. Parameters of RIPEMD-128 Compression Function

The left branch		
First Round	j of X_j	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 , 13 , 14, 15
	K_i	`0x00000000`
	F	$Q_{i-1} \oplus Q_{i-2} \oplus Q_{i-3}$
	S_i	11, 14, 15, 12, 5, 8, 7, 9, 11, 13, 14, 15, 6, 7, 9, 8
Second Round	j of X_j	7, 4, 13, 1, 10, 6, 15, 3, 12, 0, 9, 5, 2, 14, 11, 8
	K_i	$K_i =$ `0x5A827999`
	F	$(Q_{i-1} \wedge Q_{i-2}) \vee (\neg Q_{i-1} \wedge Q_{i-3})$
	S_i	7, 6, 8, 13, 11, 9, 7, 15, 7, 12, 15, 9, 11, 7, 13, 12
Third Round	j of X_j	1, 9, 11, 10, 0, 8, 12, 4, 13, 3, 7, 15, 14, 5, 6, 2
	K_i	`0x6ED9EBA1`
	F	$(Q_{i-1} \vee \neg Q_{i-2}) \oplus Q_{i-3}$
	S_i	11, 13, 6, 7, 14, 9, 13, 15, 14, 8, 13, 6, 5, 12, 7, 5
Fourth round	j of X_j	1, 9, 11, 10, 0, 8, 12, 4, 13, 3, 7, 15, 14, 5, 6, 2
	K_i	`0x8F1BBCDC`
	F	$(Q_{i-1} \wedge Q_{i-3}) \vee (Q_{i-2} \wedge \neg Q_{i-3})$
	S_i	1, 9, 11, 10, 0, 8, 12, 4, 13, 3, 7, 15, 14, 5, 6, 2
The right branch		
First round	j of X_j	5, 14, 7, 0, 9, 2, 11, 4, 13, 6, 15, 8, 1, 10, 3, 12
	K'_i	`0x50A28BE6`
	F	$(Q'_{i-1} \wedge Q'_{i-3}) \vee (Q'_{i-2} \wedge \neg Q'_{i-3})$
	S'_i	8, 9, 9, 11, 13, 15, 15, 5, 7, 7, 8, 11, 14, 14, 12, 6
Second round	j of X_j	6, 11, 3, 7, 0, 13, 5, 10, 14, 15, 8, 12, 4, 9, 1, 2
	K'_i	`0x5C4DD124`
	F	$(Q'_{i-1} \vee \neg Q'_{i-2}) \oplus Q'_{i-3}$
	S'_i	9, 13, 15, 7, 12, 8, 9, 11, 7, 7, 12, 7, 6, 15, 13, 11
Third round	j of X_j	15, 5, 1, 3, 7, 14, 6, 9, 11, 8, 12, 2, 10, 0, 4, 13
	K'_i	`0x6D703EF3`
	F	$(Q'_{i-1} \wedge Q'_{i-2}) \vee (\neg Q'_{i-1} \wedge Q'_{i-3})$
	S'_i	9, 7, 15, 11, 8, 6, 6, 14, 12, 13, 5, 14, 13, 13, 7, 5
Fourth round	j of X_j	8, 6, 4, 1, 3, 11, 15, 0, 5, 12, 2, 13, 9, 7, 10, 14
	K'_i	`0x00000000`
	F	$Q'_{i-1} \oplus Q'_{i-2} \oplus Q'_{i-3}$
	S'_i	15, 5, 8, 11, 14, 14, 6, 14, 6, 9, 12, 9, 12, 5, 15, 8

MJH: A Faster Alternative to MDC-2

Jooyoung Lee[1] and Martijn Stam[2,⋆]

[1] The Attached Institute of Electronics and Telecommunications Research Institute, Daejeon, Korea
jlee05@ensec.re.kr

[2] Royal Holloway, University of London, Egham, United Kingdom
martijn.stam@rhul.ac.uk

Abstract. In this paper, we introduce a new class of double-block-length hash functions. In the ideal cipher model (for n-bit blocks), we prove that these hash functions, dubbed MJH, are provably collision resistant up to $O(2^{\frac{2n}{3}-\log n})$ queries in the iteration.

When based on n-bit key blockciphers, our construction provides better provable security than MDC-2, the only known construction of a rate-1/2 double-length hash function based on an n-bit key blockcipher with non-trivial provable security. Moreover, since key scheduling is performed only once per message block for MJH, our proposal significantly outperforms MDC-2 in efficiency.

When based on a $2n$-bit key blockcipher, we can use the extra n bits of key to increase the amount of payload accordingly. Thus we get a rate-1 hash function that is much faster than existing proposals, such as Tandem-DM, at the expense of (for the moment) reduced provable security.

1 Introduction

A cryptographic hash function takes a message of arbitrary length, and returns a bit string of fixed length. The most common way of hashing variable length messages is to iterate a fixed-size compression function (e.g. according to the Merkle–Damgård paradigm [10, 27]). The underlying compression function can either be constructed from scratch, or be built upon off-the-shelf cryptographic primitives such as blockciphers. Recently, blockcipher-based constructions have attracted renewed interest as many dedicated hash functions, including those most common in practical applications, have started to exhibit serious security weaknesses [2, 9, 24, 26, 35, 40, 41, 42]. By instantiating a blockcipher-based construction with an extensively studied (and fully trusted) blockcipher, one can conveniently transfer the trust in the existing blockcipher to the hash function[1]. This approach is particularly useful in highly constrained environments such as RFID systems, since a single implementation of a blockcipher can be used for

⋆ This author's research was performed while at LACAL, EPFL, Switzerland.

[1] Since our proof will be in the ideal cipher model, the usual caveats related to instantiating ideal primitives apply.

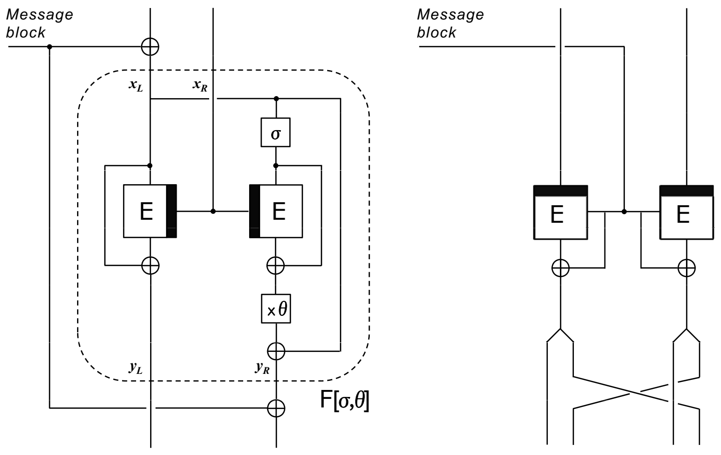
both a blockcipher and a hash function. Compared to blockciphers, most dedicated hash functions require significant amounts of state and the operations in their designs are not always as hardware friendly [6].

Compression functions based on blockciphers have been widely studied [4, 5, 14, 15, 16, 17, 20, 30, 31, 32, 33, 34, 36, 37, 38, 39]. The most common approach is to construct a $2n$-to-n bit compression function using a single call to an n-bit blockcipher. However, such a function, called a *single-block-length* (SBL) compression function, might be vulnerable to collision attacks due to its short output length. For example, one could successfully mount a birthday attack on a compression function based on AES-128 using approximately 2^{64} queries. This observation motivated substantial research on *double-block-length* (DBL) hash functions, where the output length is twice the block length of the underlying blockcipher(s).

An important distinction can be made on whether the underlying n-bit blockcipher has n-bit or $2n$-bit keys. Whereas for the latter scenario several proposals with good provable security are known, the construction of a double-length hash function based on an n-bit-key blockcipher remains elusive. Currently the only known candidate providing both efficiency and a reasonable level of provable security is MDC-2 [8, 28], which makes two calls to an n-bit key blockcipher to compress a single message block (thus its rate, the ratio of message blocks hashed per blockcipher calls, equals 1/2). In 2007, 20 years after its original proposal, Steinberger was the first to provide a non-trivial bound on the collision resistance of MDC-2 in the ideal cipher model [39]. In particular, he showed that an adversary asking fewer than $2^{3n/5-\epsilon}$ queries (for any fixed $\epsilon > 0$) has only a negligible chance of finding a collision. The best attack against MDC-2 however still requires $\Omega(2^{n-\log n})$ queries [19], leaving a considerable gap.

Our contribution. We propose a new construction, dubbed MJH, that significantly outperforms MDC-2 both in terms of efficiency and what can currently be proven about it. Figures 1(a) and 1(b) depict our proposed compression function and the MDC-2 compression function, respectively. A formal definition of MJH will follow in Section 3.1. From a high level, we first construct a $2n$-bit to $2n$-bit function by concatenating the output of two parallel blockcipher calls run in Davies–Meyer mode. Here we use Hirose's trick [17] of an involution without fixed points to achieve implicit domain separation so we can use the same blockcipher for both strains. The resulting, reasonably random looking function is subsequently used as primitive for the JH construction [43], creating a $3n$-bit to $2n$-bit compression function. Iterating it using the Merkle–Damgård transform, we obtain a hash function.

Efficiency comparison. Per message block, the MJH construction makes two calls to a blockcipher with the same key. In addition, there are four n-bit xors (since the last two xors on the right strain can be merged) and the operations σ and θ. For the involution σ it suffices to toggle a single bit. The nonzero constant θ can be efficiently implemented by an n-bit wise shift (corresponding to multiplication by x in polynomial representation of the field) followed by a conditional xor of a mask (corresponding to reduction modulo the minimal polynomial).



(a) MJH compression function (b) MDC-2 compression function

Fig. 1. The MJH compression function defined by an involution σ and a constant $\theta \in \mathbb{F}_{2^n} \backslash \mathbb{F}_2$ and the MDC-2 compression function. If the dotted box in (a) is regarded as a $2n$-bit permutation, it represents the JH compression function.

MDC-2 on the other hand has two calls to a blockcipher with *distinct* keys, but it only needs two n-bit xors and a swap to complete evaluation of the compression function. Depending on platform and implementation (and without AES instruction set), AES-128 key scheduling takes up around 25% to 35% (of the cycles) of a single call to the blockcipher. This translates into a speedup of about 10% to 15% of MJH over MDC-2 (the extra processing is not an issue here). We also note that due to recent related-key attacks [3] future blockciphers can be expected to have key scheduling algorithms that are even costlier than current ones; this would tilt the balance even further in our advantage.

On hardware we again obtain an advantage if the blockciphers is implemented twice (to increase throughput) since the key scheduling circuit can be shared. Our construction does need an extra feedforward in comparison to MDC-2 (2 vs. 1), which is a disadvantage [6].[2]

Security results. In the ideal cipher model, we can prove that our MJH hash functions are collision resistant up to $O(2^{\frac{2n}{3}-\log n})$ queries in the iteration. Like prior works dealing with collision resistance in the iteration [25,38], in the proof we consider a graph whose nodes correspond to chaining variables and where edges are drawn whenever an adversary has asked the "correct" queries to connect two chaining variables.

As a first observation, note that queries to the blockcipher are paired by the involution σ (just as in Hirose's scheme); we will call such a pair a *query-response cycle.* A challenge presents itself in that any query-response cycle typically adds

[2] While Figure 1(a) contains four feedforward wires, a simple inspection shows that in fact forwarding x_L and the message block suffices.

Table 1. Best known upper bounds on $\mathbf{Adv}^{\mathsf{coll}}_{\text{MDC-2}}(q)$ and $\mathbf{Adv}^{\mathsf{coll}}_{\text{MJH}}(q)$ for $n = 128$ (m_1 and m_2 are parameters appearing in Theorem 1)

q	$\mathbf{Adv}^{\mathsf{coll}}_{\text{MDC-2}}(q) \leq$	$\mathbf{Adv}^{\mathsf{coll}}_{\text{MJH}}(q) \leq$	(m_1, m_2)
2^{64}	7.57×10^{-7}	6.68×10^{-9}	$(2^{26}, 3)$
$2^{74.91}$	$1/2$	5.46×10^{-4}	$(2^{31}, 3)$
$2^{75.21}$	1	7.54×10^{-4}	$(2^{31}, 3)$
$2^{80.36}$	1	$1/2$	$(2^{36.31}, 3)$

2^{n+1} edges to the graph due to the JH structure. Our core observation is that *any pair* of query-response cycles can *only* be connected in four possible ways (the use of σ introduces the not-quite-uniqueness), which in turn allows us to put even more stringent bounds on the number of triplets of connected query-response cycles. This suffices to bound the probability of two largish components being connected in the graph; bounding the actual probability of constructing a collision still involves a considerable amount of additional case analysis.

Though far from optimal, our bound is the best one known for rate-1/2 n-bit key blockcipher-based DBL hash functions. As an example, a numerical comparison between Steinberger's bound for MDC-2 and ours for MJH in case of $n = 128$ is given in Table 1. As is the case for MDC-2, we believe that our bound can be improved considerably. We will comment on the expected increase in the amount of cases to get improved bounds in Appendix C.

Our analysis of MJH also opens up the possibility to get comparable bounds for JH itself. To the best of our knowledge, currently the best (published) bounds for JH appeared in [1] showing indifferentiability up to roughly $2^{n/6}$ queries (where n is the internal JH state size), whereas a straightforward mapping of our bounds would get closer to $2^{n/3}$. Unfortunately, for JH itself dealing with inverse (permutation) queries creates difficulties that we could avoid for MJH (by internally using Davies–Meyer, which would be meaningless for JH).

Variants. So far we have concentrated on building a DBL hash function based on a κ-bit key blockcipher with $\kappa = n$. However, our construction and its corresponding security proofs are sufficiently general to allow any κ-bit key blockcipher with $\kappa \geq n$. If $\kappa > n$, the surplus $n - \kappa$ bits can be used to carry extra message bits, so we compress κ bits of message per invocation of the MJH compression function (i.e. per two blockcipher calls). Thus, if we would use a $2n$-bit key blockcipher as underlying primitive, we obtain a rate-1 compression function natively supporting parallelism.

Related work. While many DBL compression functions of rate-1 have been proposed, unfortunately it turned out that a large class do not provide security in terms of collision resistance and preimage resistance beyond that already offered by single-block-length constructions [14,15,18]. This holds true both for constructions based on blockciphers with n-bit keys or $2n$-bit keys.

In the latter category, Lucks recently proposed the first DBL hash function of rate-1 with (almost) optimal security in the iteration [25,30]. Later, an alternative rate-1 secure DBL *compression* function was given by Stam [38] and generalized by Lee and Steinberger [22]. However, both constructions use full finite field multiplications, significantly degrading their efficiency.

Classical DBL compression functions of rate below 1 include MDC-2, MDC-4, Tandem-DM and Abreast-DM [8,20]. We have already discussed MDC-2 and the remarkable results obtained by Steinberger [39]. MDC-4 also uses an n-bit key blockcipher, but it is twice as inefficient (and curiously no proof of security is known for it).

Both Tandem-DM and Abreast-DM [8,20] are rate-1/2 hash functions based on a blockcipher with $2n$-bit key. The main challenge providing a proof is the fact that the same blockcipher is called twice, but recently Lee, Stam, and Steinberger [23] proved the security of Tandem-DM (correcting [11]). As in the case of MDC-2, the security bound obtained is parameterized: optimizing the parameter gives collision resistance of Tandem-DM up to the birthday bound. The collision resistance of Abreast-DM was independently proved in [12] and [21].

While the design of hash functions based on $2n$-bit key blockciphers is considerably easier than that based on n-bit key blockciphers, the former—while more robust with respect to for instance preimage resistance—are typically less efficient even at the same "rate". Indeed, a blockcipher with $2n$-bit keys is required to provide $2n$-bit security as opposed to n-bit security for the smaller n-bit key blockcipher. For example, AES-256 consists of 14 rounds, 4 rounds more than AES-128. So as a first rough estimate (ignoring key scheduling) one expects AES-256 to be about 40% slower than AES-128. (See also [7] for a comprehensive comparison of the software performance of various AES/Rijndael-based DBL hash functions). As an aside, the recent related-key attacks on AES-256 by Biryukov and Khovratovich [3], suggest that instantiating blockcipher-based hash functions with AES-256 might not guarantee a sufficient level of security.

2 Preliminaries

General notation. Let $\mathbb{F}_{2^n}$ denote a finite field of order 2^n. Throughout our work, we will identify $\mathbb{F}_{2^n}$ and $\{0,1\}^n$, assuming a fixed mapping between the two sets. For two bitstrings x and y, $x||y$ denotes the concatenation of x and y. For a bitstring $x \in \{0,1\}^{2n}$, x_L and x_R denote the unique n-bit strings such that $x = x_L||x_R$. For an event E, $\overline{\mathsf{E}}$ denotes its complement.

The ideal cipher model. For positive integers κ and n, let $BC(\kappa, n)$ be the set of all blockciphers with n-bit blocks and κ-bit keys. In the ideal cipher model, a (κ, n)-blockcipher E is chosen from $BC(\kappa, n)$ uniformly at random. It allows for two types of oracle queries $E(K, X)$ and $E^{-1}(K, Y)$ for $X, Y \in \{0,1\}^n$ and $K \in \{0,1\}^\kappa$. The response to an inverse query $E^{-1}(K, Y)$ is $X \in \{0,1\}^n$ such that $E(K, X) = Y$. Here, X, Y and K are called plaintext, ciphertext, and key, respectively. In this paper, we only consider $\kappa \geq n$ with an emphasis on the case $\kappa = n$.

Collision resistance. We review the definition of collision resistance *in the information-theoretic model.* Given a function $H = H[\mathcal{P}]$ and an IT adversary $\mathcal{A}$ both with oracle access to an ideal primitive $\mathcal{P}$, the collision resistance of H against $\mathcal{A}$ is estimated by the $\mathbf{Exp}_H^{\mathsf{coll}}\mathcal{A}$ experiment.

```
**Experiment** $\mathbf{Exp}_H^{\mathsf{coll}}(\mathcal{A})$
  $\mathcal{A}^{\mathcal{P}}$ updates $\mathcal{Q}$
  **if** $\exists\ M \neq M'$ and $u$ such that $u = H_{\mathcal{Q}}(M) = H_{\mathcal{Q}}(M')$ **then**
      output 1
  **else**
      output 0
```

This experiment records every query-response pair that $\mathcal{A}$ obtains by oracle queries into a *query history* $\mathcal{Q}$. We write $u = H_{\mathcal{Q}}(M)$ if $\mathcal{Q}$ contains all the query-response pairs required to compute $u = H(M)$. At the end of the experiment, $\mathcal{A}$ would like to find two distinct evaluations yielding a collision. The *collision-finding advantage* of $\mathcal{A}$ is defined to be

$$\mathbf{Adv}_H^{\mathsf{coll}}(\mathcal{A}) = \mathbf{Pr}\left[\mathbf{Exp}_H^{\mathsf{coll}}(\mathcal{A}) = 1\right].$$

The probability is taken over the random choice of $\mathcal{P}$ and $\mathcal{A}$'s coins (if any). For $q > 0$, we define $\mathbf{Adv}_H^{\mathsf{coll}}(q)$ as the maximum of $\mathbf{Adv}_H^{\mathsf{coll}}(\mathcal{A})$ over all adversaries $\mathcal{A}$ making at most q queries.

The Merkle–Damgård transform. For convenience, we recall the Merkle–Damgård transform as it will be applied to our (double-block-length) construction. Let $\mathsf{pad} : \{0,1\}^* \rightarrow \bigcup_{i=1}^{\infty}\{0,1\}^{\kappa i}$ be an injective padding. With this padding scheme and a predetermined constant $IV \in \{0,1\}^{2n}$, the *Merkle–Damgård transform* produces a variable-input-length function $MD[F] : \{0,1\}^* \rightarrow \{0,1\}^{2n}$ from a fixed-input-length function $F : \{0,1\}^{2n} \times \{0,1\}^{\kappa} \rightarrow \{0,1\}^{2n}$. For $M \in \{0,1\}^*$ such that $|\mathsf{pad}(M)| = l\kappa$, $MD[F](M)$ is computed as follows.

```
**Function** $MD[F](M)$
  $u[0] \leftarrow IV$
  Break $\mathsf{pad}(M)$ into $\kappa$-bit blocks, $\mathsf{pad}(M) = M[1]||\ldots||M[l]$
  **for** $i \leftarrow 1$ to $l$ **do**
      $u[i] \leftarrow F(u[i-1], M[i])$
  **return** $u[l]$
```

Since the padding is injective, we can simplify our collision analysis by assuming that the domain of a MD-iterated hash function is $\bigcup_{i=1}^{\infty}\{0,1\}^{\kappa i}$. Consequently, in the analysis of the MJH hash function we ignore the padding scheme.

The JH hash function. As said, our construction is motivated by JH, a second-round candidate of the SHA3 competition [29]. Given a noncompressing primitive $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$, let

$$\begin{aligned} \widetilde{F} : \{0,1\}^{2n} \times \{0,1\}^n &\longrightarrow \{0,1\}^{2n} \\ (u,z) &\longmapsto F\left(u + (z||0)\right) + (0||z). \end{aligned} \tag{1}$$

be its *JH transform* (a compression function). Then the *JH hash function* $JH[F]$ associated with F is defined to be $MD[\widetilde{F}]$, the Merkle–Damgård transform of $\widetilde{F}$. (In this paper, we only consider the "full" version of JH without any truncation of the final output.)

3 The MJH Hash Functions and Their Security

3.1 The MJH Construction

Let σ be an involution on $\{0,1\}^{2n}$ with no fixed point, and let $\theta \neq 0, 1$ be a constant in $\mathbb{F}_{2^n}$. Then (σ, θ) defines a noncompressing function $F[\sigma, \theta]$ based on an n-bit key blockcipher E as follows.

$$\begin{aligned} F[\sigma,\theta] : \{0,1\}^{2n} &\longrightarrow \{0,1\}^{2n} \\ (x_L||x_R) &\longmapsto (y_L||y_R), \end{aligned}$$

where $y_L = E(K,X) + X$ and $y_R = \theta\left(E(K,\sigma(X)) + \sigma(X)\right) + X$ for $(X||K) = (x_L||x_R)$. By applying the JH transform, we arrive at the compression function $\widetilde{F}[\sigma,\theta]$, as depicted in Figure 1(a). The compression function is subsequently fed to the Merkle–Damgård mechanism. Thus, the *MJH hash function* $H[\sigma,\theta]$ associated with (σ,θ) is defined by $MD[\widetilde{F}[\sigma,\theta]] = JH\left[F[\sigma,\theta]\right]$.

While the above variant (with $\kappa = n$) acts as the main proposal of our paper, we can actually prove a slight generalization where $\kappa \geq n$. Here we simply use the extra $\kappa - n$ bits of key to compress extra message bits. Since we want to ensure that both blockcipher calls are keyed identically, this allows $\kappa - n$ extra bits of message. Altogether, given a blockcipher $E \in BC(\kappa, n)$ we obtain a compression function $\{0,1\}^{2n} \times \{0,1\}^\kappa \to \{0,1\}^{2n}$ by

$$\begin{aligned} G[\sigma,\theta] : \{0,1\}^{2n} \times \{0,1\}^n \times \{0,1\}^{\kappa-n} &\longrightarrow \{0,1\}^{2n} \\ (u_L||u_R, z, z') &\longmapsto (v_L||v_R), \end{aligned}$$

where $v_L = E(K,X) + X$ and $v_R = \theta\left(E(K,\sigma(X)) + \sigma(X)\right) + X + z$ for $X = (u_L + z)$ and $K = (u_R||z')$. Note that if $\kappa = n$ we indeed get $G[\sigma,\theta] = \widetilde{F}[\sigma,\theta]$.

3.2 Query-Response Cycles and a Modified Adversary

Let $H = H[\sigma, \theta]$ be the MJH hash function defined by (σ, θ), and let $\mathcal{A}$ be an information-theoretic adversary with oracle access to E and E^{-1}. Note that $\mathcal{A}$ records a triple (X, K, Y) into the query history $\mathcal{Q}$ if $\mathcal{A}$ asks for $E(K, X)$ and gets back Y, or if it asks for $E^{-1}(K, Y)$ and gets back X. Since σ is an involution, it holds that $X_\sigma = \sigma(X)$ iff $X = \sigma(X_\sigma)$. In other words, σ induces a natural way to pair queries: for (X, K, Y) and (X_σ, K, Y_σ) in $\mathcal{Q}$ with $X_\sigma = \sigma(X)$ we call

$$\Delta = ((X, K, Y), (X_\sigma, K, Y_\sigma))$$

a *query-response cycle* (or simply a cycle) and the corresponding queries each other's *conjugates.*

We can now transform $\mathcal{A}$ into an adversary $\mathcal{B}$ that records its query history $\mathcal{Q}_\Delta$ in terms of query-response cycles as described in Figure 2. If $\mathcal{A}$ makes at most q queries, then the corresponding adversary $\mathcal{B}$ makes at most $2q$ queries, and records at most q query-response cycles. Since

$$\mathbf{Adv}_H^{\mathsf{coll}}(\mathcal{A}) \leq \mathbf{Adv}_H^{\mathsf{coll}}(\mathcal{B}),$$

it suffices to consider the security of H against a modified adversary that records exactly q query-response cycles.

Henceforth, the i-th query-response cycle is denoted

$$\Delta^i = \left((X^i, K^i, Y^i), (X_\sigma^i, K^i, Y_\sigma^i)\right).$$

If we want to distinguish between the two query-responses within a single cycle, we will refer with $i+$ to the first triple and with $i-$ to the second, where we assume that (X^i, K^i, Y^i) was obtained before $(X_\sigma^i, K^i, Y_\sigma^i)$. Additionally, we will write $[1, q] = \{1, \ldots, q\}$, $N = 2^n$ and $N' = N - 2q$ assuming $N' > N/2$.

3.3 Graph Representation of the Adversary's Endeavors

Each query-response cycle $\Delta^i = \left((X^i, K^i, Y^i), (X_\sigma^i, K^i, Y_\sigma^i)\right) \in \mathcal{Q}_\Delta$ determines two evaluations

$$F[\sigma, \theta] : (X^i || K^i) \longmapsto \left((X^i + Y^i) \,||\, (\theta(X_\sigma^i + Y_\sigma^i) + X^i)\right) \tag{2}$$

and

$$F[\sigma, \theta] : (X_\sigma^i || K^i) \longmapsto \left((X_\sigma^i + Y_\sigma^i) \,||\, (\theta(X^i + Y^i) + X_\sigma^i)\right). \tag{3}$$

Moreover, each evaluation $F[\sigma, \theta] : x \mapsto y$ determines evaluations

$$\widetilde{F}[\sigma, \theta] : (x_L + z, x_R, z) \mapsto (y_L, y_R + z),$$

for every $z \in \{0, 1\}^n$.

We now define a directed graph $\mathcal{G}$ on $\{0, 1\}^{2n}$ where the nodes correspond to chaining variables and edges are added depending on the evaluations of $\widetilde{F}[\sigma, \theta]$

```
Algorithm $\mathcal{B}^{E,E^{-1}}$
    $\mathcal{Q}_\Delta \leftarrow \emptyset$
    Run $\mathcal{A}$
    if $\mathcal{A}$ makes a fresh query for $E(K,X)$ then
        Make queries for $Y = E(K,X)$ and $Y_\sigma = E(K,X_\sigma)$ for $X_\sigma = \sigma(X)$
        $\mathcal{Q}_\Delta \leftarrow \mathcal{Q}_\Delta \cup \{\Delta\}$, where $\Delta = ((X,K,Y),(X_\sigma,K,Y_\sigma))$
        Return $Y$ to $\mathcal{A}$
    else if $\mathcal{A}$ makes a fresh query for $E^{-1}(K,Y)$ then
        Make queries for $X = E^{-1}(K,Y)$ and $Y_\sigma = E(K,X_\sigma)$ for $X_\sigma = \sigma(X)$
        $\mathcal{Q}_\Delta \leftarrow \mathcal{Q}_\Delta \cup \{\Delta\}$, where $\Delta = ((X,K,Y),(X_\sigma,K,Y_\sigma))$
        Return $X$ to $\mathcal{A}$
    else
        Return the response using query history $\mathcal{Q}_\Delta$
```

Fig. 2. The modified adversary $\mathcal{B}$. A query is called "fresh" if its response cannot be obtained from $\mathcal{B}$'s query history.

the adversary can make given the available query-response cycles. In other words, a directed edge from u to v labelled i is added to $\mathcal{G}$ when the i-th query-response cycle determines an evaluation $\widetilde{F}[\sigma,\theta](u,z) = v$ for some $z \in \{0,1\}^n$. Such a connection is denoted by $u \xrightarrow{i} v$. More specifically, we write $u \xrightarrow{i+} v$ (resp. $u \xrightarrow{i-} v$) when the evaluation of $F[\sigma,\theta]$ is obtained by (2) (resp. (3)).

3.4 Main Handle and Intuition

In this section, we present one formal lemma limiting the number of chains of length two in the graph $\mathcal{G}$. This relatively clean lemma is the main handle for our proof of collision resistance, as we will explain after the lemma. We hope this helps to develop intuition for the formal proof of collision resistance and the more involved lemmas on the graph $\mathcal{G}$ that proceed it.

Lemma 1. *Each ordered pair $(i,j) \in [1,q]^2$ determines four (not necessarily distinct) triplets (u,v,w) of nodes in $\mathcal{G}$ such that $u \xrightarrow{i} v \xrightarrow{j} w$ as follows.*

$$u = \Big((x_L^i + y_R^i + x_R^j) \,||\, x_R^i\Big),\ v = \Big(y_L^i \,||\, x_R^j\Big),\ w = \Big(y_L^j \,||\, (y_L^i + x_L^j + y_R^j)\Big),$$

where

$$(x_L^i, x_R^i, y_L^i, y_R^i) \in \{(X^i, K^i, X^i + Y^i, \theta(X_\sigma^i + Y_\sigma^i) + X^i),\\ (X_\sigma^i, K^i, X_\sigma^i + Y_\sigma^i, \theta(X^i + Y^i) + X_\sigma^i)\},$$

$$(x_L^j, x_R^j, y_L^j, y_R^j) \in \{(X^j, K^j, X^j + Y^j, \theta(X_\sigma^j + Y_\sigma^j) + X^j),\\ (X_\sigma^j, K^j, X_\sigma^j + Y_\sigma^j, \theta(X^j + Y^j) + X_\sigma^j)\},$$

for $\Delta^i = \big((X^i,K^i,Y^i),(X_\sigma^i,K^i,Y_\sigma^i)\big)$ and $\Delta^j = \big((X^j,K^j,Y^j),(X_\sigma^j,K^j,Y_\sigma^j)\big)$.

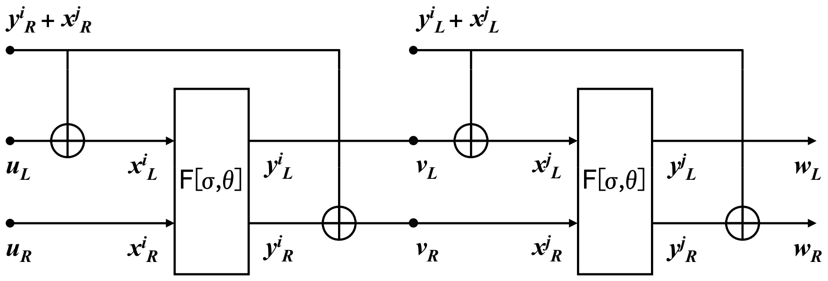


Fig. 3. Path determined by a pair of query-response cycles

Proof. The proof is straightforward as shown in Figure 3. □

Intuitively, when an adversary obtains a triplet $u \xrightarrow{i} v \xrightarrow{j} w$, he has full control over u_R and v_R (corresponding to the keys of the blockcipher calls for cycle i and j, respectively), yet little to no control over the remaining variables u_L, v_L, w_L, and w_R. Suppose these values are truly random, then the probability of many of them having the same w_R is small. Indeed, as a rule of thumb after q query-response cycles one would expect each possible value to occur roughly $4q^2/N$ times (the 4 due to σ).

This already allows bounding the number of quadruplets $u \xrightarrow{i} v \xrightarrow{j} w \xrightarrow{j'} w'$ since the key of the blockcipher used for cycle j' needs to correspond to w_R: any j' thus connects to about $8q^2/N$ triplets (again an extra factor 2 due to σ) for a total of roughly $8q^3/N$ quadruplets.

With these tentative bounds in hand, we can also look at the probability that with a single query-response cycle a triplet is connected to a quadruplet. Again, the cycle itself determines the w_R of the triplet and results in $8q^2/N$ new quadruplets with more or less random new end points. For the desired connection the end point of the new quadruplet should coincide with one of the $8q^3/N$ known start points of the existing quadruplets. This happens with probability $64q^5/N^4$ (assuming each new end point has probability $1/N^2$ of occurring). Since there are q cycles in total, a union bound shows that the event "connecting a triplet with a quadruplet" ever happening can be (informally) upper bounded by $64q^6/N^4$.

If we accept that an adversary needs to ask $\Omega(2^{2n/3})$ to cause the event above, we can still wonder whether collisions might be found faster. However, it turns out that if an adversary wants to find a collision, it needs to cause an event similar to the one above (where two larger components are merged), or else it needs to cause a collision "locally" (for instance $u \xrightarrow{i} v \xleftarrow{i} w$ with $u \neq w$). The possible ways to create a local collision can be easily enumerated and successfully bounded (and typically $\Omega(2^n)$ queries are needed to cause such a local event).

In the next section we will provide a formal security proof making the intuition above rigorous. Lemma 2 deals with the distribution of w_R in triplets (bounding the probability of exceeding some threshold) and Lemma 3 bounds various other distributions (all related to configurations involving a pair of query-response cycles only). Lemma 4 subsequently bounds the number of quadruplets and similar configurations, conditional on the thresholds from Lemmas 2 and 3 not being violated. Finally, Lemma 5 exhausts all the possible cases (events of which at least one needs to occur to cause a collision) which allows us to conclude with Theorem 1 where we bound the probabilities of the events for all the cases involved. Since the theorem statement is not very transparent, we provide Corollary 1 demonstrating its asymptotic behavior. Note that for the lemmas and the theorem we occasionally allowed ourselves some slack if this would not affect the targeted asymptotic behavior (of collision resistance up to $O(2^{\frac{2n}{3}-\log n})$ queries).

While our proof shares some obvious similarities with Steinberger's MDC-2 proof (lots of case-analysis and the use of 2-chains), there are notable differences. For MDC-2 after q queries one can roughly evaluate the compression function for q^2 values, whereas for MJH it allows $q2^n$ evaluations. It is only when we consider 2-chains that we bring this down to $O(q^2)$. In our case analysis we subsequently need to consider longer chains, whereas Steinberger does not. The *type* of case analysis is also rather distinct: Steinberger only looks at the place where the collision happens, whereas we take a more global approach (making it closer to a refinement of the BRS Type-II proof [38]).

4 Formal Collision Resistance Proof

4.1 Bounding Small Components

The following lemma, whose proof is given in Appendix A, states that if we look at the set of all nodes w that are either IV, or a child of IV, or reachable by a chain of length two, then the probability of many of them having the same w_R is small (by appropriately setting parameters m_1 and m_2). This is relevant since later we would like to upper bound the number of "reachable" nodes that the i-th query-response cycle can connect to. Note that posing the i-th query specifies the key used, corresponding to the w_R part.

Lemma 2. *For each $z \in \{0,1\}^n$ and $i \in [1,q]$, let*

$$\mathcal{R}^i(z) = \{IV\} \cup \\ \left\{ w \in \{0,1\}^{2n} : w_R = z \textit{ and } (\exists\ IV \xrightarrow{i_1} w \textit{ or } \exists\ u \xrightarrow{i_2} v \xrightarrow{i_1} w \textit{ for } i_1, i_2 \le i) \right\}.$$

For integer parameters $m_1, m_2 > 0$, let $m = 2m_1m_2$ and $M = 8m + 1$. Define event

$$\mathsf{Mult}(M) \Leftrightarrow \max_{z \in \{0,1\}^n} |\mathcal{R}^q(z)| > M.$$

Then we have

$$\mathbf{Pr}\left[\mathsf{Mult}(M)\right] \le 2N\left(\frac{12q}{mN'}\right)^m + N\left(\frac{12q^2}{mN'}\right)^m + N\left(\frac{6q}{m_2N'}\right)^{m_2} + N\left(\frac{12q^2}{m_1N'}\right)^{m_1}.$$

Now we would like to upper bound the number of nodes that are reachable by "3-chains". We will show in Lemma 4 that the number of such nodes is small without the occurrence of certain events. The following lemma defines these events and states that they occur only with small probability.

Lemma 3. *For integer parameters $M, m > 0$, define events*

$$\begin{aligned}
\mathsf{E}_1(M) &\Leftrightarrow \left|\left\{(i,j) \in [1,q]^2 : i \neq j \wedge \exists u \xrightarrow{i} v \xrightarrow{j} w \text{ such that } u_L = w_L\right\}\right| > M,\\
\mathsf{E}_2(M) &\Leftrightarrow \left|\left\{(i,j) \in [1,q]^2 : i \neq j \wedge \exists u \xrightarrow{i} w \xleftarrow{j} v\right\}\right| > M,\\
\mathsf{E}_3(m) &\Leftrightarrow \left|\left\{(i,j) \in [1,q]^2 : i \neq j \wedge \exists u \xrightarrow{i} w \xleftarrow{j} v \text{ such that } u_L = v_L\right\}\right| > m,\\
\mathsf{E}_4(4m^2) &\Leftrightarrow \max_{w \in \{0,1\}^{2n}} \left|\left\{u \in \{0,1\}^{2n} : \exists\ u \xrightarrow{i} v \xrightarrow{j} w\right\}\right| > 4m^2.
\end{aligned}$$

Then we have

$$\mathbf{Pr}\left[\mathsf{E}_1(M)\right] \leq \frac{4q^2}{MN'}\ , \qquad \mathbf{Pr}\left[\mathsf{E}_2(M)\right] \leq \frac{4q^2}{MN'}\ ,$$
$$\mathbf{Pr}\left[\mathsf{E}_3(m)\right] \leq \frac{4q^2}{m(N')^2}\ , \qquad \mathbf{Pr}\left[\mathsf{E}_4(4m^2)\right] \leq N\left(\frac{6q}{mN'}\right)^m .$$

Proof. Fix a pair $(i,j) \in [1,q]^2$ such that $i \neq j$. If there exists $u \xrightarrow{i} v \xrightarrow{j} w$ such that $u_L = w_L$, then we have

$$u_L + w_L = (x_L^i + y_R^i + x_R^j) + y_L^j = 0,$$

where $x_L^i + y_R^i \in \{X^i + Y^i, X_\sigma^i + Y_\sigma^i\}$ and $x_R^j + y_L^j \in \{K^j + X^j + Y^j, K^j + X_\sigma^j + Y_\sigma^j\}$.

If $S_{i,j}$ denotes the event that the above equation holds, then we have $\mathbf{Pr}\left[S_{i,j}\right] \leq 4/N'$. If

$$a(\mathcal{Q}) = \left|\left\{(i,j) \in [1,q]^2 : i \neq j \wedge \exists u \xrightarrow{i} v \xrightarrow{j} w \text{ such that } u_L = w_L\right\}\right|,$$

then we have

$$\mathbb{E}\left[a(\mathcal{Q})\right] = \sum_{(i,j) \in [1,q]^2, i \neq j} \mathbb{E}\left[S_{i,j}\right] \leq \frac{4q^2}{N'}\ .$$

Using Markov's inequality, we have

$$\mathbf{Pr}\left[\mathsf{E}_1(M)\right] = \mathbf{Pr}\left[a(\mathcal{Q}) > M)\right] \leq \frac{4q^2}{MN'}\ ,$$

for any positive number M. The second and the third inequality are proved similarly.

In order to prove the final inequality, define event

$$\mathsf{Ex}(m) \Leftrightarrow \max_{z \in \{0,1\}^n} \left|\left\{j \in [1,q] : y_L^j = z\right\}\right| > m,$$

where $y_L^j \in \{X^j + Y^j, X_\sigma^j + Y_\sigma^j\}$. Then we have

$$\mathbf{Pr}\left[\mathsf{Ex}(m)\right] \le N\binom{q}{m}\left(\frac{2}{N'}\right)^m \le N\left(\frac{6q}{mN'}\right)^m .$$

Fix $w = w^* \in \{0,1\}^{2n}$. If there exists an edge $v \xrightarrow{j} w^*$, then it should be the case that $y_L^j = w_L^*$. Each such index j determines at most two values of the starting node v. Therefore, without the occurrence of event $\mathsf{Ex}(m)$, the number of nodes v such that there exists an edge $v \xrightarrow{j} w^*$ for some j is at most $2m$. Again, for each such node v, there are at most $2m$ nodes u for which there exists an edge $u \xrightarrow{i} v$ (for some i). To summarize, without the occurrence of event $\mathsf{Ex}(m)$, there are at most $4m^2$ nodes u such that there exists a path $u \xrightarrow{i} v \xrightarrow{j} w^*$ for some i and j. This means $\overline{\mathsf{Ex}(m)} \subset \overline{\mathsf{E}_4(4m^2)}$ or

$$\mathbf{Pr}\left[\mathsf{E}_4(4m^2)\right] \le \mathbf{Pr}\left[\mathsf{Ex}(m)\right] \le N\left(\frac{6q}{mN'}\right)^m . \qquad \square$$

Lemma 4. *For each $i \in [1,q]$, let*

$$\begin{aligned}\mathcal{S}_1^i = \{w \in \{0,1\}^{2n} : (w = IV) \vee (\exists\ IV \xrightarrow{i_1} w)\\ \vee (\exists\ IV \xrightarrow{i_2} v \xrightarrow{i_1} w) \vee (\exists\ u' \xrightarrow{i_3} u \xrightarrow{i_2} v \xrightarrow{i_1} w) \textit{ for } i_1, i_2, i_3 \le i\},\end{aligned}$$

$$\begin{aligned}\mathcal{S}_2^i = \{w \in \{0,1\}^{2n} : (\exists\ IV \xleftarrow{i_1} w) \vee (\exists\ IV \xrightarrow{i_2} v \xleftarrow{i_1} w)\\ \vee (\exists\ u' \xrightarrow{i_3} u \xrightarrow{i_2} v \xleftarrow{i_1} w) \textit{ for } i_1 \ne i_2, i_3 \le i\},\end{aligned}$$

$$\begin{aligned}\mathcal{S}_3^i = \{w \in \{0,1\}^{2n} : (\exists\ IV \xleftarrow{i_2} v \xleftarrow{i_1} w)\\ \vee (\exists\ u' \xrightarrow{i_3} u \xleftarrow{i_2} v \xleftarrow{i_1} w) \textit{ for } i_1, i_2 \ne i_3 \le i\},\end{aligned}$$

$$\mathcal{S}_4^i = \left\{w \in \{0,1\}^{2n} : \exists\ u' \xleftarrow{i_3} u \xleftarrow{i_2} v \xleftarrow{i_1} w \textit{ for } i_1, i_2, i_3 \le i\right\}.$$

Then for integer parameters $M, m > 0$, we have the following properties.

1. *Without the occurrence of* $\mathsf{Mult}(M)$, $\left|\mathcal{S}_1^i\right| \le |\mathcal{S}_1^q| \le 2qM + 1$.
2. *Without the occurrence of* $\mathsf{E}_2(M)$, $\left|\mathcal{S}_2^i\right| \le |\mathcal{S}_2^q| \le 8qM + 2q + 4M$.
3. *Without the occurrence of* $\mathsf{E}_2(M) \cup \mathsf{E}_4(m)$, $\left|\mathcal{S}_3^i\right| \le |\mathcal{S}_3^q| \le 4qM + m$.
4. *Without the occurrence of* $\mathsf{Mult}(M) \cup \mathsf{E}_4(m)$, $\left|\mathcal{S}_4^i\right| \le |\mathcal{S}_4^q| \le qMm$.

Proof. Since it is obvious that $\left|\mathcal{S}_\alpha^i\right| \le |\mathcal{S}_\alpha^q|$ for $\alpha = 1,2,3,4$, we will prove the properties for $i = q$.

Property 1: Fix $i_1 \in [1,q]$. Then there exist at most M nodes $v \in \{0,1\}^{2n}$ such that $v \in \mathcal{R}^q(K^{i_1})$. Since v and i_1 determine two nodes w such that $v \xrightarrow{i_1} w$, we have $|\mathcal{S}_1^q| \le 2qM + 1$.

Property 2: There exist at most M pairs $(i_2, i_1) \in [1,q]^2$ such that $i_1 \neq i_2$ and "$\overset{i_2}{\rightarrow} v \overset{i_1}{\leftarrow}$" for some v. Therefore, we have at most $4M$ nodes w such that $IV \overset{i_2}{\rightarrow} v \overset{i_1}{\leftarrow} w$, and at most $8qM$ nodes w such that $u' \overset{i_3}{\rightarrow} u \overset{i_2}{\rightarrow} v \overset{i_1}{\leftarrow} w$ for some $i_3 \in [1,q]$. Since the number of nodes w such that $IV \overset{i_1}{\leftarrow} w$ for some $i_1 \in [1,q]$ is not greater than $2q$, we have $|\mathcal{S}_2^q| \leq 8qM + 2q + 4M$.

Property 3: Without the occurrence of $\mathsf{E}_4(m)$, we have at most m nodes w such that $IV \overset{i_2}{\leftarrow} v \overset{i_1}{\leftarrow} w$ for some $i_1, i_2 \in [1,q]$. Without the occurrence of $\mathsf{E}_2(M)$, we have at most M pairs $(i_3, i_2) \in [1,q]^2$ such that $i_2 \neq i_3$ and "$\overset{i_3}{\rightarrow} u \overset{i_2}{\leftarrow}$" for some u. Each $i_1 \in [1,q]$ is combined with such a pair (i_3, i_2), yielding 4 nodes w such that $u' \overset{i_3}{\rightarrow} u \overset{i_2}{\leftarrow} v \overset{i_1}{\leftarrow} w$. Therefore we have $|\mathcal{S}_3^q| \leq 4qM + m$.

Property 4: Fix $i_3 \in [1,q]$. Without the occurrence of $\mathsf{Mult}(M)$, we have at most M nodes $u \in \{0,1\}^{2n}$ such that $u \in \mathcal{R}^q(K^{i_3})$. For each of such nodes u, there exists at most m nodes w such that $u \overset{i_2}{\leftarrow} v \overset{i_1}{\leftarrow} w$ without the occurrence of $\mathsf{E}_4(m)$. Therefore we have $|\mathcal{S}_4^q| \leq qMm$. □

4.2 Decomposing a Collision

Let Coll denote the event that $\mathcal{B}$ makes a collision of H. We can decompose the event Coll as follows.

Lemma 5. *Let C_α, $\alpha = 1, \ldots, 12$, be events defined as follows.*

$$\begin{aligned}
\mathsf{C}_1 &: \exists\ u \overset{i}{\rightarrow} v \overset{i}{\leftarrow} w \textit{ such that } u \neq w,\\
\mathsf{C}_2 &: \exists\ u \overset{i}{\rightarrow} v \overset{j}{\leftarrow} u \textit{ s.t. } j < i,\\
\mathsf{C}_3 &: \exists\ u \overset{i+}{\rightarrow} v \overset{i-}{\leftarrow} u,\\
\mathsf{C}_4 &: \exists\ u \overset{i}{\rightarrow} v \overset{i}{\rightarrow} w \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i),\\
\mathsf{C}_5 &: \exists\ u \overset{i}{\rightarrow} v \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } v \in \mathcal{S}_1^{i-1},\\
\mathsf{C}_6 &: \exists\ u \overset{i}{\rightarrow} v \overset{j}{\leftarrow} w \overset{j'}{\leftarrow} w' \overset{i}{\leftarrow} w'', \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } j \neq j' < i,\\
\mathsf{C}_7 &: \exists\ u \overset{i}{\rightarrow} v \overset{j}{\leftarrow} w \overset{i}{\leftarrow} w' \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } j < i,\\
\mathsf{C}_8 &: \exists\ u \overset{i}{\rightarrow} v \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } v \in \mathcal{S}_2^{i-1},\\
\mathsf{C}_9 &: \exists\ u \overset{i}{\rightarrow} v \overset{j}{\rightarrow} w \overset{j'}{\leftarrow} v' \overset{i}{\leftarrow} v'', \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } j \neq j' < i,\\
\mathsf{C}_{10} &: \exists\ u \overset{i}{\rightarrow} v \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } v \in \mathcal{S}_3^{i-1},\\
\mathsf{C}_{11} &: \exists\ u \overset{i}{\rightarrow} v \overset{j}{\rightarrow} w \overset{i}{\rightarrow} w' \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } j < i,\\
\mathsf{C}_{12} &: \exists\ u \overset{i}{\rightarrow} v \textit{ s.t. } u \in \mathcal{R}^{i-1}(K^i) \textit{ and } v \in \mathcal{S}_4^{i-1}.
\end{aligned}$$

Then we have $\mathsf{Coll} \subset \bigcup_{\alpha=1}^{12} \mathsf{C}_\alpha$.

Proof. We will show that $\mathsf{Coll} \cap \overline{(\mathsf{C}_1 \cup \mathsf{C}_2 \cup \mathsf{C}_3 \cup \mathsf{C}_4)} \subset \bigcup_{\alpha=5}^{12} \mathsf{C}_\alpha$. Suppose that the i^*-th query-response cycle of $\mathcal{B}$ completes a collision of H on a certain value w. Then, for some nonnegative integers s and t, there would exist two paths

$$\begin{aligned} \mathsf{P}_1: &\quad IV \overset{\alpha_1}{\to} u[1] \overset{\alpha_2}{\to} u[2] \overset{\alpha_3}{\to} \cdots \overset{\alpha_{s-1}}{\to} u[s-1] \overset{\alpha_s}{\to} w, \\ \mathsf{P}_2: &\quad IV \overset{\beta_1}{\to} v[1] \overset{\beta_2}{\to} v[2] \overset{\beta_3}{\to} \cdots \overset{\beta_{t-1}}{\to} v[t-1] \overset{\beta_t}{\to} w, \end{aligned}$$

such that $\alpha_1, \ldots, \alpha_s, \beta_1, \ldots, \beta_t \le i^*$. Excluding events C_1, C_2, C_3 and C_4, we can assume that

1. $u[s-1] \neq v[t-1]$ (i.e., w is an earliest possible collision),
2. P_1 contains an i^*-labelled edge,
3. $\beta_t < i^*$.

There might be several edges labelled i^* in P_1. Among such edges, focus on the "furthest" edge $u \overset{i^*}{\to} v$ from IV such that u is contained in $\mathcal{R}^{i^*-1}(K^{i^*})$. Then we can make the following observations.

1. If v is the colliding node, then we have one of the configurations in C_5, C_6 and C_7.
2. If v is one-edge away from the colliding node, then we have one of the configurations in C_8 and C_9.
3. If v is two-edges away from the colliding node, then we have one of the configurations in C_{10} and C_{11}.
4. Otherwise, we have one of the configurations in C_{11} and C_{12}.

These observations complete the proof. □

4.3 Putting the Pieces Together

Now we are ready to prove the following theorem.

Theorem 1. *Suppose that a modified adversary $\mathcal{B}$ records q query-response cycles. Let $m = 2m_1m_2$ and $M = 8m + 1$ for any integer parameters $m_1, m_2 > 0$. Then,*

$$\begin{aligned} &\mathbf{Adv}_H^{\mathsf{coll}}(\mathcal{B}) \le 2N\left(\frac{12q}{mN'}\right)^m + N\left(\frac{12q^2}{mN'}\right)^m + 2N\left(\frac{6q}{m_2N'}\right)^{m_2} + N\left(\frac{12q^2}{m_1N'}\right)^{m_1} \\ &+ \frac{8q^2}{MN'} + \frac{4q^2}{m_2(N')^2} + \frac{1}{N'}\left(2qMm_2 + 18qM + 2q\right) \\ &+ \frac{1}{(N')^2}\left(8q^3 + 8q^2M^2m_2^2 + 28q^2M^2 + 16q^2M + 2q^2 + 16qM^2 + 8qMm_2^2 + 2qM\right) \\ &:= \epsilon_N(q, m_1, m_2). \end{aligned}$$

Proof. Let $\mathsf{D}_\alpha = \mathsf{C}_\alpha \cap \overline{(\mathsf{Mult}(M) \cup \mathsf{E}_1(M) \cup \mathsf{E}_2(M) \cup \mathsf{E}_3(m_2) \cup \mathsf{E}_4(4m_2^2))}$ for $\alpha = 1, \dots, 12$. Then, by Lemma 5,

$$\begin{aligned}\mathbf{Adv}_H^{\mathsf{coll}}(\mathcal{B}) = \mathbf{Pr}[\mathsf{Coll}] \le \sum_{\alpha=1}^{12} \mathbf{Pr}[\mathsf{C}_\alpha] \le \mathbf{Pr}[\mathsf{Mult}(M)] + \mathbf{Pr}[\mathsf{E}_1(M)] \\ + \mathbf{Pr}[\mathsf{E}_2(M)] + \mathbf{Pr}[\mathsf{E}_3(m_2)] + \mathbf{Pr}\left[\mathsf{E}_4(4m_2^2)\right] + \sum_{\alpha=1}^{12} \mathbf{Pr}[\mathsf{D}_\alpha]. \quad (4)\end{aligned}$$

Now we can prove that

$$\begin{aligned}
&\mathbf{Pr}[\mathsf{D}_1] \le \frac{q}{N'}, && \mathbf{Pr}[\mathsf{D}_2] \le \frac{2q^2}{(N')^2},\\
&\mathbf{Pr}[\mathsf{D}_3] \le \frac{q}{N'}, && \mathbf{Pr}[\mathsf{D}_4] \le \frac{2qM}{N'},\\
&\mathbf{Pr}[\mathsf{D}_5] \le \frac{2qM(2qM+1)}{(N')^2}, && \mathbf{Pr}[\mathsf{D}_6] \le \frac{8q(q^2+M^2)}{(N')^2},\\
&\mathbf{Pr}[\mathsf{D}_7] \le \frac{8q^2M}{(N')^2}, && \mathbf{Pr}[\mathsf{D}_8] \le \frac{2qM(8qM+2q+4M)}{(N')^2},\\
&\mathbf{Pr}[\mathsf{D}_9] \le \frac{2qM(m_2+8)}{N'}, && \mathbf{Pr}[\mathsf{D}_{10}] \le \frac{2qM(4qM+4m_2^2)}{(N')^2},\\
&\mathbf{Pr}[\mathsf{D}_{11}] \le \frac{4q^2M}{(N')^2}, && \mathbf{Pr}[\mathsf{D}_{12}] \le \frac{8q^2M^2m_2^2}{(N')^2}.
\end{aligned}$$

It is straightforward to prove each inequality. Here we only upper bound $\mathbf{Pr}[\mathsf{D}_5]$ as one of the dominating terms. The full analysis is given in Appendix B.

In order to upper bound $\mathbf{Pr}[\mathsf{D}_5]$, we fix $i \in [1, q]$, $u \in \mathcal{R}^{i-1}(K^i)$ and $v \in \mathcal{S}_1^{i-1}$. Then the probability that the i-th query-response cycle Δ^i completes the configuration C_5 with an edge from u to v is at most $2/(N')^2$. Since $\left|\mathcal{R}^{i-1}(K^i)\right| \le M$ (without the occurrence of $\mathsf{Mult}(M)$) and $\left|\mathcal{S}_1^{i-1}\right| \le 2qM+1$ by Lemma 4, we have $\mathbf{Pr}[\mathsf{D}_5] \le 2qM(2qM+1)/(N')^2$.

Now by combining the above inequalities with (4), Lemma 2 and Lemma 3, we obtain the theorem. □

Corollary 1. *Let $N = 2^n$, $q = \frac{N^{2/3}}{n}$, $m_1 = N^{1/3}$ and $m_2 = 3$. Then we have*

$$\lim_{n\to\infty} \epsilon_N(q, m_1, m_2) = 0.$$

Proof. The proof is straightforward since

$$\epsilon_N(q, m_1, m_2) = \epsilon_N\left(\frac{N^{2/3}}{n}, N^{1/3}, 3\right) = O\left(\frac{qM}{N}\right) = O\left(\frac{1}{n}\right).$$

References

1. Bhattacharyya, R., Mandal, A., Nandi, M.: Security analysis of the mode of JH hash function. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 168–191. Springer, Heidelberg (2010)
2. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and reduced SHA-1. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)
3. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
4. 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)
5. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function construction from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–325. Springer, Heidelberg (2002)
6. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash functions and RFID tags: mind the gap. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer, Heidelberg (2008)
7. Bos, J., Özen, O.: Multi-block length hashing using the AES instruction set. IACR ePrint Archive 2010/576 (2010)
8. Brachtl, B., Coppersmith, D., Heyden, M., Matyas, S., Meyer, C., Oseas, J., Pilpel, S., Schilling, M.: Data authentication using modification detection codes based on a public one-way encryption function. US Patent #4,908,861. Awarded March 13, 1990 (filed August 28, 1987)
9. 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)
10. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
11. E. Fleischmann, M. Gorski and S. Lucks. On the security of Tandem-DM. FSE 2009, LNCS 5665, pp. 85–105, Springer, Heidelberg (2009).
12. Fleischmann, E., Gorski, M., Lucks, S.: Security of cyclic double block length hash functions. In: Parker, M.G. (ed.) IMACC 2009. LNCS, vol. 5921, pp. 153–175. Springer, Heidelberg (2009)
13. Gladman, B.: Implementation experience with AES candidate algorithms. In: Second AES Conference (1999)
14. Hattori, M., Hirose, S., Yoshida, S.: Analysis of double block length hash functions. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 290–302. Springer, Heidelberg (2003)
15. Hirose, S.: A security analysis of double-block-length hash functions with the rate 1. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E89-A(10), 2575–2582 (2006)
16. Hirose, S.: Provably secure double-block-length hash functions in a black-box model. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 330–342. Springer, Heidelberg (2005)
17. Hirose, S.: Some plausible construction of double-block-length hash functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)

18. Knudsen, L.R., Massey, J.L., Preneel, B.: Attacks on fast double block length hash functions. Journal of Cryptology 11(1), 59–72 (1998)
19. Knudsen, L.R., Mendel, F., Rechberger, C., Thomsen, S.S.: Cryptanalysis of MDC-2. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 106–120. Springer, Heidelberg (2009)
20. Lai, X., Massey, J.L.: Hash function based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
21. Lee, J., Kwon, D.: The security of Abreast-DM in the ideal cipher model. IACR ePrint Archive 2009/225 (2009)
22. Lee, J., Steinberger, J.: Multi-property-preserving domain extension using polynomial-based modes of operation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 573–596. Springer, Heidelberg (2010)
23. Lee, J., Stam, M., Steinberger, J.: The collision security of Tandem-DM in the ideal cipher model. IACR ePrint Archive 2010/409 (2010)
24. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
25. Lucks, S.: A collision-resistant rate-1 double-block-length hash function. Symmetric Cryptography, Dagstuhl Seminar Proceedings 07021 (2007)
26. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of step-reduced SHA-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 126–143. Springer, Heidelberg (2006)
27. Merkle, R.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
28. Meyer, C., Schilling, M.: Chargement securise d'un programma avec code de detection de manipulation (1987)
29. NIST: Cryptographic Hash Competition, `http://www.nist.gov/hash-competition`
30. Ozen, O., Stam, M.: Another glance at double-length hashing. In: Parker, M.G. (ed.) IMACC 2009. LNCS, vol. 5921, pp. 176–201. Springer, Heidelberg (2009)
31. 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)
32. Ristenpart, T., Shrimpton, T.: How to build a hash function from any collision-resistant function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 147–163. Springer, Heidelberg (2007)
33. 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)
34. Rogaway, P., Steinberger, J.: Security/efficiency tradeoffs for permuation-based hashing. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008)
35. 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)
36. Shrimpton, T., Stam, M.: Building a collision-resistant function from non-compressing primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008)
37. Stam, M.: Beyond uniformity: Security/efficiency tradeoffs for compression functions. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 397–412. Springer, Heidelberg (2008)

38. Stam, M.: Blockcipher based hashing revisited. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 67–83. Springer, Heidelberg (2009)
39. Steinberger, J.: The collision intractability of MDC-2 in the ideal-cipher model. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 34–51. Springer, Heidelberg (2007)
40. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
41. Wang, X., Lai, X., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
42. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
43. Wu, H.: The Hash Function JH. Submission to NIST (2008), http://icsd.i2r.a-star.edu.sg/staff/hongjun/jh/jh.pdf

A Proof of Lemma 2

For each $z \in \{0,1\}^n$, let

$$\mathcal{R}_1(z) = \left\{ w \in \{0,1\}^{2n} : \exists\ IV \xrightarrow{i} w \text{ such that } w_R = z \right\},$$
$$\mathcal{R}_2(z) = \left\{ w \in \{0,1\}^{2n} : \exists\ u \xrightarrow{i} v \xrightarrow{i} w \text{ such that } w_R = z \right\},$$
$$\mathcal{R}_3(z) = \left\{ w \in \{0,1\}^{2n} : \exists\ u \xrightarrow{j} v \xrightarrow{i} w \text{ such that } i > j \text{ and } w_R = z \right\},$$
$$\mathcal{R}_4(z) = \left\{ w \in \{0,1\}^{2n} : \exists\ u \xrightarrow{i} v \xrightarrow{j} w \text{ such that } i > j \text{ and } w_R = z \right\}.$$

Then we have

$$\mathcal{R}^q(z) = \mathcal{R}_1(z) \cup \mathcal{R}_2(z) \cup \mathcal{R}_3(z) \cup \mathcal{R}_4(z) \cup \{IV\}.$$

Define events

$$\mathsf{Mult}_\alpha(2m) \Leftrightarrow \max_{z\in\{0,1\}^n} |\mathcal{R}_\alpha(z)| > 2m,$$

for $\alpha = 1,\ldots,4$. Since $M = 8m+1$, we have

$$\mathbf{Pr}\left[\mathsf{Mult}(M)\right] \le \sum_{i=1}^{4} \mathbf{Pr}\left[\mathsf{Mult}_i(2m)\right]. \tag{5}$$

Event $\mathsf{Mult}_1(2m)$*:* If there exists a path $IV \xrightarrow{i} w$ such that $w_R = z$ for a fixed $z \in \{0,1\}^n$, then it should hold that $IV_L + x_L^i + y_R^i = z$ for some

$$(x_L^i, y_R^i) \in \left\{ (X^i, \theta(X_\sigma^i + Y_\sigma^i) + X^i), (X_\sigma^i, \theta(X^i + Y^i) + X_\sigma^i) \right\}.$$

Some rewriting then shows that $X_\sigma^i + Y_\sigma^i = \theta^{-1}(IV_L + z)$ or $X^i + Y^i = \theta^{-1}(IV_L + z)$ needs to happen (where we used that $\theta \neq 0$). Suppose that the (X^i, K^i, Y^i) triple arose from a forward query. Since the $i-1$ query-response cycles can have resulted in at most $2(i-1)$ queries to the blockcipher with key K^i, the current response Y^i is uniform over a set of size at least $N - 2(i-1) \geq N'$ and as a consequence the probability of $X^i + Y^i$ hitting a predetermined value is at most $1/N'$. Almost exactly the same argument holds when the triple arose from an inverse query or the conjugate query.

The number of indices i satisfying this equation is greater than m with probability at most $\binom{q}{m}(2/N')^m$. Since each index i determines at most two values of w, we have

$$\mathbf{Pr}\left[\mathsf{Mult}_1(2m)\right] \leq N\binom{q}{m}\left(\frac{2}{N'}\right)^m. \tag{6}$$

Event $\mathsf{Mult}_2(2m)$*:* Fix $z \in \{0,1\}^n$. If there exists a path $u \xrightarrow{i} v \xrightarrow{i} w$ such that $w_R = z$ for some u, v and w, then it should hold that $y_L^{i'} + x_L^{i''} + y_R^{i''} = z$, where

$$y_L^{i'} \in \left\{X^i + Y^i, X_\sigma^i + Y_\sigma^i\right\},$$

and

$$(x_L^{i''}, y_R^{i''}) \in \left\{(X^i, \theta(X_\sigma^i + Y_\sigma^i) + X^i), (X_\sigma^i, \theta(X^i + Y^i) + X_\sigma^i)\right\}.$$

The number of indices i satisfying this equation is greater than m with probability at most $\binom{q}{m}(4/N')^m$. Here we additionally use that $\theta \neq 1$, for instance the case $y_L^{i'} = X^i + Y^i$ and $(x_L^{i''}, y_R^{i''}) = (X_\sigma^i, \theta(X^i + Y^i) + X_\sigma^i)$ would require $(\theta + 1)(X^i + Y^i) = z$ (which we cannot successfully bound for $z = 0$ if $\theta = 1$). Since each index i determines at most two values of w, we have

$$\mathbf{Pr}\left[\mathsf{Mult}_2(2m)\right] \leq N\binom{q}{m}\left(\frac{4}{N'}\right)^m. \tag{7}$$

Event $\mathsf{Mult}_3(2m)$*:* Fix $z \in \{0,1\}^n$. For each $i \in [1,q]$, the probability that there exists $j < i$ such that $u \xrightarrow{j} v \xrightarrow{i} w$ and $w_R = z$ is not greater than $4q/N'$. Similar to the above analysis, we have

$$\mathbf{Pr}\left[\mathsf{Mult}_3(2m)\right] \leq N\binom{q}{m}\left(\frac{4q}{N'}\right)^m. \tag{8}$$

Event $\mathsf{Mult}_4(2m)$*:* This analysis is rather complicated since for each i, there might be a multiple number of indices $j < i$ that determine distinct nodes w such that $w_R = z$. So we would like to limit this number. Define event

$$\mathsf{Ex}(m_2) \Leftrightarrow \max_{z' \in \{0,1\}^n} \left|\left\{j \in [1,q] : x_L^j + y_R^j = z'\right\}\right| > m_2,$$

where

$$(x_L^j, y_R^j) \in \left\{(X^j, \theta(X_\sigma^j + Y_\sigma^j) + X^j), (X_\sigma^j, \theta(X^j + Y^j) + X_\sigma^j)\right\}.$$

Then it is easy to prove

$$\mathbf{Pr}\left[\mathsf{Ex}(m_2)\right] \le N\binom{q}{m_2}\left(\frac{2}{N'}\right)^{m_2}. \tag{9}$$

Consider the event $\mathsf{Mult}_4(2m) \cap \overline{\mathsf{Ex}(m_2)}$. For each $y_L^i \in \left\{X^i+Y^i, X_\sigma^i+Y_\sigma^i\right\}$, there would be at most m_2 indices $j < i$ that determine distinct nodes w such that $w_R = y_L^i + x_L^j + y_R^j = z$. It means that there should exist at least m_1 indices i such that there exists $j < i$ satisfying $w_R = y_L^i + x_L^j + y_R^j = z$. Since this event occurs with probability at most $\binom{q}{m_1}(4q/N')^{m_1}$, we have

$$\mathbf{Pr}\left[\mathsf{Mult}_4(2m) \cap \overline{\mathsf{Ex}(m_2)}\right] \le N\binom{q}{m_1}\left(\frac{4q}{N'}\right)^{m_1}. \tag{10}$$

From inequalities (5), (6), (7), (8), (9) and (10), we have

$$\mathbf{Pr}\left[\mathsf{Mult}(M)\right] \le 2N\left(\frac{12q}{mN'}\right)^{m} + N\left(\frac{12q^2}{mN'}\right)^{m} + N\left(\frac{6q}{m_2N'}\right)^{m_2} + N\left(\frac{12q^2}{m_1N'}\right)^{m_1},$$

where we use inequality

$$\binom{q}{\alpha} \le \left(\frac{3q}{\alpha}\right)^{\alpha},$$

for any positive integer α such that $\alpha \le q$.

B Upper Bounding $\mathbf{Pr}\left[\mathsf{D}_{\boldsymbol{\alpha}}\right]$

Throughout the following analysis, we will fix $i \in [1, q]$ and compute the probability that the i-th query-response cycle Δ^i completes each configuration.

Event D_1*:* Since $v_L \in \left\{X^i+Y^i, X_\sigma^i+Y_\sigma^i\right\}$ and $u \ne w$, we have $X^i+Y^i = X_\sigma^i+Y_\sigma^i$. Since this equality holds with probability at most $1/N'$, we have

$$\mathbf{Pr}\left[\mathsf{D}_1\right] \le \frac{q}{N'}.$$

Event D_2*:* Fix $j (< i)$. Then we have

$$(v_L, u_L + v_R) \in \left\{(Z^i, \theta Z_\sigma^i), (Z_\sigma^i, \theta Z^i)\right\} \cap \left\{(Z^j, \theta Z_\sigma^j), (Z_\sigma^j, \theta Z^j)\right\},$$

where $Z^i = X^i + Y^i$, $Z_\sigma^i = X_\sigma^i + Y_\sigma^i$, $Z^j = X^j + Y^j$ and $Z_\sigma^j = X_\sigma^j + Y_\sigma^j$. (We will use these notations in the following events as well.) By checking out all the possible 2 sets of equations (up to equivalence), we see the probability of the above inclusion is at most $2/(N')^2$. Therefore, we have

$$\mathbf{Pr}\left[\mathsf{D}_2\right] \le \frac{2q^2}{(N')^2}.$$

Event D_3*:* Since $v_L = Z^i = Z^i_\sigma$ and this equation holds with probability at most $1/N'$, we have

$$\mathbf{Pr}\left[\mathsf{D}_3\right] \le \frac{q}{N'}.$$

From now on, we will also fix $u \in \mathcal{R}^{i-1}(K^i)$. Note that $\left|\mathcal{R}^{i-1}(K^i)\right| \le M$ without the occurrence of $\mathsf{Mult}(M)$.

Event D_4*:* By Lemma 1, we have $u_L \in \left\{K^i + \theta Z^i, K^i + \theta Z^i_\sigma\right\}$. Since this inclusion holds with probability at most $2/N'$, we have

$$\mathbf{Pr}\left[\mathsf{D}_4\right] \le \frac{2qM}{N'}.$$

Event D_5*:* Fix $v \in \mathcal{S}_1^{i-1}$, where $\left|\mathcal{S}_1^{i-1}\right| \le 2qM + 1$ by Lemma 4. Since the probability that the i-th query-response cycle determines $u \xrightarrow{i} v$ is at most $2/(N')^2$, we have

$$\mathbf{Pr}\left[\mathsf{D}_5\right] \le \frac{2qM(2qM+1)}{(N')^2}.$$

Event D_6*:* First, consider the case where the two i-labelled edges have the same sign. In this case, fix a pair of indices (j, j') such that there exists a configuration $v \xleftarrow{j} w \xleftarrow{j'} w'$ for some v and w' with $v_L = w'_L$. The number of such pairs is at most M without the occurrence of $\mathsf{E}_1(M)$. Since (j, j') determines 4 nodes v, and the probability of $u \xrightarrow{i} v$ (for a given u) is at most $2/(N')^2$, the overall probability of D_6 with the i-labelled edges of the same sign is at most $8qM^2/(N')^2$.

Next, consider the case where the two i-labelled edges have opposite signs. In this case, fix a pair $(j, j') \in [1, q]^2$. Then for each pair, four possible pairs (v_L, w'_L) are determined. For each of these, the probability that either $(X+Y, X_\sigma+Y_\sigma) = (v_L, w'_L)$ or $(X+Y, X_\sigma+Y_\sigma) = (w'_L, v_L)$ is at most $2/(N')^2$. Therefore, the probability of D_6 with the i-labelled edges of opposite signs is at most $8q^3/(N')^2$. To summarize, we have

$$\mathbf{Pr}\left[\mathsf{D}_6\right] \le \frac{8q(q^2+M^2)}{(N')^2}.$$

Event D_7*:* Fix $j(< i)$. Then we have

$$\begin{aligned} v \in &\left\{(Z^i, u_L + \theta Z^i_\sigma), (Z^i_\sigma, u_L + \theta Z^i)\right\} \\ &\cap \left\{(Z^j, Z^i + \theta Z^j_\sigma), (Z^j_\sigma, Z^i + \theta Z^j), (Z^j, Z^i_\sigma + \theta Z^j_\sigma), (Z^j_\sigma, Z^i_\sigma + \theta Z^j)\right\}. \end{aligned}$$

By checking out all the possible 8 sets of equations, we see the probability of the above inclusion is at most $8/(N')^2$. Therefore, we have

$$\mathbf{Pr}\left[\mathsf{D}_7\right] \le \frac{8q^2M}{(N')^2}.$$

Event D_8*:* Fix $v \in \mathcal{S}_2^{i-1}$, where $\left|\mathcal{S}_2^{i-1}\right| \leq 8qM + 2q + 4M$ by Lemma 4. Since the probability that the i-th query-response cycle determines $u \xrightarrow{i} v$ is at most $2/(N')^2$, we have

$$\mathbf{Pr}\left[\mathsf{D}_8\right] \leq \frac{2qM(8qM + 2q + 4M)}{(N')^2}.$$

Event D_9*:* First, consider the case where the two i-labelled edges have the same sign. In this case, fix a pair of colliding indices (j, j') such that there exists a configuration $v \xrightarrow{j} w \xleftarrow{j'} v'$ for some v and v' with $v_L = v'_L$. The number of such pairs is at most m_2 without the occurrence of $\mathsf{E}_3(m_2)$. Since the probability that $v_R = K^j \in \left\{u_L + \theta Z^i, u_L + \theta Z^i_\sigma\right\}$ is at most $2/N'$, the probability of D_9 with the i-labelled edges of the same sign is at most $2qMm_2/N'$.

Next, consider the case where the two i-labelled edges have opposite signs. In this case, fix a pair of colliding indices (j, j'). The number of such pairs is at most M without the occurrence of $\mathsf{E}_2(M)$. Then for each $w_R \in \{0,1\}^n$, (v_L, v'_L) is determined with four possibilities. For each possibility, the probability that either $(X + Y, X_\sigma + Y_\sigma) = (v_L, v'_L)$ or $(X + Y, X_\sigma + Y_\sigma) = (v'_L, v_L)$ is at most $2/(N')^2$. Therefore, the probability of D_9 with the i-labelled edges of opposite signs is at most $8qMN/(N')^2$. To summarize, we have

$$\mathbf{Pr}\left[\mathsf{D}_9\right] \leq \frac{2qMm_2}{N'} + \frac{8qMN}{(N')^2} \leq \frac{2qM(m_2 + 8)}{N'}.$$

Event D_{10}*:* Fix $v \in \mathcal{S}_3^{i-1}$, where $\left|\mathcal{S}_3^{i-1}\right| \leq 4qM + 4m_2^2$ by Lemma 4. Since the probability that the i-th query-response cycle determines $u \xrightarrow{i} v$ is at most $2/(N')^2$, we have

$$\mathbf{Pr}\left[\mathsf{D}_{10}\right] \leq \frac{2qM(4qM + 4m_2^2)}{(N')^2}.$$

Event D_{11}*:* Fix $j (< i)$. Then we have

$$\begin{aligned}(v_R, w_R) = (K^j, K^i) \in \{&(u_L + \theta Z^i, Z^i_\sigma + Z^j), (u_L + \theta Z^i, Z^i_\sigma + Z^j_\sigma),\\ &(u_L + \theta Z^i_\sigma, Z^i + Z^j), (u_L + \theta Z^i_\sigma, Z^i + Z^j_\sigma)\}.\end{aligned}$$

By checking out all the possible 4 sets of equations, we see the probability of the above inclusion is at most $4/(N')^2$. Therefore, we have

$$\mathbf{Pr}\left[\mathsf{D}_{11}\right] \leq \frac{4q^2M}{(N')^2}.$$

Event D_{12}*:* Fix $v \in \mathcal{S}_4^{i-1}$, where $\left|\mathcal{S}_4^{i-1}\right| \leq 4qMm_2^2$ by Lemma 4. Since the probability that the i-th query-response cycle determines $u \xrightarrow{i} v$ is at most $2/(N')^2$, we have

$$\mathbf{Pr}\left[\mathsf{D}_{12}\right] \leq \frac{8q^2M^2m_2^2}{(N')^2}.$$

C Towards Better Bounds

While MJH's provable collision resistance of up to $2^{\frac{2n}{3}-\log n}$ is an unmatched achievement for hash functions of its kind, we do not believe the result to be tight and more detailed case analysis might lead to even better bounds. *Grosso modo* the 12 cases of Lemma 5 can be divided into three categories: local aberrations $(\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3)$, connecting a size-2 component with a size-3 component $(\mathsf{C}_5, \mathsf{C}_8, \mathsf{C}_{10}, \mathsf{C}_{12})$, and connecting from a size-2 component to another component twice $(\mathsf{C}_4, \mathsf{C}_6, \mathsf{C}_7, \mathsf{C}_9, \mathsf{C}_{11})$. From the proof, it follows that the aberrations are not expected to occur until very close to 2^n queries have been made. For the remaining cases, the "bad" probabilities are all of the type qM/N' or $(qM/N')^2$, where M is related to the maximum occurrence of w_R over all 2-chains $u \xrightarrow{i} v \xrightarrow{j} w$. This implies $M > q^2/N$ resulting in an upper bound (on the advantage) polynomial in q^3/N^2, loosing meaning when $q > N^{2/3}$ (and it actually happens slightly sooner).

If we were to consider larger components instead, there is hope that we can reduce M and the bound accordingly. For instance, the maximum occurrence of w'_R over all 3-chains $u \xrightarrow{i} v \xrightarrow{j} w \xrightarrow{j'} w'$ is only expected to be q^3/N^2. If we, optimistically, plug in this value for M in qM/N, we arrive at q^4/N^3 indicating security might be provable up to $N^{3/4}$ queries using this approach. The problem is that to formalize this all and make it rigorous is a serious undertaking: without care the equivalent of Lemma 2 could take on 7 extra cases, all the sets in Lemma 4 grow and an extra set is likely to be needed, extra cases will need to be added to Lemma 5, and all added cases tend to be more complicated than the current ones.

Online Ciphers from Tweakable Blockciphers

Phillip Rogaway and Haibin Zhang

Dept. of Computer Science, University of California, Davis, California 95616, USA
{rogaway,hbzhang}@cs.ucdavis.edu

Abstract. Online ciphers are deterministic length-preserving permutations $\mathcal{E}_K$: $(\{0,1\}^n)^+ \rightarrow (\{0,1\}^n)^+$ where the i-th block of ciphertext depends only on the first i blocks of plaintext. Definitions, constructions, and applications for these objects were first given by Bellare, Boldyreva, Knudsen, and Namprempre. We simplify and generalize their work, showing that online ciphers are rather trivially constructed from tweakable blockciphers, a notion of Liskov, Rivest, and Wagner. We go on to show how to define and achieve online ciphers for settings in which messages need *not* be a multiple of n bits.

Keywords: Online ciphers, modes of operation, provable security, symmetric encryption, tweakable blockciphers.

1 Introduction

Background. Informally, a cryptographic transform is said to be *online* if it can be computed by an algorithm that reads in the (unknown number of) input bits—in order, one at a time—as it writes out the corresponding output bits—again in order, one at a time—never using more than a constant amount of memory or incurring more than a constant amount of latency.[1] Most blockcipher modes of operation *are* online—for example, modes like CBC, HMAC, and GCM certainly are. But one kind of transformation is not online, and can never be online: a *general cipher* [20], one secure in the customary sense of a PRP (a pseudorandom permutation). Such objects take a key K and a plaintext M of unbounded length and produce a ciphertext C of length $|M|$, doing so in such a way that the mapping resembles a random permutation. EME2 is a soon-to-be-standardized example [14]. The reason an online cipher can't be PRP-secure is simple: the first bit of output, for example, has got to depend on every bit of input, else it is trivial to distinguish the cipher from a random permutation. This requirement makes bounded memory, or latency, an impossibility.

One can weaken PRP security to capture what *is* possible in the online setting. Bellare, Boldyreva, Knudsen, and Namprempre (BBKN) were the first to do so, defining *online ciphers* [3]. The authors fix a parameter n (likely the blocksize of some underlying blockcipher). They then demand that the i-th (n-bit) block of ciphertext depend only on the first i blocks of plaintext (and, of course,

[1] For online ciphers, two alternative formulations—one corresponding to the opening sentence of the abstract, plus one other—will subsequently be described.

the key).[2] BBKN did not explicitly demand that encryption and decryption be computable with constant memory and latency, but follow-on work by Boldyreva and Taesombut strengthened BBKN's definition in a way that ensures this is so [9].

Like other kinds of ciphers, online ciphers can be secure in either the CPA (chosen plaintext) or CCA (chosen ciphertext, or "strong") sense, depending on whether the adversary is given oracle access to the decryption (or "backwards") functionality as well as the encryption (or "forwards") functionality. For online ciphers it initially seemed as though CCA-security was harder to achieve than CPA-security [3], but, at present, the most efficient CCA-secure construction has essentially the same overhead as the most efficient CPA-secure one, needing just one extra xor per block [22].

Online ciphers are useful tools. For example, BBKN demonstrate a simple recipe for turning a CCA-secure online cipher into an authenticated-encryption scheme (one prepends and appends a random value $R \in \{0,1\}^n$) [4]; Boldyreva and Taesombut (following, Fouque Joux, Martinet, and Valette [10]) show how to turn a CCA-secure online cipher into either an online encryption scheme secure against "blockwise-adaptive" CCA attacks, or else an online authenticated-encryption (AE) scheme likewise secure against BA-CCA attacks [9]; and Amanatidis, Boldyreva, and O'Neill describe the use of online ciphers to solve a database-security problem [1].

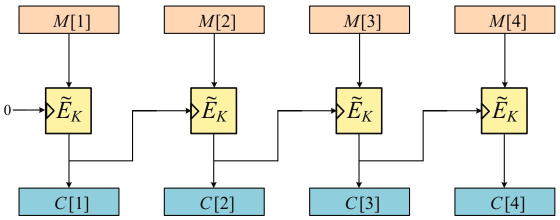
Our contribution. In this paper we make two contributions. First, we recast the constructions of BBKN [3,4], plus a subsequent construction by Nandi [21, 22], into the language of *tweakable blockciphers*, a notion of Liskov, Rivest, and Wagner [17]. The new starting point yields constructions more general and transparent than those before. See Fig. 1. Second, we show how to relax the notion of an online cipher to deal with messages that are *not* a multiple of n bits. Besides definitions, we provide a simple and efficient construction to handle this setting. Dealing with arbitrary-length inputs is a necessary precursor to practical schemes, which we also describe.

Discussion. The original BBKN paper had fairly complex schemes and proofs [3]. Nandi found some bugs in these proofs and offered up his own [21, 22]. BBKN corrected the issues in their proofs, which they regarded as minor, but the proofs remain complex [4]. BBKN's modes relied on xor-universal hash functions, and subsequent work did too, or else doubled the number of blockcipher calls [3,4,21, 22]. Our own constructions are simple, and they are natural generalizations of the existing schemes. The proofs are simple too. We do not regard this simplicity as a defect. Without the tweakable-blockcipher abstraction, constructions and proofs in this domain are *not* simple, as the above history suggests.

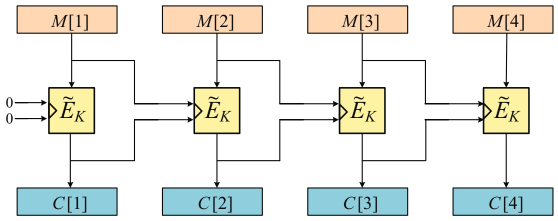
The question of fractional final blocks was earlier asked by Nandi [22, p. 361]. Note that one cannot just say to pad to the next multiple of n (as suggested, for example, by Bard [2, p. 134]). This doesn't make sense definitionally, because

[2] It follows that the i-th block of plaintext will likewise depend only on the key and the first i blocks of ciphertext.

```
TC1
algorithm $\mathcal{E}_K(M)$
if $M \notin (\{0,1\}^n)^+$ return $\perp$
$m \leftarrow |M|/n$;  $T \leftarrow 0^n$
for $j \leftarrow 1$ to $m$ do
    $C[j] \leftarrow \widetilde{E}_K^T(M[j])$
    $T \leftarrow C[j]$
return $C$
```



```
TC2
algorithm $\mathcal{E}_K(M)$
if $M \notin (\{0,1\}^n)^+$ return $\perp$
$m \leftarrow |M|/n$;  $T \leftarrow 0^{2n}$
for $j \leftarrow 1$ to $m$ do
    $C[j] \leftarrow \widetilde{E}_K^T(M[j])$
    $T \leftarrow M[j] \,\|\, C[j]$
return $C$
```



```
TC3
algorithm $\mathcal{E}_K(M)$
if $M \notin (\{0,1\}^n)^+$ return $\perp$
$m \leftarrow |M|/n$;  $T \leftarrow 0^n$
for $j \leftarrow 1$ to $m$ do
    $C[j] \leftarrow \widetilde{E}_K^T(M[j])$
    $T \leftarrow M[j] \oplus C[j]$
return $C$
```

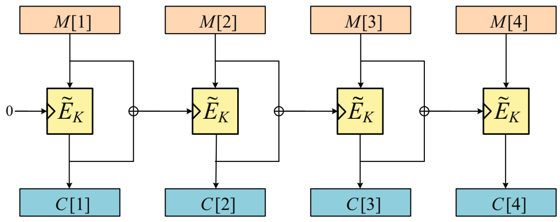


Fig. 1. Modes TC1, TC2, and TC3. The first is a CPA-secure online cipher; the next two are CCA-secure. Plaintexts must have a length divisible by n. In the diagrams, the object $\widetilde{E}_K$ is a keyed tweakable blockcipher; the tweak comes in at left, the n-bit input comes in at the top, and the n-bit output emerges from the bottom. TC1, TC2, and TC3 simplify and generalize HCBC1 [4, 3], HCBC2 [4], and MHCBC [21, 22], respectively.

it leaves one without any notion for what it *means* to have an online-encipher outside of $(\{0,1\}^n)^+$, and it doesn't make sense procedurally, because, if you *did* pad and then encipher, you would no longer have a cipher at all (ciphers preserve length).

Dealing with arbitrary-length inputs is important for efficiency: if we are going to turn an online cipher into an authenticated-encryption scheme or a blockwise-adaptive (BA) online encryption scheme, there will be ciphertext expansion if is forced to pad.

Correctly constructing arbitrary-input-length online ciphers would be difficult without the tweakable blockcipher abstraction. Liskov, Rivest, and Wagner had argued that tweakable blockciphers would be useful tools for designing symmetric protocols [17]. Our results bolster this point of view.

Joux, Martinet, and Valette [15] implicitly argue that our notion of security for online ciphers is not strong enough, since it treats the encryption operation as "atomic," not attending to attacks that, for example, select the second block of a plaintext being encrypted based on the encryption of the first. We do not dispute this insight, but nonetheless prefer not to deal with these blockwise-adaptive adversarial attacks. First, enriching the security notion to allow for them hardly changes things in the CPA-setting [11, Theorem 8]. Second, "atomic" online ciphers are already useful for higher-level applications, as described above. Finally, "true" CCA security becomes impossible in the BA setting, leading to more subtle definitions for what actually is achievable [9].

2 Preliminaries

Notation. A *string* is a member of $\{0,1\}^*$. The notation $A \parallel B$, or just $A\,B$, denotes the concatenation of strings A and B. If X is a string then $|X|$ denotes its length. The empty string is denoted ε. Throughout this paper we fix a positive number n called the *blocksize.* The set $(\{0,1\}^n)^+$ is the set of all strings having length jn for some $j \geq 1$. If $X \in (\{0,1\}^n)^+$ we let $X[i]$ denotes its ith n-bit block, so $X = X[1] \cdots X[m]$ where $m = |X|/n$. We will later extend this notation to the case when X is not a multiple of n bits. We write $X[i..j]$ for $X[i] \cdots X[j]$.

Ciphers. A map $f\colon \mathcal{X} \to \mathcal{X}$ for $\mathcal{X} \subseteq \{0,1\}^*$ is a *length-preserving function* if $|f(x)| = |x|$ for all $x \in \{0,1\}^*$. It is a length-preserving *permutation* if it is also a permutation. A *cipher* is a map $\widetilde{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}$ is a nonempty set (finite or otherwise endowed with some distribution), $\mathcal{M} \subseteq \{0,1\}^*$ is a nonempty set, and $\mathcal{E}_K = \mathcal{E}(K,\cdot)$ is a length-preserving permutation for all $K \in \mathcal{K}$. The set $\mathcal{K}$ is called the *key space* and $\mathcal{M}$ is called the *message space.* If $\mathcal{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ is a cipher then its *inverse* is the cipher $\mathcal{E}^{-1}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ defined by $\mathcal{E}^{-1}(K,Y) = \mathcal{E}_K^{-1}(Y)$ being the unique point X such that $\mathcal{E}_K(X) = Y$.

Blockciphers and tweakable blockciphers. A *blockcipher* is a function $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite nonempty set and $E_K(\cdot) = E(K,\cdot)$ is a permutation on $\{0,1\}^n$ for every $K \in \mathcal{K}$. Equivalently, a blockcipher is a cipher with message space $\mathcal{M} = \{0,1\}^n$. A *tweakable* blockcipher is a function $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite nonempty set and $\mathcal{T}$ is a nonempty set (the *tweak space*) and $\widetilde{E}_K^T(\cdot) = \widetilde{E}(K,T,\cdot)$ is a permutation on $\{0,1\}^n$ for every $K \in \mathcal{K}$, $T \in \mathcal{T}$.

Let $\mathrm{Perm}(n)$ be the set of all permutations on n bits, $\mathrm{Perm}(\mathcal{M})$ be the set of all length-preserving permutations on the finite set $\mathcal{M} \subseteq \{0,1\}^*$, and $\mathrm{Perm}(\mathcal{T},n)$ the set of all functions $\pi\colon \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\pi_T(\cdot) = \pi(T,\cdot)$ is a permutation for each $T \in \mathcal{T}$. We may regard $\mathrm{Perm}(n)$, $\mathrm{Perm}(\mathcal{M})$, and $\mathrm{Perm}(\mathcal{T},n)$ as blockciphers, ciphers, and tweakable blockciphers, respectively; they are the ideal blockcipher on n bits, the ideal cipher on $\mathcal{M}$, and the ideal tweakable blockcipher on n bits and tweak space $\mathcal{T}$. When an adversary $\mathcal{A}$ is run with an oracle $\mathcal{O}$ we let $\mathcal{A}^{\mathcal{O}} \Rightarrow 1$ denote the event that $\mathcal{A}$ outputs 1. Define the prp, $\pm$prp, $\widetilde{\mathrm{prp}}$, and $\pm\widetilde{\mathrm{prp}}$ *advantage* of $\mathcal{A}$ against E or $\widetilde{E}$ by:

$$\begin{aligned}
\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{A}) &= \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{E_K} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \mathrm{Perm}(n) : \mathcal{A}^{\pi} \Rightarrow 1] \\
\mathbf{Adv}_E^{\pm\mathrm{prp}}(\mathcal{A}) &= \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{E_K, E_K^{-1}} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \mathrm{Perm}(n) : \mathcal{A}^{\pi,\pi^{-1}} \Rightarrow 1] \\
\mathbf{Adv}_{\widetilde{E}}^{\widetilde{\mathrm{prp}}}(\mathcal{A}) &= \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\widetilde{E}_K} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n) : \mathcal{A}^{\pi} \Rightarrow 1] \\
\mathbf{Adv}_{\widetilde{E}}^{\pm\widetilde{\mathrm{prp}}}(\mathcal{A}) &= \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\widetilde{E}_K,\, \widetilde{E}_K^{-1}} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n) : \mathcal{A}^{\pi,\, \pi^{-1}} \Rightarrow 1]
\end{aligned}$$

Online ciphers. A length-preserving function $f\colon (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$ is *online* if, for all i, $f(X)[1..i]$ depends only on $X[1..i]$. Here we say that $f(X)[1..i]$ depends only on $X[1..i]$ if $f(XY)[1..i] = f(XY')[1..i]$ for all $X \in \{0,1\}^{in}$ and $Y, Y' \in \{0,1\}^*$ where $f(XY)$ and $f(XY')$ are defined. A cipher $\mathcal{E}\colon \mathcal{K} \times (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$ is online if each $\mathcal{E}_K$ is. Let Online(n): $\mathcal{K} \times (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$ be the *ideal* online cipher on n bits: each key names one of the possible online ciphers, the set being given the uniform distribution in the natural way. If $\mathcal{E}\colon \mathcal{K} \times (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$ is an online cipher and $\mathcal{A}$ is an adversary we define:

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{E}}^{\mathrm{oprp}}(\mathcal{A}) &= \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \mathrm{Online}(n) : \mathcal{A}^{\pi} \Rightarrow 1] \\
\mathbf{Adv}_{\mathcal{E}}^{\pm\mathrm{oprp}}(\mathcal{A}) &= \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K,\, \mathcal{E}_K^{-1}} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \mathrm{Online}(n) : \mathcal{A}^{\pi,\, \pi^{-1}} \Rightarrow 1]
\end{aligned}$$

We comment that the definitions allow variable-input-length (VIL) attacks: the adversary may ask queries of varying lengths. On the other hand, definitions only countenance ciphers on $\mathcal{M} = (\{0,1\}^n)^+$, and it is not obvious what to do beyond this domain, Online(n) being quite specific to it.

Discussion. The notion for an online cipher just given, taken from BBKN [3], can be criticized for not prohibiting, for example, that that computation of $C[m]$ requires one to retain all of $M[1] \cdots M[m]$. A stronger notion appears in Boldyreva and Taesombut [9], the definition asserting that $C[i]$ may only depend on $M[i]$, $M[i-1]$, $C[i-1]$, and the underlying key. We believe that this requirement does not make for a desirable security definition: the cipher in which each $C[i]$ is a random permutation of $M[i]$, tweaked by $M[i-1] \,\|\, C[i-1]$, ought not to be regarded ideal, since one can easily do better and still, intuitively, be "online." Still, our *constructions* enjoy the BT-style locality property, ensuring that they can be implemented with constant latency and memory.

An alternative notion for an online cipher would capture the intuition from the opening paragraph of this paper, saying that a cipher $\mathcal{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ is Online[m] if it can be implemented by an algorithm that is fed in bits one at a time, and that retains just m bits of state. This would natively handle ciphers on arbitrary bit strings. We leave it as an open question to explore these ideas.

3 Online Ciphers Achieving CPA-Security

Let $\widetilde{E}\colon \mathcal{K} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. From this primitive we define a cipher $\mathcal{E}$ =TC1[$\widetilde{E}$] with key space $\mathcal{K}$ and message space $\mathcal{M} = (\{0,1\}^n)^+$. See Fig. 1. The construction is online CPA-secure, as formalized below.

Theorem 1 (TC1 is oprp-secure). *Let* $\widetilde{\pi} = \text{Perm}(\{0,1\}^n, n)$*. If* $\mathcal{A}$ *asks queries having at most* σ *blocks then* $\mathbf{Adv}^{\text{oprp}}_{\text{TC1}[\widetilde{\pi}]}(\mathcal{A}) \le 1.5\,\sigma^2/2^n$*.* ∎

We omit the proof because we will in a moment be proving, by analogous but slightly more involved means, what is essentially a stronger result: online CCA-security for the equally efficient cipher TC3. The complexity-theoretic analog for the theorem, using a "real" tweakable PRP $\widetilde{E}$ instead of the ideal tweakable PRP $\widetilde{\pi}$, follows by standard techniques. One would need $\widetilde{E}$ to be secure in the prp-sense. We omit the theorem statement, showing later how it would look for scheme TC3.

Mechanism TC1 is a generalization of BBKN's mode of operation HCBC1 [4] (formerly named HCBC [3]); the latter can be realized as a special case of TC1 by selecting the tweakable blockcipher $\widetilde{E}$: $(\mathcal{K}_1 \times \mathcal{K}_2) \times \{0,1\}^n \to \{0,1\}^n$ to be $\widetilde{E}^T_{K1\ K2}(X) = E_{K1}(M \oplus H_{K2}(T))$ where H: $\mathcal{K}_2 \times \{0,1\}^n \to \{0,1\}^n$ is an almost-xor universal hash function and E is a blockcipher.[3] This is in fact the "standard" construction of a tweakable blockcipher from an ordinary one [17]. Of course one can instantiate the tweakable blockcipher $\widetilde{E}$ from an ordinary blockcipher E in a variety of other ways as well. We comment that TC1 can also be regarded as Liskov, Rivest, and Wagner's "tweak block chaining" mode [17, Section 4] but with a zero IV.

Note that TC1 is not a secure online cipher with respect to CCA attacks. A simple attack is as follows. The adversary makes a decryption query of $C \,\|\, C \,\|\, C$ for any $C \in \{0,1\}^n$. The oracle returns $M_1 \,\|\, M_2 \,\|\, M_3$ as the reply. If $M_2 = M_3$, return 1; otherwise, return 0. Under the TC1 construction, one will always have that $M_2 = M_3$, but with a random on-line cipher this will rarely be true.

4 Online Ciphers Achieving CCA-Security

Let $\widetilde{E}$: $\mathcal{K} \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. From this primitive we define a cipher $\mathcal{E}$ =TC2[$\widetilde{E}$] with key space $\mathcal{K}$ and message space $\mathcal{M} = (\{0,1\}^n)^+$. Again see Fig. 1. The construction is CCA-secure, as formalized below.

Theorem 2 (TC2 is ±oprp-secure). *Let* $\widetilde{\pi} = \text{Perm}(\{0,1\}^{2n}, n)$*. If* $\mathcal{A}$ *asks queries having at most* σ *blocks then* $\mathbf{Adv}^{\pm\text{oprp}}_{\text{TC2}[\widetilde{\pi}]}(\mathcal{A}) \le 1.5\,\sigma^2/2^n$*.* ∎

We again omit the proof, and the complexity-theoretic analog, which would this time need the $\pm\widetilde{\text{prp}}$ assumption, preferring, for concision, to do this just for TC2's more efficient cousin, TC3.

Mechanism TC2 is a generalization of BBKN's mode HCBC2 [4] (formerly named HPCBC), which can be regarded as TC2 with a tweakable blockcipher

[3] We recall the definition, due to Krawczyk [16], that H: $\mathcal{K}2 \times \mathcal{X} \to \{0,1\}^n$ is ϵ-almost XOR universal (ϵ-AXU) if for all distinct $X, X' \in \mathcal{X}$ and all $C \in \{0,1\}^n$ we have that $\Pr[H_K(X) \oplus H_K(X') = C] \le \epsilon$, the probability over $K \xleftarrow{\$} \mathcal{K}2$. Simple constructions achieve $\epsilon = 2^{-n}$, the minimum value possible.

$\widetilde{E}$: $(\mathcal{K}_1 \times \mathcal{K}_2) \times \{0,1\}^n \to \{0,1\}^n$ of $\widetilde{E}^T_{K1\ K2}(X) = E_{K1}(M \oplus H_{K2}(T)) \oplus H_{K2}(T)$ where H: $\mathcal{K}_2 \times \{0,1\}^{2n} \to \{0,1\}^n$ is an almost-xor universal hash function and E is a blockcipher. This is also the "standard" construction of a strong tweakable blockcipher from an ordinary one [17].

We are now ready to consider TC3. Let $\widetilde{E}$: $\mathcal{K} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. From this primitive define the online cipher $\mathcal{E}$ =TC3$[\widetilde{E}]$ with key space $\mathcal{K}$ and message space $\mathcal{M} = (\{0,1\}^n)^+$. Again see Fig. 1. The construction is CCA-secure, as formalized below.

Theorem 3 (TC3 is ±oprp-secure). *Let* $\widetilde{\pi} = \text{Perm}(\{0,1\}^n, n)$. *If* $\mathcal{A}$ *asks queries having at most* σ *blocks then* $\mathbf{Adv}^{\pm\text{oprp}}_{\text{TC3}[\widetilde{\pi}]}(\mathcal{A}) \leq 1.5\,\sigma^2/2^n$. ▮

The idea of the proof is to "give up"—regard the adversary as having won—if we ever generate a "new" tweak that collides with any prior one.

Proof. Without loss of generality we can assume that $\mathcal{A}$ is deterministic and makes queries totaling exactly σ blocks. We can further assume that it never repeats an encryption query, never repeats a decryption query, never asks a decryption query of a value that it earlier received from an encryption query, and never asks an encryption query of a value that it earlier received from a decryption query. For strings X, $X_1, \ldots, X_I \in (\{0,1\}^n)^*$, let find$(X;\ X_1, \ldots, X_I)$ be the unique pair of numbers $(\imath, \ell)$ for which X and $X_\imath$ share a common prefix $X[1..\ell] = X_\imath[1..\ell]$, no $X_\jmath$ ($\jmath \in [1..I]$) shares a *longer* common prefix with X ($X[1..\ell+1] = X_\jmath[1..\ell+1]$), and $\imath$ is the *smallest* index in $[1..I]$ for which the above is true. If $X = \varepsilon$ define find$(X;\ X_1, \ldots, X_I) = (0,0)$. By way of examples, if $a, b, c \in \{0,1\}^n$ are distinct blocks then find$(abca;\ abaa, abcb, abcc) = (2,3)$, find$(abca;\ a, abc, abcab) = (3,4)$, and find$(abca;\ bbab, cba, b) = (1,0)$.

We employ the code-based games [6] shown in Fig. 2. Booleans are silently initialized to false and integers to 0. The one variable that is a set, $\mathcal{T}$, is silently initialized to $\{0^n\}$. (This is done because $\mathcal{T}$ will be used to record the set of tweaks that have been utilized and, in effect, 0^n is a tweak that is always used—it is used in processing each query's first block.) Partial functions π_x (where $x \in \{0,1\}^*$) are, initially, everywhere *undefined*. As they grow we refer to their current domain and range by domain(π_x) and range(π_x). We write codomain(π_x) and corange(π_x) for the complements relative to $\{0,1\}^n$.

We begin Fig. 2 with game G_1, which precisely emulates the TC3 construction with the ideal tweakable blockcipher $\widetilde{E}$. We end with game G_6, which precisely emulates the ideal online cipher. Thus we have that $\mathbf{Adv}^{\text{oprp}}_{\widetilde{E}}(\mathcal{A}) = \Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_6^{\mathcal{A}} \Rightarrow 1]$. Games G_2, G_3, G_4, and G_5 are hybrid games in between these two extremes, and we bound the $\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_6^{\mathcal{A}} \Rightarrow 1]$ as $\sum_{1 \leq j \leq 5} \left(\Pr[G_j^{\mathcal{A}} \Rightarrow 1] - \Pr[G_{j+1}^{\mathcal{A}} \Rightarrow 1]\right)$.

Passing from games G_1 to G_2 is just the usual approach of lazy sampling [6]; the games G_1 and G_2 are adversarially indistinguishable. By the game-playing

```
100 procedure $E(M)$
101 $m \leftarrow |M|/n$;  $(\imath, \ell) \leftarrow$ find$(M;\ M_1, \ldots, M_i)$
102 $C[1..\ell] \leftarrow C_\imath[1..\ell]$
103 for $j \leftarrow \ell + 1$ to $m$ do
104     if $j = 1$ then $t \leftarrow 0^n$
105     else $t \leftarrow M[j-1] \oplus C[j-1]$;
106     $C[j] \leftarrow \pi_t(M[j])$
107 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$
108 return $C$

150 procedure $D(C)$
151 $m \leftarrow |C|/n$;  $(\imath, \ell) \leftarrow$ find$(C;\ C_1, \ldots, C_i)$
152 $M[1..\ell] \leftarrow M_\imath[1..\ell]$
153 for $j \leftarrow \ell + 1$ to $m$ do
154     if $j = 1$ then $t \leftarrow 0^n$
155     else $t \leftarrow M[j-1] \oplus C[j-1]$;
156     $M[j] \leftarrow \pi_t^{-1}(C[j])$
157 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$
158 return $M$                                        Game $G_1$
```

```
200 procedure $E(M)$
201 $m \leftarrow |M|/n$;  $(\imath, \ell) \leftarrow$ find$(M;\ M_1, \ldots, M_i)$
202 $C[1..\ell] \leftarrow C_\imath[1..\ell]$
203 for $j \leftarrow \ell + 1$ to $m$ do
204     if $j = 1$ then $t \leftarrow 0^n$
205     else $t \leftarrow M[j-1] \oplus C[j-1]$;
206     $x \leftarrow M[j]$
207     if $x \in$ domain$(\pi_t)$ then
208         $bad_1 \leftarrow$ true; [ $C[j] \leftarrow \pi_t(x)$; next ]
209     $y \xleftarrow{\$} \{0,1\}^n$
210     if $y \in$ range$(\pi_t)$ then
211         $bad_2 \leftarrow$ true; [ $y \xleftarrow{\$}$ corange$(\pi_t)$ ]
212     $\pi_t(x) \leftarrow y$;  $C[j] \leftarrow y$;  $t \leftarrow x \oplus y$
213     if $t \in \mathcal{T}$ then $bad_3 \leftarrow$ true
214     $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
215 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$
216 return $C$

250 procedure $D(C)$
251 $m \leftarrow |C|/n$;  $(\imath, \ell) \leftarrow$ find$(C;\ C_1, \ldots, C_i)$
252 $M[1..\ell] \leftarrow M_\imath[1..\ell]$
253 for $j \leftarrow \ell + 1$ to $m$ do
254     if $j = 1$ then $t \leftarrow 0^n$
255     else $t \leftarrow M[j-1] \oplus C[j-1]$;
256     $y \leftarrow C[j]$
257     if $y \in$ range$(\pi_t)$ then
258         $bad_1 \leftarrow$ true; [ $M[j] \leftarrow \pi_t^{-1}(y)$; next ]
259     $x \xleftarrow{\$} \{0,1\}^n$
260     if $x \in$ domain$(\pi_t)$ then
261         $bad_2 \leftarrow$ true; [ $x \xleftarrow{\$}$ codomain$(\pi_t)$ ]
262     $\pi_t(x) \leftarrow y$;  $M[j] \leftarrow x$;  $t \leftarrow x \oplus y$
263     if $t \in \mathcal{T}$ then $bad_3 \leftarrow$ true
264     $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
265 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$     [ Game $G_2$ ]
266 return $M$                                        Game $G_3$
```

```
300 procedure $E(M)$
301 $m \leftarrow |M|/n$;  $(\imath, \ell) \leftarrow$ find$(M;\ M_1, \ldots, M_i)$
302 $C[1..\ell] \leftarrow C_\imath[1..\ell]$
303 for $j \leftarrow \ell + 1$ to $m$ do $C[j] \xleftarrow{\$} \{0,1\}^n$
304 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$
305 return $C$

350 procedure $D(C)$
351 $m \leftarrow |C|/n$;  $(\imath, \ell) \leftarrow$ find$(C;\ C_1, \ldots, C_i)$
352 $M[1..\ell] \leftarrow M_\imath[1..\ell]$
353 for $j \leftarrow \ell + 1$ to $m$ do $M[j] \xleftarrow{\$} \{0,1\}^n$
354 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$
355 return $M$                                        Game $G_4$
```

```
400 procedure $E(M)$
401 $m \leftarrow |M|/n$;  $(\imath, \ell) \leftarrow$ find$(M;\ M_1, \ldots, M_i)$
402 $C[1..\ell] \leftarrow C_\imath[1..\ell]$
403 for $j \leftarrow \ell + 1$ to $m$ do
404     $P \leftarrow M[1..j-1]$; $x \leftarrow M[j]$; $y \xleftarrow{\$} \{0,1\}^n$
405     if $y \in$ range$(\pi_P)$ then
406         $bad \leftarrow$ true; [ $y \xleftarrow{\$}$ corange$(\pi_P)$ ]
407     $\pi_P(x) \leftarrow y$;  $C[j] \leftarrow y$
408 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$
409 return $C$

450 procedure $D(C)$
451 $m \leftarrow |C|/n$;  $(\imath, \ell) \leftarrow$ find$(C;\ C_1, \ldots, C_i)$
452 $M[1..\ell] \leftarrow M_\imath[1..\ell]$
453 for $j \leftarrow \ell + 1$ to $m$ do
454     $P \leftarrow M[1..j-1]$; $y \leftarrow C[j]$; $x \xleftarrow{\$} \{0,1\}^n$
455     if $x \in$ domain$(\pi_P)$ then
456         $bad \leftarrow$ true; [ $x \xleftarrow{\$}$ codomain$(\pi_P)$ ]
457     $\pi_P(x) \leftarrow y$;  $M[j] \leftarrow x$
458 $i \leftarrow i + 1$;  $(M_i, C_i) \leftarrow (M, C)$     Game $G_5$
459 return $M$                                        [ Game $G_6$ ]
```

Fig. 2. Games used in the proof of Theorem 3. Game G_2 includes the bracketed statements while game G_3 does not. Similarly, game G_6 includes the bracketed statements while game G_5 does not.

lemma, $\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]$ is at most the probability that game $\mathcal{A}$ manages to set one of the bad_j variables in game G_3. The crux of the proof is the following observation:

Claim: Every execution of game G_3 that sets flag bad_1 also sets flag bad_3.

In fact, flag bad_3 was introduced as a trick for bounding the probability that bad_1 gets set. The proof of the claim is as follows. Suppose we are executing adversary $\mathcal{A}$ with game G_2 and, at some point in time it happens that, at line 207, we have $x \in \text{domain}(\pi_t)$, so that bad_1 will get set in the following line. Fix the current values M, m, $(\imath, \ell)$, $C[1..\ell]$, t, and x. Now x belonging to domain(π_t) means that some triple (t, x, y) was already added into the set of triples that constitute the partial function π (that is, (t, x, y) is "in" π if we have set $\pi_t(x) = y$). This triple had to have been added to π at some earlier execution of line 212 or 262. We distinguish two possibilities: (t, x, y) is the *only* triple in π with this t-value; or else there are, already, at least two distinct triples (t, x, y), (t, x', y') with this particular t-value. In the latter case, when we added the temporally second of these two triples into π we checked, at line 213 or 263, if t was *already* in $\mathcal{T}$. It would have been, so bad_3 would already have been set. What remains is the case that (t, x, y) is the *only* triple in π with the given value t. Focus on the fact that $M[1] \cdots M[\ell]$ matches $M_\imath[1] \cdots M_\imath[\ell]$. Now if the latter string is *all* of $M_\imath$ then there were two prior times that t was generated: one is when (t, x, y) got added to π, and another is when the final t-value was generated in response to the $\imath$-th query—that is, when we executed the final statement at line 212 or 262. The temporally second of these t-producing events would have resulted in production of a t that was already in $\mathcal{T}$ and bad_3 would have been set. If, instead, $M[1] \cdots M[\ell] = M_\imath[1] \cdots M_\imath[\ell]$ and $M_\imath$ continues with at least one nonempty block $M_\imath[\ell+1]$, then we know that $M[\ell+1] \neq M_\imath[\ell+1]$ and the one and only triple in π with the given t-value must be $(t, M_\imath[\ell+1], C_\imath[\ell+1])$, so $x = M[\ell+1] \neq M_\imath[\ell+1]$ could not have caused line 207 to evaluate to true.

The case where bad_1 gets set on a decryption query, at line 258, is symmetric with the paragraph above: again bad_3 will already have been set. This completes the proof of the claim. □

Continuing, we now know that $\Pr[bad_1 \wedge bad_2 \wedge bad_3] = \Pr[bad_2 \wedge bad_3]$, which is at most $\Pr[bad_2] + \Pr[bad_3]$. The first probability is at most $0.5\,\sigma(\sigma-1)/2^n$ and the second is at most $0.5\,\sigma(\sigma+1)/2^n$ (recall that $\mathcal{T}$ was initially seeded with a point). We thus have that $\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1] \leq \sigma^2/2^n$.

Games G_3 and G_4 are easily seen to be adversarially indistinguishable; we have simply eliminated the pointless code. Games G_4 and G_5 are adversarially indistinguishable; here we are introducing using lazy sampling. Passing from G_5 to G_6 can be regarded as a form of the PRP/PRF switching lemma (cf. [4, Lemma 3.7]) and the probability that *bad* gets set to true in game G_6 is at most $0.5\,\sigma^2/2^n$. The theorem now follows. ▮

The complexity-theoretic analog easily follows. This time, we show how the theorem looks.

Corollary 1 (TC3 is ±oprp-secure). *Let* $\widetilde{E}\colon \mathcal{K} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *be a tweakable blockcipher. Let $\mathcal{A}$ be an adversary that runs in time t and asks queries totaling at most σ blocks. Then there exists an adversary $\mathcal{B}$ such that* $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\mathrm{TC3}[\widetilde{E}]}(\mathcal{B}) \geq \mathbf{Adv}^{\mathrm{oprp}}_{\mathcal{E}}(\mathcal{A}) - 1.5\,\sigma^2/2^n$. *Adversary $\mathcal{B}$ runs in time at most $t + cn\sigma$, for some absolute constant c, and asks at most σ queries.* ▮

Mode TC3, beyond being a natural simplification to the generalization to mode HCBC2 [4], is a generalization of Nandi's mode MHCBC [22]; the latter can be realized as a special case of TC3 by selecting the tweakable blockcipher

$\widetilde{E}\colon (\mathcal{K}_1 \times \mathcal{K}_2) \times \{0,1\}^n \to \{0,1\}^n$ to be $\widetilde{E}^T_{K1\,K2}(X) = E_{K1}(M \oplus H_{K2}(T)) \oplus H_{K2}(T)$ where $H\colon \mathcal{K}_2 \times \{0,1\}^n \to \{0,1\}^n$ is an almost-xor universal hash function and E is a blockcipher [17].

5 Online Ciphers for Arbitrary-Length Strings

We start out by extending the notation $M[i]$ so that, for each nonempty string M we have that $M = M[1] \cdots M[m-1]\, M[m]$ where $m = \lfloor |M|/n \rfloor$, $|M[i]| = n$ for all $1 \le i \le m-1$, and $n \le |M[m]| \le 2n-1$. In other words, when M is not a multiple of n bits its final block $M[m]$ is chosen to be *long*, having between $n+1$ and $2n-1$ bits. All other blocks remain n-bits in length. With this notation in hand we define $\text{Online}^*(n)$ to be the set of all length-preserving permutations $\pi\colon \{0,1\}^* \to \{0,1\}^*$ such that $\pi(M[1] \cdots M[i])$ depends only on $M[1] \cdots M[i]$ (for all $i \ge 1$). This set can be regarded as an idealized cipher, just like $\text{Perm}(n)$ and $\text{Online}(n)$. Now if $\mathcal{E}\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^*$ is an online cipher and $\mathcal{A}$ is an adversary we can extend our prior definitions by using $\text{Online}^*(n)$ in our reference experiment:

$$\mathbf{Adv}^{\text{oprp}}_{\mathcal{E}}(\mathcal{A}) = \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \text{Online}^*(n) : \mathcal{A}^{\pi} \Rightarrow 1]$$
$$\mathbf{Adv}^{\pm\text{oprp}}_{\mathcal{E}}(\mathcal{A}) = \mathbf{Pr}[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K,\, \mathcal{E}_K^{-1}} \Rightarrow 1] - \mathbf{Pr}[\pi \xleftarrow{\$} \text{Online}^*(n) : \mathcal{A}^{\pi,\, \pi^{-1}} \Rightarrow 1]$$

There is an alternative notion of security where, when M is not a multiple of n bits, the final block is *short* (having 1 to $n-1$ bits) instead of long (having $n+1$ to $2n-1$ bits). There are problems with this alternative notion. First, it is too weak. If the adversary learns $C = E_K(X \,\|\, 0)$, where $X \in (\{0,1\}^n)^+$, then it also knows $C' = E_K(X \,\|\, 1)$, which is just C with its final bit flipped. Second, despite this alternative notion being weak, instantiations are hard. This is because it is not known how to construct from an n-bit blockcipher an efficient and provably-secure cipher, with good bounds, for arbitrary input lengths less than n; see the literature on "format-preserving encryption" for a discussion of this problem [5]. While this short-string enciphering problem cannot be avoided if the original message M has fewer than n bits, there is no need to deal with it when $|M|$ has more than n bits, which, in applications, is likely to be most or all the time.

Now turning to constructions, let $\widetilde{E}\colon \mathcal{K} \times \{0,1\}^n \times \{0,1\}^{\le 2n-1} \to \{0,1\}^{\le 2n-1}$ be a tweakable cipher. From this primitive define the online cipher $\mathcal{E} = \text{TC3}^*[\widetilde{E}]$ with key space $\mathcal{K}$ and message space $\mathcal{M} = \{0,1\}^*$. See Fig. 3. The construction is CCA-secure, as formalized below. We will take up in the next section how one constructs a tweakable cipher with message space $\{0,1\}^{\le 2n-1}$.

Theorem 4 (TC3* is ±oprp-secure). *Let $\widetilde{\pi} = \text{Perm}(\{0,1\}^{\le 2n-1})$. If $\mathcal{A}$ asks queries having at most σ blocks then $\mathbf{Adv}^{\pm\text{oprp}}_{\text{TC3}^*[\widetilde{\pi}]}(\mathcal{A}) \le 1.5\,\sigma^2/2^n$.* ▮

We omit the proof since it is almost the same as that for Theorem 3.

```
TC3*
40  algorithm $\mathcal{E}_K(M)$
41  if $M = \varepsilon$ then return $\varepsilon$
42  $m \leftarrow \lfloor |M|/n \rfloor$;  $T \leftarrow 0^n$
43  for $j \leftarrow 1$ to $m-1$ do
44      $C[j] \leftarrow \widetilde{E}_K^T(M[j])$
45      $T \leftarrow M[j] \oplus C[j]$
46  $C[m] \leftarrow \widetilde{E}_K^T(M[m])$
47  return $C$
```

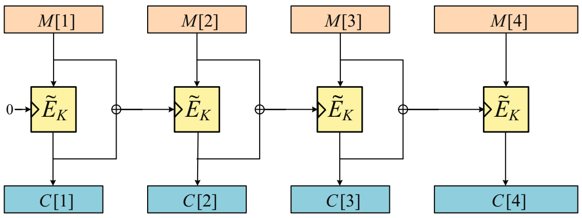


Fig. 3. Mode TC3*. The CCA-secure online cipher now takes an input of arbitrary bit length, but it depends on a richer primitive than does TC3: we start with a cipher $\widetilde{E}$: $\{0,1\}^{\leq 2n-1} \to \{0,1\}^{\leq 2n-1}$. The block input to $\widetilde{E}$ is "usually" n bits, but a single long-block call (up to $2n-1$ bits) will be used when $|M| \geq n$ and n doesn't divide $|M|$, while a single short-block call will be needed if $|M| < n$.

6 Instantiating the Schemes

Let us consider how to instantiate TC3 starting from a conventional (instead of a tweakable) blockcipher E: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. The simplest and most natural solution is to create the tweakable blockcipher by way of $\widetilde{E}_K^T(X) = E_{K_1}(X \oplus \Delta) \oplus \Delta$ where $\Delta = T \cdot K_2$ and $K = K_1 \,\|\, K_2$ for $|K_1| = k$ and $|K_2| = n$. Here multiplication, $T \cdot K_2$, is in GF(2^n), representing field points as n-bit strings in the usual way. We know that $\widetilde{E}$: $\mathcal{K} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ will be a CCA-secure tweakable PRP as long as E is a CCA-secure conventional PRP; this is the well-known construction from Liskov, Rivest, and Wagner [17, Theorem 2], together with the fact that multiplication in GF(2^n)—that is, $H_K(X) = K \cdot X$—is a 2^{-n}-AXU hash function.

The above construction is quite efficient, involving one blockcipher invocation and one GF(2^n) multiply for each message block. This is comparable to the work involved with the authenticated-encryption scheme GCM [19], which has, for example, been implemented by Gueron and Kounavis to run as fast as 3.54 cycles per byte [13] (on Intel processors supporting AES and PCLMULQDQ assembly instructions). This timing figure, however, would overestimate the expected speed of TC3, on similar hardware, since the blockcipher chaining in TC3 will decrease instruction-level parallelism.

Following Boldyreva and Taesombut, TC3 can be augmented, with little overhead, to provide a solution to the problem of blockwise-adaptive CCA-secure authenticated-encryption [9]. Doing so would give an AE scheme with efficiency roughly comparable to GCM but provably achieving a useful security property that GCM does not achieve.

Instantiating TC3* from a conventional blockcipher is more involved than instantiating TC3, as now we need a map $\widetilde{E}$: $\mathcal{K} \times \{0,1\}^n \times \{0,1\}^{\leq 2n-1} \to \{0,1\}^{\leq 2n-1}$. Actually our tweakable cipher will not have to deal with messages having fewer than n bits unless the higher-level construction $\mathcal{E}$ is itself asked to encipher messages of fewer than n bits, so let us put this case aside. Our problem then is to create from an ordinary n-bit blockcipher E a VIL-secure tweakable cipher that can encipher messages of n to $2n-1$ bits. Fortunately there are some ready solutions to this problem. Four or more rounds of Feistel would be the classical approach [18]. One would use a blockcipher-based, tweak-dependent round function. A different possibility is the EME2 cipher (formerly named EME*) of Halevi [14]; the mechanism was recently approved as the IEEE standard P1619.2-2010. The scheme is simple, provably secure, and, using five blockcipher calls and a modest amount of additional overhead, provides a tweakable and VIL cipher over the domain that we need. More efficient still would be the XLS construction of Ristenpart and Rogaway [24]. This can encipher strings of $n+1$ to $2n-1$ bits using three blockcipher calls and very little extra work.[4]

Enciphering strings of fewer than n bits takes special techniques. One proposal is FFX [7], which uses a conventional, unbalanced, or alternating Feistel network on these small domains. We note that if one is going to deal with short final blocks by a patchwork of techniques, one for strings in $\{0,1\}^{\leq n-1}$ and one for other strings that are not a multiple of n bits, it is important to use distinct keys, or to use other techniques, to provably ensure VIL security.

Acknowledgments

Many thanks for the perceptive comments from the anonymous referees. Our apologies that we have not expanded on points where this ought be done.

The authors gratefully acknowledge the support of NSF grant CNS 0904380.

References

1. Amanatidis, G., Boldyreva, A., O'Neill, A.: Provably-secure schemes for basic query support in outsourced databases. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 14–30. Springer, Heidelberg (2007)
2. Bard, G.: A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL. In: International Conference on Security and Cryptography, SECRYPT 2006, pp. 99–109. INSTICC Press (2006)
3. Bellare, M., Boldyreva, A., Knudsen, L., Namprempre, C.: On-line ciphers and the hash-CBC constructions. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 292–309. Springer, Heidelberg (2001)
4. Bellare, M., Boldyreva, A., Knudsen, L., Namprempre, C.: On-line ciphers and the hash-CBC constructions. Cryptology ePrint report 2007/197, June 29 (2007) Full version of [3]
5. 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)

[4] In fact, none of the methods just surveyed was designed specifically to create a tweakable cipher with n-bit tweaks and a message space of n to $2n-1$ bits. With this limited goal in mind we believe that provably-good methods related to those of Naor-Reingold [23] will give a cipher needing just two blockcipher calls for strings of $n+1$ to $2n-1$ bits.

6. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
7. Bellare, M., Rogaway, P., Spies, T.: The FFX mode of operation for format-preserving encryption (draft 1.1). NIST submission (February 2010); See also the addendum (September 2010) by the same authors
8. Bernstein, D., Schwabe, P.: New AES software speed records. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 322–336. Springer, Heidelberg (2008)
9. Boldyreva, A., Taesombut, N.: Online encryption schemes: new security notions and constructions. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 1–14. Springer, Heidelberg (2004)
10. Fouque, P., Joux, A., Martinet, G., Valette, F.: Authenticated on-line encryption. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 145–159. Springer, Heidelberg (2004)
11. Fouque, P., Joux, A., Poupard, G.: Blockwise adversarial model for on-line ciphers and symmetric encryption schemes. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 212–226. Springer, Heidelberg (2004)
12. Fouque, P., Martinet, G., Poupard, G.: Practical symmetric on-line encryption. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 362–375. Springer, Heidelberg (2003)
13. Gueron, S., Kounavis, M.: Intel carry-less multiplication instruction and its usage for computing the GCM mode (revision 2). White paper (May 2010), http://www.intel.com
14. 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)
15. Joux, A., Martinet, G., Valette, F.: Blockwise-adaptive attackers: revisiting the (in)security of some provably secure encryption models: CBC, GEM, IACBC. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 17–30. Springer, Heidelberg (2002)
16. Krawczyk, H.: LFSR-based hashing and authentication. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 129–139. Springer, Heidelberg (1994)
17. Liskov, M., Rivest, R., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
18. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal of Computing 17(2), 373–386 (1988)
19. McGrew, D., 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)
20. Meyer, C., Matyas, M.: Cryptography: A New Dimension in Data Security. John Wiley & Sons, New York (1982)
21. Nandi, M.: A simple security analysis of Hash-CBC and a new efficient one-key online cipher,Cryptology ePrint report 2007/158, May 7 (2007)
22. Nandi, M.: Two New Efficient CCA-secure online ciphers: MHCBC and MCBC. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 350–362. Springer, Heidelberg (2008); Also Cryptology ePrint report 2008/401 (September 20, 2008)
23. Naor, M., Reingold, O.: On the construction of pseudorandom permutations: Luby-Rackoff revisited. Journal of Cryptology 12(1), 29–66 (1999)
24. Ristenpart, T., Rogaway, P.: How to enrich the message space of a cipher. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 101–118. Springer, Heidelberg (2007)

Meet-in-the-Middle Attacks on Reduced-Round XTEA[*]

Gautham Sekar[**], Nicky Mouha[***], Vesselin Velichkov[†], and Bart Preneel

Department of Electrical Engineering ESAT/SCD-COSIC,
Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
and
Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium
{Gautham.Sekar,Nicky.Mouha,Vesselin.Velichkov,
Bart.Preneel}@esat.kuleuven.be

Abstract. The block cipher XTEA, designed by Needham and Wheeler, was published as a technical report in 1997. The cipher was a result of fixing some weaknesses in the cipher TEA (also designed by Wheeler and Needham), which was used in Microsoft's Xbox gaming console. XTEA is a 64-round Feistel cipher with a block size of 64 bits and a key size of 128 bits. In this paper, we present meet-in-the-middle attacks on twelve variants of the XTEA block cipher, where each variant consists of 23 rounds. Two of these require only 18 known plaintexts and a computational effort equivalent to testing about 2^{117} keys, with a success probability of $1 - 2^{-1025}$. Under the standard (single-key) setting, there is no attack reported on 23 or more rounds of XTEA, that requires less time and fewer data than the above. This paper also discusses a variant of the classical meet-in-the-middle approach. All attacks in this paper are applicable to XETA as well, a block cipher that has not undergone public analysis yet. TEA, XTEA and XETA are implemented in the Linux kernel.

Keywords: Cryptanalysis, block cipher, meet-in-the-middle attack, Feistel network, XTEA, XETA.

1 Introduction

Timeline: The TEA family of block ciphers

- **1994.** The cipher TEA (Tiny Encryption Algorithm) is a 64-round Feistel cipher that operates on 64-bit blocks and uses a 128-bit key. Designed by

[*] This work was supported in part by the Research Council K.U.Leuven: GOA TENSE, and by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.
[**] This author is supported by an FWO project.
[***] This author is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).
[†] DBOF Doctoral Fellow, K.U.Leuven, Belgium.

Wheeler and Needham, it was presented at FSE 1994 [24]. Noted for its simple design, the cipher was subsequently well studied and came under a number of attacks.

- **1996.** Kelsey *et al.* established that the effective key size of TEA was 126 bits [12]. This result led to an attack on Microsoft's Xbox gaming console where TEA was used as a hash function [23].
- **1997.** Kelsey, Schneier and Wagner constructed a related-key attack on TEA with 2^{23} chosen plaintexts and 2^{32} time [13]. Following these results, TEA was redesigned by Needham and Wheeler to yield Block TEA and XTEA (eXtended TEA) [18]. While XTEA has the same block size, key size and number of rounds as TEA, Block TEA caters to variable block sizes for it applies the XTEA round function for several iterations. Both TEA and XTEA are implemented in the Linux kernel.
- **1998.** To correct weaknesses in Block TEA, Needham and Wheeler designed Corrected Block TEA or XXTEA, and published it in a technical report [19]. This cipher uses an unbalanced Feistel network and operates on variable-length messages. The number of rounds is determined by the block size, but it is at least six. An attack on the full Block TEA is presented in [20], where some weaknesses in XXTEA are also detailed.
- **2002–2010.** A number of cryptanalysis results on the TEA family were reported in this period. Table 1 lists the attacks on XTEA and their complexities. In [11], it was shown that an ultra-low power implementation of XTEA might be better suited for low resource environments than AES. Note that XTEA's smaller block size also makes it advantageous if an application requires fewer than 128 bits of data to be encrypted at a time.

The meet-in-the-middle attack. The meet-in-the-middle attack was first introduced by Diffie and Hellman in 1977 [5]. Since then, this technique and its variants have been successfully used against several block ciphers, including reduced-round DES [4,6] and the full KeeLoq [10]. Unlike Diffie and Hellman's original attack, the meet-in-the-middle attacks in this paper[1] have negligible memory requirements.

We denote the message space and the key space by $\mathcal{M}$ and $\mathcal{K}$ respectively. Now consider two block ciphers $A_K, B_K : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{M}$ and let $Y_K = B_K \circ A_K$, where $\circ$ denotes function composition. In a meet-in-the-middle attack, the adversary deduces K from a given plaintext-ciphertext pair (p, c), where $c = Y_K(p)$, by solving the equation

$$A_K(p) = B_K^{-1}(c) \ . \quad (1)$$

Contribution of this paper. This paper presents meet-in-the-middle attacks on block ciphers with 7, 15 and 23 rounds of XTEA. Our attacks are under

[1] The attack presented in Sect. 5 of this paper can also be seen as a meet-in-the-middle attack, however the (partial) encryptions and decryptions cannot be performed over all rounds, as the attacker only searches exhaustively over parts of the key. We therefore use a technique similar to the partial matching technique of Sasaki and Aoki. This very recent technique was successfully applied to several hash functions, including MD4 [2], MD5 [21], HAS-160 [9] and SHA-2 [1].

Table 1. Key recovery attacks on XTEA where the time complexities are averages, if explicitly stated in the original paper, average success probabilities are given as well (KP: known plaintext, CP: chosen plaintext, RK: in a related-key setting)

Attack	**Ref.**	**# rounds**	**Time**	**Data**	**Pr(Success)**
• Attacks in the standard (single-key) setting					
Meet-in-the-middle	**This paper**	**7**	$\mathbf{2^{95.00}}$	**2 KPs**	$\mathbf{1-2^{-33}}$
Impossible differential	[17]	14	2^{85}	$2^{62.5}$ CPs	Not given
Differential	[8]	15	2^{120}	2^{59} CPs	Not given
Meet-in-the-middle	**This paper**	**15**	$\mathbf{2^{95.00}}$	**3 KPs**	$\mathbf{1-2^{-65}}$
Truncated differential	[8]	23	$2^{120.65}$	$2^{20.55}$ CPs	0.969
Meet-in-the-middle	**This paper**	**23**	$\mathbf{2^{117.00}}$	**18 KPs**	$\mathbf{1-2^{-1025}}$
• Attacks in a related-key setting					
Related-key truncated differential	[14]	27	$2^{115.15}$	$2^{20.5}$ RK-CPs	0.969
Related-key rectangle (for $2^{108.21}$ weak keys)	[15]	34	$2^{31.94}$	2^{62} RK-CPs	Not given
Related-key rectangle	[16]	36	$2^{126.44}$	$2^{64.98}$ RK-CPs	0.63
Related-key rectangle (for $2^{110.67}$ weak keys)	[16]	36	$2^{104.33}$	$2^{63.83}$ RK-CPs	0.80
Related-key	[3]	37	2^{125}	2^{63} RK-CPs	Not given
Related-key (for $2^{107.5}$ weak keys)	[3]	51	2^{123}	2^{63} RK-CPs	Not given

the standard setting, giving the attacker less freedom than under a related-key setting. In Table 1, we see that there is no attack on 23 or more rounds of XTEA, that is better than ours given the standard setting. Furthermore, each of our attacks requires only a few *known plaintexts*, whereas every attack listed in Table 1 requires many *chosen plaintexts*.

The Linux kernel not only includes XTEA, but also a variant called XETA [7]. The cipher XETA resulted from a bug in the C implementation of XTEA, where higher precedence was incorrectly given to exclusive-OR over addition in the round function. From this paper, it is easy to verify that all our results to XTEA directly apply to XETA as well. This is because our attacks exploit weaknesses in the key schedule, which is the same for both XTEA and XETA. To the best of our knowledge, this paper is the first to give cryptanalysis results on XETA.

Organization. This paper is organized as follows. Section 2 lists the notation and conventions that we follow. The description of XTEA is provided in Sect. 3. Our main observation is presented in Sect. 4 and it is developed into an attack on 15-round XTEA in Sect. 5. Here, we also provide other sets of 15 rounds that could be similarly attacked. Section 6 describes our attack on 23 rounds on

XTEA and provides other sets of 23 rounds that could be attacked in a similar way. Section 7 concludes the paper and provides an interesting open problem. In Appendix A, we show which countermeasures can be introduced to XTEA to prevent all the attacks in this paper. The 23-round attack is illustrated in Appendix B.

2 Notation and Conventions

The notation and conventions used in this paper are listed in Table 2.

Table 2. Notation

Symbol / Notation	**Meaning**
$\boxplus$	Addition modulo 2^{32}
$\oplus$	Exclusive-OR
$\ll$	Left shift
$\gg$	Right shift
$\|\|$	Concatenation
$\lfloor x \rfloor$	$\max_{y\in\mathbb{Z}}(y \leq x)$, $\mathbb{Z}$ is the set of integers
LSB	Least significant bit
MSB	Most significant bit
$[i]$	Select bit i, $i = 0$ is the LSB
$[j \ldots i]$	Select bits k where $j \geq k \geq i$, $k = 0$ is the LSB
0^k	Concatenation of k times the string '0'

3 Description of XTEA

The block cipher XTEA has block size of 64 bits and key size of 128 bits. It uses a 64-round Feistel network (see Fig. 1). The F-function of the Feistel network (see Fig. 2) takes a 32-bit input x and produces a 32-bit output as:

$$F(x) = ((x \ll 4) \oplus (x \gg 5)) + x \ . \tag{2}$$

The 128-bit key K of XTEA is divided into four 32-bit subkeys $K_0, \ldots, K_3$. At every round, one of the 4 subkeys is selected according to a key schedule. A constant $\delta = \lfloor(\sqrt{5}-1)\cdot 2^{31}\rfloor$ is defined, derived from the golden ratio. Two bits from a different multiple of δ are used at every round as the index of the subkey. The 32-bit subkey α_t used in round t, where $1 \leq t \leq 64$, is chosen from the set $\{K_0, K_1, K_2, K_3\}$ according to the following rule:

$$\alpha_t \leftarrow \begin{cases} K_{\delta_t[1\ldots0]} & \text{if } t \text{ is odd} \ , \\ K_{\delta_t[12\ldots11]} & \text{if } t \text{ is even} \ , \end{cases} \tag{3}$$

where

$$\delta_t = \left\lfloor \frac{t}{2} \right\rfloor \delta, \quad 1 \leq t \leq 64 \ . \tag{4}$$

The 64-bit input to round t of XTEA consists of two 32-bit parts L_{t-1} and R_{t-1} (see Fig. 1). For round 1, the plaintext p is used as input: $(L_0 \parallel R_0) \leftarrow p$. The input for round $t+1$ is computed recursively from the input to round t as given by:

$$L_t \leftarrow R_{t-1} \ , \tag{5}$$

$$R_t \leftarrow L_{t-1} \boxplus ((\delta_t \boxplus \alpha_t) \oplus F(R_{t-1})) \ , \tag{6}$$

where α_t is selected according to (3). For reference, we also list the subkeys used in every round in Table 3.

The ciphertext c of XTEA is produced by concatenating the two parts obtained after the 64th round: $c \leftarrow L_{64} \parallel R_{64}$.

Finally, we note that in the description above by *round* we mean a *Feistel round*. This is not to be confused with the term *cycle* used in the original proposal of XTEA [18]. A cycle is equivalent to two Feistel rounds. Therefore XTEA has 64 rounds or 32 cycles.

Table 3. Subkeys used in XTEA

Rounds	**Subkey used**
$1, 8, 9, 10, 17, 18, 20, 25, 30, 33, 40, 41, 49, 50, 57, 60$	K_0
$3, 6, 11, 16, 19, 26, 27, 28, 35, 36, 38, 43, 46, 48, 51, 58, 59$	K_1
$4, 5, 13, 14, 21, 24, 29, 34, 37, 44, 45, 53, 54, 56, 61, 64$	K_2
$2, 7, 12, 15, 22, 23, 31, 32, 39, 42, 47, 52, 55, 62, 63$	K_3

4 Motivational Observation

We begin by observing that the subkey K_2 is not used in rounds 6–12. For the remainder of this section, let $K \leftarrow (K_0, K_1, X, K_3)$, where X can be any 32-bit value, as subkey K_2 is irrelevant in the analysis. Given one plaintext-ciphertext pair (p_0, c_0), with each key guess, the attacker checks whether

$$E_K^{(6\ldots12)}(p_0) = c_0 \ , \tag{7}$$

where $E_K^{(6\ldots12)}$ denotes the 7-round (rounds 6–12) encryption using the key K. At first glance, it may appear that 1 KP is sufficient. However, it is to be noted that the key space (2^{96} keys K) is larger than the ciphertext space (2^{64} ciphertext blocks).

We now show that obtaining a second KP (p_1, c_1) is sufficient for an attack with an average time complexity of $2^{95.00}$ 7-round encryptions and an average success probability of $1 - 2^{-33}$. The attacker iterates over the 2^k keys K, where $k = 96$. For every candidate key K, (7) is tested using the first KP. If this equality is satisfied, the second KP is used to check

$$E_K^{(6\ldots12)}(p_1) = c_1 \ . \tag{8}$$

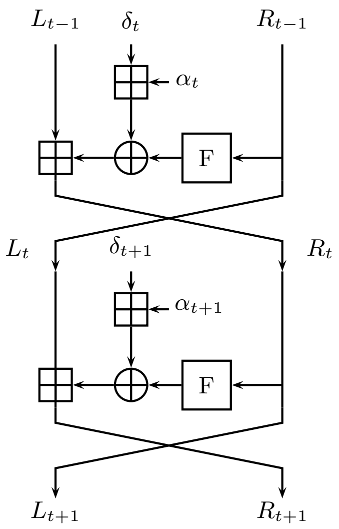


Fig. 1. The Feistel structure of XTEA showing two rounds

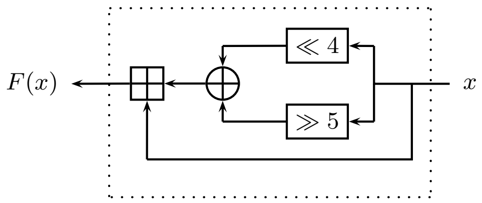


Fig. 2. The function F used in the round function of XTEA

If either (7) or (8) is not satisfied, the candidate key K is incorrect and can be *sieved*. The approximate number of plaintext-ciphertext pairs that are needed can also be estimated from Shannon's unicity distance [22].

We make the reasonable assumption throughout this paper, that every block cipher we consider has perfect confusion and diffusion properties [22]. If either the plaintext or the key, or both are changed, it is assumed that the corresponding ciphertext will be generated uniformly at random, independent from previously obtained ciphertexts.

Under this assumption, each of the 64-bit conditions that result from (7) and (8) is satisfied with probability 2^{-64}. All time complexities are stated as the number of equivalent encryptions of the reduced-round block cipher.

The average success probability can be calculated as follows. The two 64-bit conditions are simultaneously satisfied with probability $2^{-2\cdot 64} = 2^{-128}$. We can therefore eliminate a wrong key with probability $1 - 2^{-128}$. Assume that key i is the correct key, where $0 \leq i < 2^k$. It will be output by the algorithm if all previous keys are eliminated. This happens with probability $(1 - 2^{-128})^i$. The

Table 4. All 7-round attacks; each attack requires 2 KPs and on average $2^{95.00}$ 7-round encryptions for an average success probability of $1 - 2^{-33}$

Cipher consisting of XTEA rounds	**Unused subkey**
6–12	K_2
24–30	K_3
42–48	K_0
46–52	K_2

correct key can be located anywhere among the list of 2^k candidate keys with equal probability. Therefore, the average success probability is

$$2^{-k} \cdot \sum_{i=0}^{2^k-1} (1 - 2^{-128})^i = 2^{128-k} \cdot (1 - (1 - 2^{-128})^{2^k}) \approx 2^{128-k} \cdot (1 - e^{-2^{k-128}}) \approx 1 - 2^{-33} \ . \tag{9}$$

The approximations result from using the first and the second order Taylor approximations of e^x around 0. We now calculate the time complexity of the attack. For a candidate key K to be determined as wrong, the expected number of trials is $1 + 2^{-64}$. This is because for every key, (7) is always checked, and for 2^{-64} keys (8) is checked as well. If the candidate key is correct, two encryptions are always performed. As the correct key can be located anywhere in the list of 2^k candidates keys with equal probability, the average number of encryptions of the algorithm is

$$2^{-k} \cdot \sum_{i=0}^{2^k-1} \big(i \cdot (1 + 2^{-64}) + 2\big) = 2^{-1} \cdot (1 + 2^{-64}) \cdot (2^k - 1) + 2 \approx 2^{95.00} \ . \tag{10}$$

From Table 3, we obtain several other 7-round block ciphers that can be attacked in a similar way. Table 4 lists all such ciphers. Finally, we note that for $n = 0$ and $n = 1$ respectively, one can replace both (7) and (8) with

$$E_K^{(6\ldots r-1)}(p_n) = D_K^{(r\ldots 12)}(c_n) \ , \tag{11}$$

where $r \in \{6, \ldots, 12\}$, $E_K^{(6\ldots 5)}(p_n) = p_n$, and $D_K^{(r\ldots 12)}$ denotes (13-r)-round (rounds r–12) decryption using the key K. Therefore, what we essentially constructed above can be viewed as meet-in-the-middle attacks. In (11), the value of r determines the subkeys that are required for encryption and decryption.

5 Attacks on 15 Rounds of XTEA

The attack described in Sect. 4 on rounds 6–12, can be extended to rounds 6–20 as follows. First, the attacker performs a meet-in-the-middle attack, where (partial) encryptions and decryptions cannot be performed over all rounds, the attacker only exhaustively searches over part of the key. From the remaining rounds, however, the number of possibilities for the full key is reduced. Only three known plaintexts (p_n, c_n), $0 \leq n < 2$ are required for the attack.

Let us now split a reduced-round XTEA block cipher into *outer rounds* and *inner rounds*. In the outer rounds, one particular subkey is not used, whereas the inner rounds use only this subkey. The attack is described for rounds 6–20. As can be seen from Table 3, the outer rounds (6–12) and (15–20) do not involve K_2, whereas the two inner rounds (13–14) use only K_2.

By encrypting plaintext p_0 from round 6 to round 12 (i.e., until the beginning of round 13) and decrypting the corresponding ciphertext c_0 for 6 rounds starting backwards from round 20, we obtain the subkeys used in the inner rounds. They are denoted as $K_2^{'}$ and $K_2^{''}$ for inner rounds 13 and 14 respectively. Then, the attacker checks whether $K_2^{'} = K_2^{''}$. This can be understood from Fig. 1. Therefore, not the ciphertext values (as in Sect. 4), but the key values "meet in the middle". To the best of our knowledge, such an approach has not been described in previous literature.

If $K_2^{'} \neq K_2^{''}$, the candidate key of (K_0, K_1, K_3) cannot be correct, and the attacker proceeds to the next candidate key. Otherwise, the candidate key is extended to $(K_0, K_1, K_2,\ K_3)$, where $K_2 = K_2^{'} = K_2^{''}$. Then, the meet-in-the-middle attack is performed as described in Sect. 4. That is, a plaintext is encrypted with candidate keys $(K_0, K_1, K_2,\ K_3)$, to check which of the computed ciphertexts agrees with the actual (corresponding) ciphertext. For the 15-round attack, it is sufficient to use two additional known plaintexts (p_1, c_1) and (p_2, c_2).

The average success probability can be calculated as follows. Using (p_0, c_0) a 32-bit condition is obtained when $K_2^{'} = K_2^{''}$ is checked. Then, (p_1, c_1) and (p_2, c_2) each gives an additional 64-bit condition. A wrong key will pass these tests with probability[2] $2^{-32} \cdot \left(2^{-64}\right)^2 = 2^{-160}$. Thus, with probability $1 - 2^{-160}$, a wrong key is eliminated. Assume that i is the correct key, where $0 \leq i < 2^k$. It will be output by the algorithm if all previous keys are eliminated. This happens with probability $(1 - 2^{-160})^i$. The correct key can be located anywhere among the list of 2^k candidate keys with equal probability. The average success probability is

$$2^{-96} \cdot \sum_{i=0}^{2^{96}-1} (1 - 2^{-160})^i = 2^{160-96} \cdot (1 - (1 - 2^{-160})^{2^{96}}) \approx 2^{64} \cdot (1 - e^{-2^{64}}) \approx 1 - 2^{-65} \ . \tag{12}$$

We now calculate the time complexity of the attack. For a candidate key (K_0, K_1, K_3) to be determined as wrong, the expected number of trials is $1 + 2^{-32} + 2^{-96}$. This is because for every candidate key (K_0, K_1, K_3), the attacker always checks whether $K_2^{'} \neq K_2^{''}$. For 2^{-32} and 2^{-96} candidate keys, the attacker encrypts using the second and third known plaintext respectively. If the

[2] If the texts obtained by encrypting p_0 and decrypting c_0, in the 13 outer rounds, are uniformly distributed at random, then so are the subkeys $K_2^{'}$ and $K_2^{''}$. This fact, explained in Appendix C, is explicitly stated here because the assumption of perfect confusion and diffusion was made for ciphertexts, and not for subkeys.

candidate key is correct, the equivalent of three encryptions is always performed. As the correct key can be located anywhere in the list of 2^{96} candidates keys with equal probability, the average number of (equivalent) encryptions of the algorithm is

$$2^{-96} \cdot \sum_{i=0}^{2^{96}-1} \left(i \cdot (1 + 2^{-32} + 2^{-96}) + 3\right) = 2^{-1} \cdot (1 + 2^{-32} + 2^{-96}) \cdot (2^{96} - 1) + 3 \approx 2^{95.00} \ . \tag{13}$$

Finally, in Table 5, we provide a list of all 15-round block ciphers that can be attacked with the same complexity.

Table 5. All 15-round attacks; each attack requires 3 KPs and on average $2^{95.00}$ computations of the 15 rounds for an average success probability of $1 - 2^{-65}$

Cipher consisting of XTEA rounds	Inner rounds	Inner round subkey
6–20	13,14	K_2
16–30	22,23	K_3
24–38	31,32	K_3
34–48	40,41	K_0
38–52	44,45	K_2
42–56	49,50	K_0

6 Attacks on 23 Rounds of XTEA

In this section, we extend the 15-round attack of Sect. 5 to 23 rounds. This 23-round attack has an average time complexity of $2^{117.00}$ (equivalent) encryptions and an average success probability of $1 - 2^{-1025}$. It requires only 18 known (not chosen) plaintexts and corresponding ciphertexts. For the same number of rounds, both the time complexity and the data complexity of our attack are much lower than those in [8]. Our attack is therefore the best attack on 23-round XTEA so far in the standard setting, and the only attack requiring such a low number of plaintexts and corresponding ciphertexts. We note that we have optimized our attack to have the time complexity as low as possible. It is possible to reduce the number of known plaintexts even further, but not without increasing the time complexity of the attack.

The technique used is a meet-in-the-middle attack, similar to the attacks in [4]. As in Sect. 5, the reduced-round XTEA block cipher is split into *outer rounds* and *inner rounds*. In the outer rounds, one subkey is not used. The inner rounds can contain any of the subkeys. Our attack applies to rounds 16–38 of XTEA. Rounds 16–21 and 33–38 are the outer rounds, and do not involve subkey K_3. The inner rounds are rounds 22–32. The attack is a *sieving attack*, as the correct key is found by eliminating keys that lead to contradictions. The attack is given in Algorithm 1.

The k-bit key is recovered in two stages. First, the attacker exhaustively searches over k_1 bits of the key K and use m known plaintexts to check a one-bit condition that each of the m plaintexts yield. These k_1 bits consist of K_0, K_1, K_2, and the 21 least significant bits of K_3. This one-bit condition, tested in `test_keys_1(`K`)`, results from the following observation, also illustrated in Appendix B. We see that, without using $K_3[31 \ldots 21]$, the attacker can calculate $L_{27}[0] \leftarrow E_K^{(16\ldots27)}(p)[0]$, and $L_{27}^{'}[0] \leftarrow D_K^{(28\ldots38)}(c)[0]$. As $L_{27}[0] = L_{27}^{'}[0]$ always holds if the candidate key K is correct, a wrong key can be discarded if $L_{27}[0] \neq L_{27}^{'}[0]$. Note that only k_1 bits of the candidate key K are used to test this condition, as the remaining k_2 bits do not affect this condition.

If none of the m plaintexts cause a key to be discarded, the attacker exhaustively searches over the remaining k_2 bits of key K in `test_keys_2(`K`)`. These k_2 bits are the 11 most significant bits of K_3. In this stage, $\ell \leq m$ of the m plaintexts are reused. Now, $(L_{27}, R_{27}) \leftarrow E_K^{(16\ldots27)}(p)$ and $(L_{27}^{'}, R_{27}^{'}) \leftarrow D_K^{(28\ldots38)}(c)$ are recalculated using the full key K. For efficiency, this calculation is sped up by using stored values $p_n^{\star}$ and $c_n^{\star}$ for the outer rounds, and encrypting only the inner rounds. Equations $L_{27} = R_{27}$ and $L_{27}^{'} = R_{27}^{'}$ yield only 63-bit conditions, as $L_{27}[0] = L_{27}^{'}[0]$ was already tested. If both equations are satisfied for all ℓ plaintexts, the candidate key K is output as the correct key, and the algorithm halts.

Let us now determine the average time complexity and the average success probability of Algorithm 1.

The algorithm succeeds if no wrong key K that passes all $m + \ell$ tests is encountered before the correct key. How efficiently the attacker searches through these candidate keys K, does not influence the success probability of Algorithm 1. We therefore assume that the exhaustive search is over 2^k keys, and then both `test_keys_1(`K`)` and `test_keys_2(`K`)` are performed for each of these keys.

Each of the m plaintexts yields a one-bit condition in `test_keys_1(`K`)`, satisfied randomly with a probability of 2^{-1}. When $\ell \leq m$ of these plaintexts are reused in `test_keys_2(`K`)`, there is a condition on the 63 remaining bits, satisfied randomly with a probability of 2^{-63}. A wrong key will be detected if at least one of the $m + \ell$ tests fail. This eliminates a wrong key with a probability of $1 - 2^{-m} \cdot 2^{-63\ell}$. Assume that i is the correct key, where $0 \leq i < 2^k$. Then, it will be output by the algorithm if all previous candidate keys lead to contradictions. This happens with probability $(1 - 2^{-m} \cdot 2^{-63\ell})^i$. As the correct key can be located anywhere in the list of 2^k candidate keys with equal probability, the average success probability of the algorithm is

$$2^{-k} \cdot \sum_{i=0}^{2^k-1} (1 - 2^{-m} \cdot 2^{-63\ell})^i = 2^{m+63\ell-k} \cdot (1 - (1 - 2^{-m-63\ell})^{2^k})$$
$$\approx 2^{m+63\ell-k} \cdot (1 - e^{-2^{k-m-63\ell}}) \ . \qquad (14)$$

We now calculate the time complexity of the attack. Let i and j (where $0 \leq i < 2^{k_1}$ and $0 \leq j < 2^{k_2}$) be parts of the correct key K^c where $i = (K_0^c, K_1^c, K_2^c, K_3^c[20 \ldots 0])$ and $j = K_3^c[31 \ldots 21]$. Any 117-bit key $(K_0, K_1, K_2, K_3\ [20 \ldots 0])$, tested

Algorithm 1. Recovering the key of the 23-round XTEA block cipher consisting of rounds 16–38; an average $2^{117.00}$ (equivalent) encryptions and 18 KPs are required for an average success probability of $1 - 2^{-1025}$

```
Require: $m$ known plaintexts $p_0 \dots p_{m-1}$ and corresponding ciphertexts $c_0 \dots c_{m-1}$ .
Ensure: The output key $K$ (of length $k$ bits) is the correct key with probability
    $2^{m+63\ell-k}(1 - e^{-2^{k-m-63\ell}})$, where $\ell$ is chosen such that $\ell \leq m$.
 1: global $p_0^\star \dots p_{m-1}^\star$, $c_0^\star \dots c_{m-1}^\star$ .
 2: function test_key_1($K$) do
 3:    for $n \leftarrow 0 \dots m-1$ do
 4:       $p_n^\star \leftarrow E_K^{(16\dots21)}(p_n)$
 5:       $c_n^\star \leftarrow D_K^{(33\dots38)}(c_n)$
 6:       $(L_{27}, R_{27}) \leftarrow E_K^{(22\dots27)}(p_n^\star)$
 7:       $(L'_{27}, R'_{27}) \leftarrow D_K^{(28\dots32)}(c_n^\star)$
 8:       if $L_{27}[0] \neq L'_{27}[0]$ then
 9:          return false
10:    return true
11: function test_key_2($K$) do
12:    for $n \leftarrow 0 \dots \ell-1$ do
13:       $(L_{27}, R_{27}) \leftarrow E_K^{(22\dots27)}(p_n^\star)$
14:       $(L'_{27}, R'_{27}) \leftarrow D_K^{(28\dots32)}(c_n^\star)$
15:       if $L_{27} \neq L'_{27}$ or $R_{27} \neq R'_{27}$ then
16:          return false
17:    return true
18: for $(K_0, K_1, K_2) \leftarrow (0 \dots 2^{32}-1, 0 \dots 2^{32}-1, 0 \dots 2^{32}-1)$ do
19:    for $K_3[20 \dots 0] \leftarrow 0 \dots 2^{21}-1$ do
20:       $K \leftarrow (K_0, K_1, K_2, 0^{11} \parallel K_3[20 \dots 0])^\dagger$
21:       if test_key_1($K$) then
22:          for $K_3[31 \dots 21] \leftarrow 0 \dots 2^{11}-1$ do
23:             if test_key_2($K$) then
24:                output $K$ and halt
```

†Since the 11 bits $K_3[31 \dots 21]$ do not affect $L_{27}[0]$ or $L'_{27}[0]$, one can have any value β from the set $\{1, \dots, 2^{11}-1\}$ in place of 0^{11}. We have used 0^{11} for ease of understanding how the attack works.

in `test_keys_1(`K`)` before the correct key, passes `test_keys_1(`K`)` with probability 2^{-m}. Therefore, of the i 117-bit keys tested before the correct key, $i \cdot 2^{-m}$ keys are expected to pass `test_keys_1(`K`)`. For each of these $i \cdot 2^{-m}$ keys, `test_keys_2()` is performed 2^{k_2} times. Summarizing,

– the attacker performs an expected $i \cdot T_1$ 23-round computations, where T_1 is the expected number of 23-round computations for a wrong key under `test_keys_1()`;
– the attacker additionally performs an expected $i \cdot 2^{-m} \cdot 2^{k_2} \cdot T_2$ 23-round computations, where T_2 is the expected number of 23-round computations for a wrong key under `test_keys_2()`.

It is easy to see that

$$T_1 \triangleq \sum_{i=0}^{m-1} 2^{-i} \ . \tag{15}$$

To compute T_2, note that `test_keys_2()` only encrypts the 11 inner rounds again, and uses stored values for (partial) encryptions and decryptions of the outer rounds. This is equivalent to 11/23 encryptions of the 23-round block cipher and therefore

$$T_2 \triangleq \frac{11}{23} \cdot \sum_{j=0}^{\ell-1} 2^{-63j} \ . \tag{16}$$

For the correct (partial) key i, the number of steps under `test_keys_1()` is m. To determine the remaining part of the correct 128-bit key K^c, the attacker performs an expected $j \cdot T_2 + (11/23) \cdot \ell$ 23-round computations, where

1. $j \cdot T_2$ is the expected number of 23-round computations, under `test_keys_2()`, for all the j wrong (partial) keys preceding key j;
2. ℓ is the number of 11-round steps under `test_keys_2()` for the correct key j.

As the correct key j can take any value in the set $\{0, \ldots, 2^{k_2} - 1\}$, the average number of 23-round computations corresponding to the correct key i, is

$$2^{-k_2} \cdot \sum_{j=0}^{2^{k_2}-1} \left(j \cdot T_2 + \frac{11}{23} \cdot \ell \right) \ . \tag{17}$$

As the correct key i can take any value in the set $\{0, \ldots, 2^{k_1} - 1\}$, the average number of 23-round computations in total is

$$2^{-k_1} \cdot \sum_{i=0}^{2^{k_1}-1} \left(i \cdot T_1 + m + i \cdot 2^{-m} \cdot 2^{k_2} \cdot T_2 + 2^{-k_2} \cdot \sum_{j=0}^{2^{k_2}-1} \left(j \cdot T_2 + \frac{11}{23} \cdot \ell \right) \right) \tag{18}$$

The derivation of (18) will be more clear from Fig. 3 in Appendix B.

We now choose the parameters m and ℓ for the attack on rounds 16–38. From (18), we find that we cannot lower the average time complexity below $2^{117.00}$. Therefore, we choose m and ℓ such that we have the lowest number of known plaintexts, and the highest success probability for this particular time complexity. Setting $m = \ell = 18$, we find that 18 KPs are sufficient, and that the corresponding success probability using (14) is $1 - 2^{-1025}$. Note that the success probability of exhaustive search over the full k-bit key using 18 KPs has the same success probability. This shows that all KPs are optimally used in our attack from an information theoretic point of view [22]. Note that the number of KPs can still be lowered further, but then the time complexity must increase. This can be done by either increasing ℓ (which would make the second stage dominate in the attack), or by increasing k_1 (and thus perform the meet-in-the-middle on more than one bit).[3] We do not consider such options, as the number of KPs in our attack is already low enough for a practical attack. The time complexity, however, is still beyond reach with current hardware. Each of these attacks requires only negligible memory (about $4 \cdot 64 \cdot 18 = 2^{12.17}$ bits to store (p_n, c_n) and $(p_n^\star, c_n^\star)$ values).

As shown in Table 6, a total of 12 variants of the XTEA block cipher can be attacked, where each variant consists of 23 rounds. For rounds 34–56, the attack works in exactly the same way as for 16–38, and has the same complexities. The 10 other attacks require that $k_1 = 122$: the exhaustive search is now over all but the 6 most significant bits of one subkey in Algorithm 1, in order to obtain a condition on one bit to perform the meet-in-the-middle. The *middle bit* involved in this condition is given as well in Table 6.

Using (18), we calculate the time complexity for the 10 attacks that use 12 or 13 inner rounds. The lowest possible average time complexity for our attack strategy is $2^{122.00}$. For this time complexity, the best parameters are $m = \ell = 13$. We then obtain an average success probability of $1 - 2^{-705}$, using 13 KPs. Again, each of these attacks requires only negligible memory (about $2^{11.70}$ bits to store (p_n, c_n) and $(p_n^\star, c_n^\star)$ values).

Table 6. All 23-round attacks

Total rounds	**Inner rounds**	**Middle bit**	**Unused key bits**	**# Inner rounds**
16–38	22–32	$L_{27}[0]$	$K_3[31\dots21]$	11 rounds
34–56	40–50	$L_{45}[0]$	$K_0[31\dots21]$	11 rounds
6–28	13–24	$L_{19}[0]$	$K_2[31\dots26]$	12 rounds
8–30	12–23	$L_{18}[0]$	$K_3[31\dots26]$	12 rounds
24–46	31–42	$L_{37}[0]$	$K_3[31\dots26]$	12 rounds
26–48	30–41	$L_{36}[0]$	$K_0[31\dots26]$	12 rounds
30–52	34–45	$L_{40}[0]$	$K_2[31\dots26]$	12 rounds
42–64	49–60	$L_{55}[0]$	$K_0[31\dots26]$	12 rounds
20–42	26–38	$L_{32}[0]$	$K_1[31\dots26]$	13 rounds
38–60	44–56	$L_{50}[0]$	$K_2[31\dots26]$	13 rounds
2–24	8–20	$L_{14}[0]$	$K_0[31\dots26]$	13 rounds
12–34	16–28	$L_{22}[0]$	$K_1[31\dots26]$	13 rounds

7 Conclusions and Open Problems

This paper presented several meet-in-the-middle attacks on 7-, 15- and 23-round XTEA. The main highlight of our attacks is that they require very few known

[3] In the attack, one bit in the middle is independent of 11 key bits. Two bits in the middle are simultaneously independent of fewer than 11 key bits, thereby corresponding to a larger k_1.

plaintexts (not more than 18) as opposed to previously reported attacks (the best of these attacks requires 2^{20} chosen plaintexts). Furthermore, our attacks use different approaches - the 7- and 23-round attacks use a straightforward meet-in-the-middle approach; in the 15-round attacks, the meet-in-the-middle corresponds to inner round subkeys rather than intermediary text values.

Each of our attacks on 23-round XTEA requires less time ($2^{117.00}$ 23-round computations) than the previously best-known attack on 23 rounds ($2^{120.65}$ 23-round computations) in the standard setting. The time complexities of the 7- and 15-round attacks are also significantly better than exhaustive key search, with each of these attacks requiring about 2^{95} time.

Our attacks apply to XETA as well, a close variant of XTEA that is also implemented in the Linux kernel. We are unaware of any other published cryptanalysis results on XETA.

An interesting observation from one of the anonymous reviewers, is that there is also a 15-round attack on rounds 2–16. In this case, subkey K_0 is used consecutively in the inner rounds 8, 9 and 10, but not elsewhere. By exhaustively searching over K_1, K_2, K_3 and six of the least significant bits of K_0, we can perform the same meet-in-the-middle attack that is described in Sect. 6. However, this attack has a higher time and data complexity than the other 15-round attacks of Sect. 5, for a comparable success probability.

When constructing the 23-round attack in Sect. 6, we found that for any number of *inner rounds* (where all subkeys can be used) up to 16, there is no corresponding attack on more than 23 rounds. However, if the number of inner rounds can be increased to 17, this leads to a 29-round attack. All such 29-round attacks are listed in Table 7. We present the cryptanalysis of these 29-round XTEA block ciphers as an interesting open problem.

Table 7. All reduced-round XTEA block ciphers for which a 29-round attack consists of 17 *inner rounds*

Total rounds	Inner rounds	Subkey containing unused key bits
11–39	27–33	K_0
15–43	21–37	K_2
29–57	35–51	K_1
33–61	40–56	K_3

Acknowledgments. The authors would like to thank Tor E. Bjørstad, Gaëtan Leurent, Matt Robshaw and Aleksander Wittlin for their useful comments and suggestions. Part of this work was performed at the Cryptanalysis of Light-Weight Ciphers Research Meeting, hosted by Katholieke Universiteit Leuven as an initiative of SymLab-WG2: Lightweight Cryptography of the ECRYPT II project. The authors would like to thank the anonymous reviewers for their constructive comments as well.

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. Bouillaguet, C., Dunkelman, O., Leurent, G., Fouque, P.-A.: Another Look at Complementation Properties. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 347–364. Springer, Heidelberg (2010)
4. 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)
5. Diffie, W., Hellman, M.E.: Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10(6), 74–84 (1977)
6. Dunkelman, O., Sekar, G., Preneel, B.: Improved Meet-in-the-Middle Attacks on Reduced-Round DES. In: Srinathan, K., Pandu Rangan, C., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 86–100. Springer, Heidelberg (2007)
7. Grothe, A.: Kernel v2.6.14 tea.c. Linux Headquarters (2004), http://www.linuxhq.com/kernel/v2.6/14/crypto/tea.c
8. Hong, S., Hong, D., Ko, Y., Chang, D., Lee, W., Lee, S.: Differential Cryptanalysis of TEA and XTEA. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 402–417. Springer, Heidelberg (2004)
9. Hong, D., Koo, B., Sasaki, Y.: Improved Preimage Attack for 67-Step HAS-160. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 332–348. Springer, Heidelberg (2010)
10. Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: A Practical Attack on KeeLoq. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 1–18. Springer, Heidelberg (2008)
11. Kaps, J.-P.: Chai-Tea, Cryptographic Hardware Implementations of xTEA. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 363–375. Springer, Heidelberg (2008)
12. Kelsey, J., Schneier, B., Wagner, D.: Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 237–251. Springer, Heidelberg (1996)
13. Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In: Han, Y., Okamoto, T., Qing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 233–246. Springer, Heidelberg (1997)
14. Ko, Y., Hong, S., Lee, W., Lee, S., Kang, J.-S.: Related-Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 299–316. Springer, Heidelberg (2004)
15. Lee, E., Hong, D., Chang, D., Hong, S., Lim, J.: A Weak Key Class of XTEA for a Related-Key Rectangle Attack. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 286–297. Springer, Heidelberg (2006)
16. Lu, J.: Related-key rectangle attack on 36 rounds of the XTEA block cipher. International Journal of Information Security 8(1), 1–11 (2009), http://jiqiang.googlepages.com/IJIS8.pdf
17. Moon, D., Hwang, K., Lee, W., Lee, S., Lim, J.: Impossible Differential Cryptanalysis of Reduced Round XTEA and TEA. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 49–60. Springer, Heidelberg (2002)

18. Needham, R.M., Wheeler, D.J.: Tea extensions, technical report, Computer Laboratory, University of Cambridge (October 1997), http://www.cix.co.uk/~klockstone/xtea.pdf
19. Needham, R.M., Wheeler, D.J.: Correction to xtea. Technical report, Computer Laboratory, University of Cambridge (October 1998), http://www.movable-type.co.uk/scripts/xxtea.pdf
20. Saarinen, M.-J.: Cryptanalysis of Block TEA, unpublished manuscript (October 1998), http://groups.google.com/group/sci.crypt.research/msg/f52a533d1e2fa15e
21. 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)
22. Shannon, C.E.: Communication Theory of Secrecy Systems. Bell System Technical Journal 28(4), 656–715 (1949)
23. Steil, M.: 17 Mistakes Microsoft Made in the Xbox Security System. In: Chaos Communication Congress 2005 (2005), http://events.ccc.de/congress/2005/fahrplan/events/559.en.html
24. Wheeler, D.J., Needham, R.M.: TEA, a Tiny Encryption Algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995)

A Countermeasures

The attacks in this paper are made possible because a particular subkey K_i is often not used for a large number of rounds. To prevent against the attacks in this paper, we propose to use each of the subkeys K_0, K_1, K_2, K_3 once every four rounds, in a random order. This countermeasure does not prevent trivial meet-in-the-middle attacks on 6 rounds. Note that the subkeys cannot repeat in a cyclic manner, as we want to avoid the possibility of slide attacks.

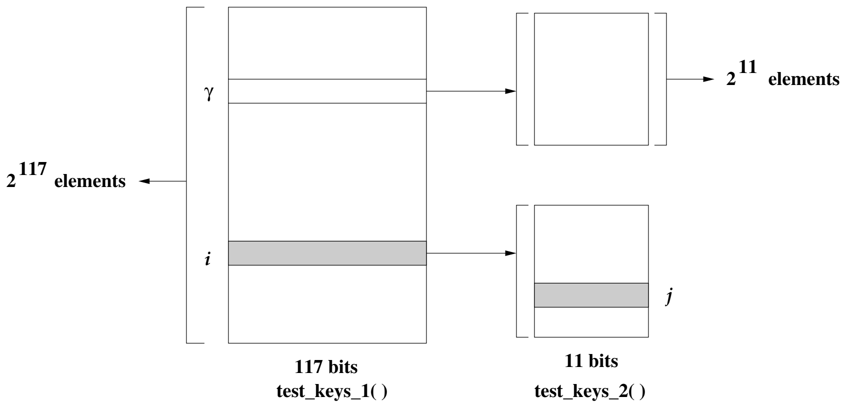


Fig. 3. Attack on rounds 16–38 using Algorithm 1: the tables (not stored in memory) denote the two stages of Algorithm 1 and the shaded 128 bits denote the correct 128-bit key; for a wrong key γ, `test_keys_2()` is performed 2^{11} times

B Illustration of the Attack on Rounds 16–38

In Fig. 4, we illustrate the 23-round attack of Sect. 6. The attack is on rounds 16–38, and uses 11 inner rounds (22–32). Grey boxes represent bits that do not depend on the value of $K_3[31\dots21]$. In Fig. 3, we illustrate Algorithm 1 from the point of view of computation of its time complexity.

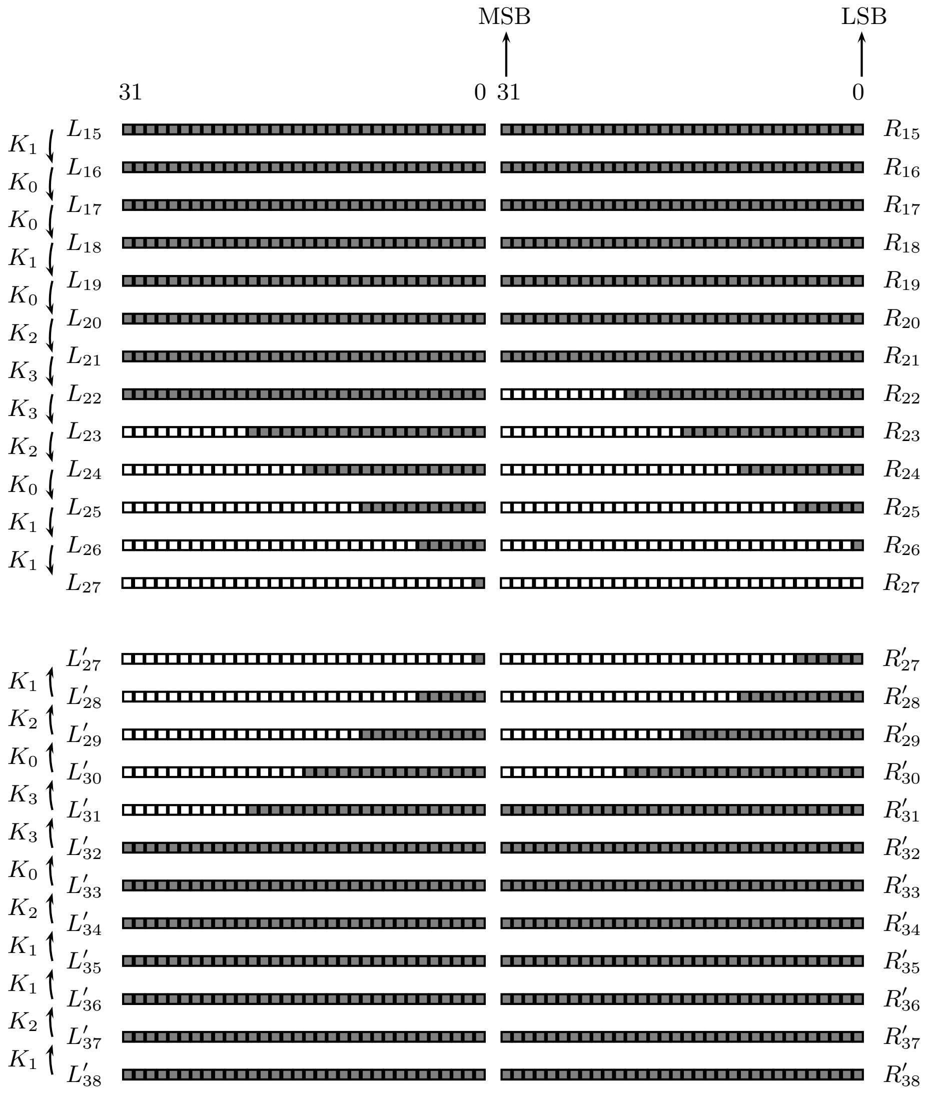

MSB
LSB
31
0
31
0
K_1 K_0 K_0 K_1 K_0 K_2 K_3 K_3 K_2 K_0 K_1 K_1
L_{15} … L_{27}
R_{15} … R_{27}
K_1 K_2 K_0 K_3 K_3 K_0 K_2 K_1 K_1 K_2 K_1
L'_{27} … L'_{38}
R'_{27} … R'_{38}

Fig. 4. 23-round attack (rounds 16–38), using 11 *inner rounds* (the grey boxes represent bits that do not depend on the value of $K_3[31\dots21]$)

C Randomness of the Inner-Round Subkeys in the 15-Round Attacks

Here, we show that if the texts obtained by encrypting p_0 and decrypting c_0 in the 13 outer rounds (of a 15-round attack) are uniformly distributed at random, then so are the subkeys in the inner rounds. As there are only two inner rounds, the problem may be viewed as follows. In Fig. 1, if $L_{t-1}||R_{t-1}$ and $L_{t+1}||R_{t+1}$ are uniformly distributed at random, then we need to show that α_t and α_{t+1} are also uniformly distributed at random. Henceforth, the term *random* means *uniformly distributed at random*.

Since F is a bijection, the output of F is random given R_{t-1} is random. We know that modular addition (or subtraction) or exclusive-OR of two random values results in a random value. Given this, since $R_t = L_{t+1}$ and $L_{t+1}||L_{t-1}$ is random, from Fig. 1 we obtain that $\delta_t \boxplus \alpha_t$ is random. As δ_t is a constant, α_t is random. By similar arguments, it is easily seen that α_{t+1} is also random.

Expedient Non-malleability Notions for Hash Functions

Paul Baecher, Marc Fischlin, and Dominique Schröder

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. Non-malleability of a cryptographic primitive is a fundamental security property which ensures some sort of independence of cryptographic values. The notion has been extensively studied for commitments, encryption and zero-knowledge proofs, but it was not until recently that the notion—and its peculiarities—have been considered for hash functions by Boldyreva et al. (Asiacrypt 2009). They give a simulation-based definition, basically saying that for any adversary mauling hash values into related ones there is a simulator which is as successful in producing such hash values, even when not seeing the original hash values. Their notion, although following previous approaches to non-malleability, is nonetheless quite unwieldy; it is hard to achieve and, due to the existential quantification over the simulator, hard to falsify. We also note that finding an equivalent indistinguishability-based notion is still open.

Here we take a different, more handy approach to non-malleability of hash functions. Our definition avoids simulators completely and rather asks the adversary to maul the hash value and to also specify a transformation ϕ of the pre-image, taken from a fixed class Φ of admissible transformations. These transformations are usually determined by group operations and include such cases such as exclusive-ors (i.e., bit flips) and modular additions. We then simply demand that the probability of succeeding is negligible, as long as the original pre-image carries enough entropy. We continue to show that our notion is useful by proving that, for example, the strengthened Merkle-Damgård transformation meets our notion for the case of bit flips, assuming an ideal compression function. We also improve over the security result by Boldyreva et al., showing that our notion of non-malleability suffices for the security of the Bellare-Rogaway encryption scheme.

Keywords: Hash function, Non-malleability.

1 Introduction

Non-malleability, first treated formally in [18] for commitments, encryption and zero-knowledge, provides some level of independence between cryptographic values. That is, a commitment or ciphertext should not help to generate another commitment or ciphertext of a related message. By this, the adversary should not be able to produce a meaningful value by flipping some bits in a commitment or ciphertext. Several subsequent works considered this notion for the aforementioned primitives [18,7,35,17,21,16,2,15,32,33,13,31,12,14,23,24,25,29,30].

Non-Malleability of Hash Functions. Non-malleability of hash functions was scrutinized only recently by Boldyreva et al. [9]. They provide a formal treatment of the primitive and discuss applications to the Bellare-Rogaway encryption scheme and client puzzles. Other applications of non-malleable hash functions include security proofs for HMAC [20] and for OAEP [10]. The idea behind non-malleable hash functions, put forward in [9], is similar to other areas but also reveals some differences, originating from the fact that, unlike commitments and encryptions, hash functions do not have secret randomness.

The definition in [9] follows the simulation paradigm: For every adversary which is able to transform hash values into related ones, there should be a simulator which outputs such related values but which is denied the original values. This ensures independency of cryptographic values because given the value does not facilitate the task. However, the definition in [9] comes with several deficiencies:

– The requirement is quite strong: the hash function must not for instance leak individual bits of the input, or else the adversary would gain a significant advantage over any simulator by learning the hash value. While this seems to be a desirable goal for commitments and encryption, and possibly for some hash function applications, in general leaking some bits may not do harm to the fact that one mauls a hash value into something meaningful. Hence, it would be preferable to separate the notions of non-malleability and of pre-image hiding.
– The existential quantification over the simulator makes it hard to falsify (cf. Naor's work on cryptographic assumptions [28]) the property for a specific hash function: one would need to show that for some adversary no appropriate simulator whatsoever exists. This is contrast to other desirable hash function properties like collision resistance.
– The definition in [9] covers pathological hash functions like constant functions: since the hash value does not lend any additional power to the adversary—after all, it is a constant— such a function is trivially non-malleable. Intuitively, though, for such functions it is easy to find related hash values and to determine hash values of the pre-image with some bits flipped. One can rule out such pathological functions by demanding collision-resistance or unpredictability, but this would introduce another assumption (which is somewhat unrelated, as we discuss).
– As discussed in [9], finding an alternative indistinguishability-based approach as for commitments and encryption seems to be hard. Very often, though, such a notion is easier to work with for proofs where the hash functions are used within larger schemes.

Non-Malleability goes Handy. We overcome the above problems by reverting to the core idea behind non-malleability: it should be hard to modify a given hash value such that the pre-image is affected in a controlled way. In contrast, the simulation-based approach somewhat guarantees more than this, by ensuring that the hash values are absolutely useless for doing so; this is formalized by having a simulator approximating the adversary's strategy but without learning the hash value.

Typically, the simulator in non-malleability proofs for hash functions runs a copy of the adversary, creates a fake hash value and presents it to the adversary in order to produce a related output. This, however, inhibits some designs which may otherwise guarantee the required "mauling resistance". As an example, if the hash function leaks a single bit of the input, then this clearly violates the simulation technique above. Nonetheless, determining a related hash value may still be hard for the adversary, because of the large unknown input portion. In fact, since practically designed hash functions should ensure that small changes in the input affect all output bits ("avalanche effect") it is likely that most hash functions would still withstand such mauling attacks: the adversary would still need to "control" the effect on the other bits. We note that the avalanche effect, albeit seemingly necessary, is not known to provide our desired level of non-malleability.[1]

We formalize the above non-malleability approach following related-key attacks on pseudorandom functions [22,8,5,26]. There, the adversary can query the pseudorandom function with secret key k on values, but also specify a transformation ϕ of the key to receive values for related key $\phi(k)$. The class Φ of admissible functions ϕ is usually restricted, else achieving security in this setting is impossible [4].

In our case, we hand the adversary a hash value y of an unknown pre-image x and ask her to specify a transformation $\phi \in \Phi$ from a class of admissible transformations, together with a hash value y^*. The adversary wins if $x^* = \phi(x)$ hashes to y^*; to avoid trivial copy-attacks, we also demand that $x^* \neq x$. Non-malleability now requires that the probability of the adversary winning is negligible, implying that the (adversarially chosen) distribution of the unknown pre-image x contains super-logarithmic min-entropy. We denote this notion by Φ-non-malleability.

Similar to related-key attacks on pseudorandom functions, we cannot hope to achieve our notion of Φ-non-malleability for arbitrary classes Φ of transformations. Still, it comprises a large set of interesting transformation. For example, it contains the important class of all "bit flips", $\phi_\delta(x) = x \oplus \delta$ for all δ, for which we denote the notion by $\oplus$-non-malleability. More generally, one may consider any operations over groups, like $\phi_\delta(x) = x + \delta \bmod 2^n$ to capture modular additions. For a group $(G, \odot)$ and such group-induced transformations $\phi_\delta^{\odot}(x) = x \odot \delta$ we speak of $\odot$-non-malleability.

Applying our Non-Malleability Notion. We show that our notion $\odot$-non-malleability is strictly weaker than the simulation-based approach of Boldyreva

[1] We also note that the idea of reverting to basic modification attacks does not seem to be applicable to non-malleable commitments or encryptions in a reasonable way; the latter primitives are designated to hide the messages, whereas hash functions are a-priori not meant to provide such a guarantee.

et al. [9]. This, per se, would not be an interesting result, unless we can prove that the notion is still useful or enriches the class of suitable hash functions. We show both.

Clearly, since simulation-based non-malleability implies $\odot$-non-malleability, we can immediately conclude from the construction in [9] that $\odot$-non-malleable (and thus $\oplus$-non-malleable) hash functions can be derived under standard assumptions in principle. However, one of the ideas behind our notion is to give more guidance on how to design practical hash functions, and we rather envision more practical constructions from $\oplus$-non-malleable hash functions.

As for positive results, we show that the strengthened Merkle-Damgård, achieves the notion of $\oplus$-non-malleability if we assume an ideal compression function, emphasizing that restricting the class of admissible transformations allows to derive non-trivial non-malleability statements. Some of our results are also negative, though. We show that, due to length extension attacks, (strengthened) Merkle-Damgård is not Φ-non-malleable for a large class of transformations, even if we model the compression function as a random oracle. We also indicate that the Matyas-Meyer-Oseas (MMO) construction where one adds the input message to the output of a cipher shows some weaknesses when it comes to $\oplus$-non-malleability. More precisely, our result says that, if the compression function can be an arbitrary $\oplus$-non-malleable function, then MMO may not preserve this property. However, note that MMO itself assumes that the compression function is a block cipher, hence our result does not immediately apply to hash functions built from MMO like Skein [19].

We finally show that $\oplus$-non-malleability suffices to show the Bellare-Rogaway (BR) encryption scheme [6] to be chosen-ciphertext secure, where a message is encrypted via $(f(r), G(r) \oplus m, H(r||m))$ for trapdoor permutation f, random r and hash functions G and H. Boldyreva et al. [9] prove that (simulation-based) non-malleability of H suffices, as long as G is still treated as a random oracle. To be precise, they also require H to be collision-resistant and perfectly one-way [11], hiding all information about the pre-image. Here we improve over their result and show the BR scheme is chosen-ciphertext secure (for random oracle G), as long as H is $\oplus$-non-malleable and perfectly one-way. This is an example where hiding the entire pre-image is now welcome and made explicit.

Other Related Work. "Bit-flipping" attacks have been known for a long time. It is the achievement of works like [18] to put them into a general and formal framework and show how to avoid them and how to apply this security property. As explained above, our definition of Φ-non-malleability follows this line. It is inspired by the notion of related-key attacks [8,22,4] and the fact that the definition of Boldyreva et al. [9] is quite bulky.

It should be mentioned that a similar notion to $\oplus$-non-malleability appeared previously in Shoup's attack on the OAEP proof [36]. Shoup gives an ad-hoc definition for XOR-malleable trapdoor permutations, i.e., trapdoor permutations which succumb to such bit-flip attacks, and he shows that OAEP is insecure when instantiated with such a permutation. His notion, besides demanding the opposite of non-malleability, is slightly different (and incomparable) to ours in the sense that the adversary runs on a given difference δ and is not allowed to choose the distribution of inputs to the permutation. Our work here provides a positive, more expedient notion of *non*-malleable hash functions.

2 Preliminary Definitions

Notations. If x is a bit string then $|x|$ denotes its length and x_i the ith bit, whereas the least significant bit is at position $i = 0$; a bit position i is modulo $|x|$. By $x||y$ we denote the concatenation of two strings x and y and assume that it has the lowest evaluation priority. We assume (unless indicated otherwise) that all algorithms run in "probabilistic polynomial-time". If A is a set, we write $a \leftarrow A$ to indicate that a is picked uniformly at random from this set. The same syntax is used if A is a distribution (a is then drawn accordingly) and if $A(\cdot)$ is an algorithm, then a denotes its output. When dealing with a concatenated bit string $x_0||x_1$ where $|x_0| = |x_1|$, we often write left-hand (or right-hand) side to refer to x_0 (or x_1) and function $\mathsf{split}(b, x_0||x_1)$ returns x_b. Functions $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ denote any function that is polynomial in λ and negligible in λ, respectively. The security parameter is denoted by 1^λ. All logarithms are base 2.

The following definition captures a very general notion of hash functions, allowing also probabilistic schemes:

Definition 1 (Hash function). *A hash function $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ consists of algorithms, where $\mathsf{HK}(1^\lambda)$ generates and outputs a key K which implicitly defines a domain $D(K)$, H for inputs K and $x \in D(K)$ evaluates to a value $y \in \{0,1\}^*$, and HVf returns a verification decision bit for inputs K, x, y. It is required for any $K \leftarrow \mathsf{HK}(1^\lambda)$, any $x \in D(K)$, and any $y \leftarrow \mathsf{H}(K, x)$ that $\mathsf{HVf}(K, x, y)$ outputs 1.*

We often require probability distributions to be non-trivial with regard to predictability. This means that it should be hard to predict which element is drawn from a given distribution when sampled, even in the presence of additional information in terms of a hint about the sample. We formalize this requirement as conditional min-entropy and resort to an equivalent formulation via predictors by [1].

Definition 2 (Conditional min-entropy). *The conditional min-entropy of a distribution $\mathcal{X}$ conditioned on distribution $\mathcal{Z}$ is*

$$\tilde{\mathbf{H}}_\infty(\mathcal{X}|\mathcal{Z}) := -\log \max_{\mathcal{A}} \Pr\left[\mathcal{A}(\mathcal{Z}) = \mathcal{X}\right]$$

where the probability is taken over the random coins of $\mathcal{A}$, $\mathcal{Z}$ and $\mathcal{X}$.

In our case, the distribution $\mathcal{Z}$ often depends on $\mathcal{X}$, e.g., a function of $\mathcal{X}$.

3 Defining Non-malleability

We first present the original definition from Boldyreva et al. [9], before introducing our notion of Φ-non-malleability and discussing their relationship.

3.1 Simulation-Based Non-malleability

The idea of simulation-based non-malleability originates from [18], where it has been defined, among others, for private key encryption. It states, intuitively, that for given ciphertext $C(x)$ of a message x, it should be computationally hard to find $C(x^*)$ where x and x^* are related in some interesting way. In their recent work Boldyreva et al. [9] provide a simulation-based definition of non-malleable hash functions. This definition constitutes the natural translation of simulation-based non-malleable encryption: For given $\mathsf{H}(x)$, it should be computationally hard to find $\mathsf{H}(x^*)$ where x and x^* are meaningfully related, defined via a relation R form a class of relations $\mathcal{R}$.

As with any simulation-based definition, the hardness of finding x and x^* is expressed over the probability that an efficient attacker is not significantly more successful than a simulator in an idealized version of the same task. That means, it is required that for any efficient attacker there exists a corresponding simulator that does just as well. In this specific case, both algorithms have to output x^* eventually, but the attacker is given $\mathsf{H}(x)$ while the simulator does not have access to $\mathsf{H}(x)$. The following definition is almost verbatim from [9] with a minor change (discussed afterwards):

Definition 3 (NM-Hash). *A hash function* $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ *is called non-malleable (with respect to probabilistic algorithm* hint *and relation class* $\mathcal{R}$*) if for any PPT algorithm* $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_y, \mathcal{A}_x)$ *there exists a PPT algorithm* $\mathcal{S} = (\mathcal{S}_d, \mathcal{S}_x)$ *such that for every relation* $R \in \mathcal{R}$ *the difference*

$$\Pr\left[\mathbf{Exp}^{nmh\text{-}1}_{\mathcal{H},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{nmh\text{-}0}_{\mathcal{H},\mathcal{S}}(\lambda) = 1\right]$$

is negligible, where:

```
Experiment $\mathbf{Exp}^{nmh\text{-}1}_{\mathcal{H},\mathcal{A}}(\lambda)$
    $K \leftarrow \mathsf{HK}(1^\lambda)$
    $(\mathcal{X}, st_d) \leftarrow \mathcal{A}_d(K)$
    $x \leftarrow \mathcal{X}(1^\lambda),\ h_x \leftarrow \mathsf{hint}(K, x)$
    $y \leftarrow \mathsf{H}(K, x)$
    $(y^*, st_y) \leftarrow \mathcal{A}_y(y, h_x, st_d)$
    $(x^*, r) \leftarrow \mathcal{A}_x(x, st_y)$
    Return 1 iff
        $R(\mathcal{X}, x, x^*, r)$
        $\wedge (x, y) \neq (x^*, y^*)$
        $\wedge \mathsf{HVf}(K, x^*, y^*) = 1$
```

```
Experiment $\mathbf{Exp}^{nmh\text{-}0}_{\mathcal{H},\mathcal{S}}(\lambda)$
    $K \leftarrow \mathsf{HK}(1^\lambda)$
    $(\mathcal{X}, st_d) \leftarrow \mathcal{S}_d(K)$
    $x \leftarrow \mathcal{X}(1^\lambda),\ h_x \leftarrow \mathsf{hint}(K, x)$
    $(x^*, r) \leftarrow \mathcal{S}_x(h_x, st_d)$
    Return 1 iff
        $R(\mathcal{X}, x, x^*, r)$
```

where $\mathcal{X}$ *denotes a distribution that is chosen by the attacker* $\mathcal{A}_d$ *or simulator* $\mathcal{S}_d$ *and st stands for state information passed among the algorithms.* $\mathcal{H}$ *is called entropically non-malleable if the above only holds for algorithms* $\mathcal{A}_d$ *and* $\mathcal{S}_d$ *that output distributions* $\mathcal{X}$ *such that* $\tilde{\mathbf{H}}_\infty(\mathcal{X}|\mathsf{HK}, \mathsf{hint}) \in \omega(\log \lambda)$.

A wealth of remarks is presented in the original work; we only comment on the most important details:

- Both experiments provide the algorithms with a hint h_x, which reflects information about x that might have been collected from previous actions, such as other protocol executions that involve x.
- The informal statement that x and x^* are related in a meaningful way is represented by quantification over a class of relations $\mathcal{R}$. Ideally, we would want to allow any relation, but subtle technicalities described in [9] demand that the set of all possible relations is restricted; the definition becomes unachievable otherwise. This restriction can be expressed by excluding the problematic relations from $\mathcal{R}$.
- We introduced a small change concerning the output of $\mathcal{A}_x$ and $\mathcal{S}_x$ as well as the relation. In particular, the original definition omits the value r which gives the attacker a little more control over the relation. By specifying a parameter r along with x^*, the attacker is effectively able to select a specific subset of the relation. This can be viewed as a decomposition of a corresponding relation that omits r into subsets where the resulting subsets are indexed by r. Applying this modification will turn out useful later for our analysis, and does not affect the results in [9].
- We also define entropically non-malleability as a variant of non-malleability that requires attackers not to output trivial preimage distributions. This is useful because the attacker could otherwise attempt to guess the preimage that the experiment samples from the distribution instead of attacking non-malleability. The original definition does not formalize this explicitly but mentions it as a naturally arising condition.

3.2 Game-Based Non-malleability

We avoid the drawbacks of the simulation-based approach mentioned in the introduction, by proposing an alternative definition that does not rely on a simulator. Our goal is to provide a definition that is more compact and accessible to both cryptanalysts and practitioners while it still captures the idea of non-malleability.

The Idea. One candidate realization of our goal is captured by the following game. After selecting a preimage distribution of inputs to some hash function h, the attacker is given image $\mathsf{H}(x)$ and is required to output $\mathsf{H}(x^*)$ along with a description of how to transform x into x^* in terms of a function ϕ. Note that this does not imply that the attacker knows x or x^*. Consider the following experiment which outlines this idea.

1. Attacker $\mathcal{A}$ outputs a preimage distribution $\mathcal{X}$.
2. A random preimage x is picked according to $\mathcal{X}$. Let $y = \mathsf{H}(x)$.
3. On input y, the attacker outputs y^* along with a function ϕ which maps one preimage to another, e.g., $\phi = \phi_\delta$ may be the function which maps x to $\phi_\delta(x) = x \oplus \delta$.
4. The output of the experiment is defined to be 1 if, and only if, $\mathsf{H}(\phi(x)) = y^*$ (and we have $\phi(x) \neq x$).

As a natural consequence for a non-malleability definition, we would require the probability that this experiment outputs 1 to be negligible in the security parameter. Modeling the description of how to transform the preimage with a function also allows the attacker to output modifications like $\phi(x) := x + 1$. Unfortunately, this broad definition is unachievable. Consider, for example, an attacker that simply picks x^* at random from the preimage domain and outputs the constant function $\phi(\cdot) := x^*$ and $y := \mathsf{H}(x^*)$. It succeeds unconditionally in this experiment since $\mathsf{H}(\phi(x)) = \mathsf{H}(x^*) = y^*$, regardless of the hash function in question. This problem occurs whenever the adversary can provide a function which significantly reduces the size of the resulting range of the transformation function. We thus restrict the class Φ of functions ϕ.

Φ-*Non-Malleability.* Seeing that the above experiment is unachievable, the question remains if a weaker—but working—game-based definition exists that still reflects the intuition of non-malleability. We approach this question with the following idea: If two related values x and x^* exist, then there is a difference $\delta := x \oplus x^*$ that essentially describes how to modify one value in order to obtain the other. Yet, given the difference only, no information in absolute terms about x (or x^*, separately) can be inferred. We can view an attacker that outputs δ as a specialization of the experiment above by fixing $\phi(x) = x \oplus \delta$. The practical impact of this idea would be that an attacker has an understanding of how bit flips in the output affect bit flips in the input of the function (or vice versa).

It is possible to generalize this concept by allowing any group operation instead of bitwise differences. We note that a similar issue of specifying such a transformation function has been already used in the formal treatment of related-key attacks on pseudorandom functions, where the adversary is allowed to provide a key transformation. Here, Lucks [26] proposes the class of group-induced transformations, a class which is neither too powerful nor too restrictive: It is the set of functions which apply the group operation to their argument and a fixed group element. Subsequent work in the area by Bellare and Cash [3] advocates the use of this class and strengthens our confidence that this is a "natural" choice for related key attacks. Since the requirements and issues seem to be very similar, we adopt this class in our definition of non-malleability.

We now turn this idea into a general formal definition that is similar to the simulation-based experiments. First, HK generates a key K which implicitly contains the security parameter λ. On input K, the first stage of the attacker $\mathcal{A}_d$ outputs a valid distribution of preimages $\mathcal{X}$ in the sense that it has a sufficient min-entropy. A random element x is drawn from this distribution and mapped to image y. The function hint outputs hint h_x about x. The attacker in the second stage then receives image y and hint h_x. It is required to output a modified image y^* and a preimage transformation function $\phi \in \Phi$ (where Φ is known to the adversary). The output of the experiment is defined to be 1 if, and only if, $\mathsf{HVf}(K, \phi(x), y^*) = 1$ and $\phi(x) \neq x$.

Definition 4 (Φ-NM-Hash). *A hash function $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ is called Φ-non-malleable (with respect to probabilistic function* hint*) if for any PPT algorithm $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_y)$ the probability $\Pr[\mathbf{Exp}^{\Phi\mathsf{nm}}_{\mathcal{H},\mathcal{A}}(\lambda) = 1]$ is negligible in λ, where:*

```
Experiment $\mathbf{Exp}^{\Phi\mathsf{nm}}_{\mathcal{H},\mathcal{A}}(\lambda)$
    $K \leftarrow \mathsf{HK}(1^\lambda)$
    $(\mathcal{X}, \mathrm{st}) \leftarrow \mathcal{A}_d(K)$
    $x \leftarrow \mathcal{X}(1^\lambda)$, $h_x \leftarrow \mathsf{hint}(K, x)$
    $y \leftarrow \mathsf{H}(K, x)$
    $(y^*, \phi) \leftarrow \mathcal{A}_y(y, h_x, \mathrm{st})$
    Return 1 iff
        $\mathsf{HVf}(K, \phi(x), y^*) = 1$
        $\wedge\, \phi(x) \neq x$
```

where $\phi \in \Phi$. It is required that algorithm $\mathcal{A}_d$ only outputs efficiently sampleable distributions $\mathcal{X}$ such that $\tilde{\mathbf{H}}_\infty(\mathcal{X}|\mathsf{HK}, \mathsf{hint}) \in \omega(\log \lambda)$.

We require $\mathcal{A}_d$ to output a non-trivial distribution $\mathcal{X}$ in this experiment that is not predictable by demanding a conditional min-entropy strictly greater than logarithmic in λ. This requirement makes sense because, otherwise, an attacker can choose a distribution such that it succeeds to predict the most likely event of this distribution with high probability. In the context of our experiment, this would imply that the attacker guesses x with non-negligible probability and trivially succeeds.

Group-Induced Non-Malleability. Note that we let the adversary choose from a set of predetermined transformation functions. From this general version we get the aforementioned group-induced transformations by letting $\Phi = \{\phi^{\odot}_{\delta} : \delta \in G\}$ where $\phi^{\odot}_{\delta}(x) = x \odot \delta$ for some group $(G, \odot)$ for which group operations can be efficiently performed. Although this definition allows us to model even richer classes of transformation functions (e.g. many different group operations in the same experiment), one must take care to exclude cases where the definition becomes unachievable. In practice, one will likely want to work with one specific group. In fact, for this paper, we restrict ourselves to group-induced transformations. The following definition captures two special cases of group-induced transformations and the $\oplus$ operation:

Definition 5 ($(G, \odot)$-NM-Hash and $\oplus$-NM-Hash). *Let $(G, \odot)$ denote a group. A hash function $\mathcal{H}$ is called $(G, \odot)$-non-malleable if $\mathcal{H}$ is $\Phi^{\odot}$-non-malleable where $\Phi^{\odot} = \{\phi^{\odot}_{\delta} : \delta \in G\}$ and $\phi^{\odot}_{\delta}(x) = x \odot \delta$. We omit G if it is clear from the context. In particular, we call a $(\{0,1\}^*, \oplus)$-non-malleable function simply $\oplus$-non-malleable.*

It should be observed that Definition 5 is weaker than the original simulation-based definition. We discuss this in depth in the next section. Nevertheless, despite this limitation, we feel that Definition 5 is suitable to capture an essential aspect of non-malleability and a broad class of transformation functions. The absence of a simulator makes it relatively easy to work with and relates well to a practical view of an attacker.

3.3 Relations between Simulation-Based and $\odot$-Non-malleability

In this section, we show that every simulation-based non-malleable hash function is also $\odot$-non-malleable but $\odot$-non-malleable hash functions exist that are not simulation-based non-malleable.

Simulation-Based Non-malleability $\Rightarrow$ $\odot$-Non-malleability. We show by black-box reduction that every function which is not $\odot$-non-malleable is also not entropically non-malleable in the simulation-based sense.

Proposition 1. *Let $R_\odot(\mathcal{X}, x, x^*, r)$ denote any relation that outputs 1 if, and only if, $r = x \odot x^*$. If $\mathcal{H}$ is an entropically non-malleable hash function with respect to arbitrary* hint *and relation class $\mathcal{R} \ni R_\odot$, then $\mathcal{H}$ is $\odot$-non-malleable with respect to* hint.

Our proof in the full version of the paper shows the equivalent proposition that if $\mathcal{H}$ is not $\odot$-non-malleable then $\mathcal{H}$ is not entropically non-malleable.

We remark that the implication actually holds in a more general sense, beyond group-induced transformations. Namely, assume that for every $\phi \in \Phi$ there exists $\phi^{-1} \in \Phi$ such that $\phi^{-1}(\phi(x)) = x$ for all x. Now define the relation $R_{-1}(\mathcal{X}, x, x^*, r)$ (instead of $R_\odot$) which outputs 1 if and only if $r(x^*) = x$ for $r \in \Phi$. Then any successful adversary against Φ-non-malleability could still be turned into a successful adversary against entropical non-malleability, whereas a simulator could output (x^*, r) predicting $r(x^*) = x$ with negligible probability only. As an example consider the class $\Phi^{||}$ of concatenation transformation consisting of function $\phi_\delta(x) = x||\delta$ and such that $\phi_\delta^{-1}(x||\delta) = x$ (or equals the input, in case the input string does not end on δ). This example will be useful when showing that Merkle-Damgård does not preserve Φ-non-malleability.

Φ-Non-Malleability $\not\Rightarrow$ Simulation-Based Non-Malleability. We show that if Φ-non-malleable functions exist, then there is one which is not (simulation-based) non-malleable. In particular, Construction 1 below is one example for such a function.

Construction 1. *Let $\mathcal{F} = (\mathsf{FK}, \mathsf{F}, \mathsf{HVf})$ denote a hash function. Define $\mathcal{G} := (\mathsf{GK}, \mathsf{G}, \mathsf{GVf})$ where $\mathsf{GK} := \mathsf{FK}$, $\mathsf{G}(K, x) := \mathsf{F}(K, x)||x_1$, and $\mathsf{GVf}(K, x, y) := \mathsf{HVf}(K, x, y_1 y_2 \dots y_{n-1}) \wedge (y_n = x_1)$.*

Lemma 1. *For any class Φ it holds that, if $\mathcal{F}$ is Φ-non-malleable, then $\mathcal{G}$ is Φ-non-malleable.*

The proof follows straightforwardly by a black-box reduction as an adversary against $\mathcal{G}$ can simply guess the extra bit and still succeed with non-negligible probability. The proof appears in the full version of the paper.

Proposition 2. *Let $R_1(\mathcal{X}, x, x^*, r)$ denote any relation that outputs 1 if, and only if, $x_1 = x_1^*$ and $\mathcal{X}$ is the uniform distribution. Then Construction 1 is not (simulation-based) non-malleable with respect to* $\mathsf{hint} = \emptyset$ *and relation class $\mathcal{R} \ni R_1$.*

For the proof of Proposition 2 we consider the specific relation R_1 consisting of inputs x, x^* which are equal in the first bit. In addition, this relation rejects distributions if they are not the uniform distribution. Since an attacker against Construction 1 is given the first bit as part of the image, it succeeds unconditionally. On the other hand, the simulator does not have this information and is forced to guess, but it succeeds only with a probability that makes the difference non-negligible. To complete the picture we finally note that relation R_1 is in the set of relations for which Boldyreva et al. [9] derive (simulation-based) non-malleable hash functions.

See again the full version for the full proof where we also show that non-malleability is not implied by collision-resistance or one-wayness.

Finally, we point out that it is an open question to identify precisely how much weaker Definition 4 is in contrast to the original definition. In particular, it is unclear if a Φ-non-malleable hash function that also hides all preimage information is also non-malleable in the simulation-based sense.

4 Constructions

As explained in the introduction, the Merkle-Damgård construction does not preserve (Φ-)non-malleability because of length-extension attacks, even if the compression function is modeled as a random oracle. Clearly, this is the case for the class of transformations $\Phi^{||}$ with $\phi_\delta(x) = x||\delta$ first appending the padding and then arbitrary data.[2] It follows that Merkle-Damgård is not simulation-based non-malleable either.

Merkle-Damgård, Random Oracles, and $\oplus$-Non-Malleability. Somewhat surprisingly, we show that Merkle-Damgård does provide $\oplus$-non-malleability (for fixed-length messages) if the compression function is modeled as random oracle. This again shows the advantage of restricting the class of transformation Φ: while Merkle-Damgård is not simulation-based non-malleable and not $\Phi^{||}$-non-malleable according to the discussion above, it is for another class of transformation which suffices to show security of the BR encryption schemes (see Section 5).

Construction 2 (MD). *Let $\mathsf{pad}(M)$ be a padding function which maps messages to multiples of the block length such that the final 64 bits contain the message length. Then*

$$MD^h_{iv}(M) = h^*_{iv}(M_1||\ldots||M_k) = h(\cdots h(h(iv, M_1), M_2)\cdots)$$

where $M_1||\ldots||M_k = \mathsf{pad}(M)$.

In the following we usually denote by y_i the value $h^*_{\mathrm{iv}}(M_1||\ldots||M_i)$, i.e., the i-th intermediate value when iterating h.

[2] Note that an attacker is not even limited to fixed-length preimage distributions, since the length is polynomial and can simply be guessed.

Proposition 3. *For a random oracle h the hash function MD^h in Construction 2 is $\oplus$-non-malleable (with respect to arbitrary* hint*).*

Proof. Consider an adversary $\mathcal{A}$ against $\oplus$-non-malleability. Assume that this adversary for random $M \leftarrow \mathcal{X}$ and the hash value $y = h^*_{\text{iv}}(\mathsf{pad}(M))$ eventually outputs (y^*, δ). Note that, with overwhelming probability, if the adversary has not queried *all* pairs (M^*_i, y^*_i) of $M^*_1 || \dots || M^*_k = \mathsf{pad}(M \oplus \delta)$ and intermediate values y^*_i in the hash computations to h, then the final output y^* does not constitute a valid hash value. This holds because if the adversary simply copies $y^* = y$ this would contradict the collision-resistance for $\delta \neq 0$; in any other case the adversary would otherwise be able to predict random oracle values.

Hence, given that the adversary has queried about all intermediate values, and since they are unique with overwhelming probability, we can deduce the adversary's message M^* and together with the output δ thus the original message M. This, however, would contradict the super-logarithmic min-entropy of $\mathcal{X}$ (because the additional hash value of M can only decrease this entropy by a logarithmic term for the efficient adversary making at most polynomial many queries to h). $\square$

We are not aware if one can show that Merkle-Damgård is $\oplus$-non-malleable under standard assumptions. However, we note that $\oplus$-non-malleability of the compression function alone is not sufficient, but that collision resistance of the compression function is necessary. Consider for example a given $\odot$-non-malleable compression function h and modify it into a compression function

$$h'(w,x) = \begin{cases} h(w,x) & \text{if } x \neq 1\dots1 \\ h(w,0\dots0) & \text{if } x = 1\dots1 \end{cases}$$

Then h' inherits $\odot$-non-malleability from h because the non-trivial min-entropy of the input distribution guarantees that x hits $1\dots1$ and that the adversary outputs ϕ_δ with $\phi_\delta(x) = x \odot \delta = 1\dots1$ only with negligible probability. In any other case breaking non-malleability of h' requires breaking the non-malleability of h.

Now consider the MD construction based on h'. It is easy for the adversary to specify a distribution $\mathcal{X}$ which outputs λ message blocks, and each block consists of $0\dots0$ or $1\dots1$ with probability $1/2$ each. Clearly, this distribution has super-logarithmic min-entropy. But, the adversary can now output the same hash value $y^* = y$ and $\delta = (1\dots1)^\lambda \neq 0$ and win the $\oplus$-non-malleability game (because $x^* = x \oplus \delta$ maps to the same hash value).

The Matyas-Meyer-Oseas-Construction. A common technique to build hash functions—more precisely compressions functions—is to start from a block cipher. Preneel et al. [34] discuss 64 such variants which all rely on one call to a block cipher. One of these variants, proposed earlier by Matyas et al. [27], is the Matyas-Meyer-Oseas (MMO) scheme given by $h(k,m) = C(g(k),m) \oplus m$ where g an arbitrary function. Amongst others, the SHA-3 candidate hash function Skein [19] is known to adopt this scheme.

In this section, we deal with a construction called *hash-$\oplus$-composition* which is quite similar to the MMO scheme and show that it does not sustain $\oplus$-non-malleability. The difference is that, instead of using a cipher C (and function g), we substitute this part with an inner function h', i.e. $h(k, m) = h'(k, m) \oplus m$. This may be a hash function or, more general, any compression function. We then construct a $\oplus$ non-malleable h' and show that the overall construction becomes $\oplus$-malleable. The general idea is presented below; for a full formal treatment see the full version of the paper.

Consider a hash function u, an $\oplus$-non-malleable hash function f and define

$$h'(x_0||x_1) = (u(x_0) \oplus (f(x_0)||f(x_1)))||(x_0 \oplus x_1).$$

Function h' is then $\oplus$-non-malleable under certain assumptions, namely that (a) an attacker against $\oplus$-non-malleability outputs only uniform distributions and (b) function u is modeled as a random oracle. Intuitively, the second assumption assures that the first half of h' is padded by true randomness and the first assumption guarantees that the second half of h' is uniformly random. These two properties combined ensure that h' is $\oplus$-non-malleable. However, it becomes completely insecure in the above scheme. Since x_1 is canceled out, the attacker learns x_0 and is thus able to recompute u and f for arbitrary new values. It is easy to see that this breaks ($\oplus$-)non-malleability.

How does the above result affect hash functions constructed by the MMO paradigm such as Skein? Strictly speaking, it is not applicable, because the MMO schemes uses a cipher whereas the hash-$\oplus$-composition (and the counterexample above) assumes a hash function. Yet, we feel that it is not far-fetched to consider a cipher with unknown key and a hash function functionally similar. For example, collision resistance is implied by the necessary permutation property of a cipher. Likewise, one-wayness is closely related to the security of the cipher: If one is able to invert an "image" of the cipher without knowing its key, then the cipher is blatantly broken.

5 Application

In this section we revisit the proof in [9] that (simulation-based) non-malleability of the hash function H in the Bellare-Rogaway encryption [6] scheme $\mathsf{f}(r)$, $G(r) \oplus m$, $H(r||m)$, together with some form of perfect one-wayness hiding the entire pre-image, suffices to achieve chosen-ciphertext security (as long as G is still modeled as a random oracle). Exploiting the fact that the component $G(r) \oplus m$ uses the exclusive-or we show that $\oplus$-non-malleability (and perfect one-wayness) is sufficient for H.

Preliminaries. We first recall the BR encryption scheme $\mathrm{BR}^{G,\mathcal{H}}[\mathcal{F}] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and the instantiation of the random oracle H through a hash function $\mathcal{H} = (\mathsf{HK}, \mathsf{H}, \mathsf{HVf})$ formally. The key generation algorithm $\mathcal{K}$ of the encryption scheme outputs a random $\mathcal{F}$-instance f and its inverse f^{-1} as the public and secret key, respectively. It also runs HK to generate a key $K \leftarrow \mathsf{HK}(1^k)$ for a hash function and puts K into both the public and secret key. The encryption algorithm $\mathcal{E}$ on inputs f, K and m picks random r in the domain of f and outputs $(\mathsf{f}(r), G(r) \oplus m, \mathsf{H}(K, r||m))$. The decryption algorithm on inputs f^{-1}, K and (y, g, h) first computes $r \leftarrow \mathsf{f}^{-1}(y)$, then $m \leftarrow g \oplus G(r)$, and outputs m iff $\mathsf{HVf}(K, r||m, h) = 1$ (and $\perp$ otherwise).

We would like to show that the encryption is IND-CCA, meaning that for any adversary with access to a decryption oracle and receiving the public key as input, the adversary cannot distinguish encryptions for chosen messages m_0, m_1 (of equal length) better than by guessing. It is understood that the adversary, after receiving the challenge ciphertext of m_b for random bit b cannot query the decryption oracle about this ciphertext.

Next, we define the hiding property of the hash function $\mathcal{H}$ formally. We start by recalling the definition of a perfectly one-way hash function [11]:

Definition 6 (POWHF). *A hash function* $\mathcal{P} = (\mathsf{POWK}, \mathsf{POW}, \mathsf{POWVf})$ *is called a perfectly one-way hash function (with respect to probabilistic function* hint*) if for any PPT algorithm* $\mathcal{B} = (\mathcal{B}_d, \mathcal{B}_b)$*, where* $\mathcal{B}_b$ *has binary output, the following random variables are computationally indistinguishable:*

$K \leftarrow \mathsf{POWK}(1^k)$	$K \leftarrow \mathsf{POWK}(1^k)$
$(\mathcal{X}, st_d) \leftarrow \mathcal{B}_d(K)\ ; x \leftarrow \mathcal{X}(1^k)$	$(\mathcal{X}, st_d) \leftarrow \mathcal{B}_d(K)\ ; x \leftarrow \mathcal{X}(1^k)$
	$x' \leftarrow \mathcal{X}(1^k)$
$h_x \leftarrow \mathsf{hint}(K, x)\ ; y \leftarrow \mathsf{POW}(K, x)$	$h_x \leftarrow \mathsf{hint}(K, x)\ ; y' \leftarrow \mathsf{POW}(K, x')$
$b \leftarrow \mathcal{B}_b(y, h_x, st_d)$	$b \leftarrow \mathcal{B}_b(y', h_x, st_d)$
return (K, x, b)	*return* (K, x, b)

Security Proof. As in [9] we also assume for technical reasons that the hash function $\mathcal{H}$ is $\oplus$-non-malleable when a random instance of $\mathcal{F}$ is included with the key output by HK. Let $\mathcal{H} = (\mathsf{HK}_{\mathcal{F}}, \mathsf{H}, \mathsf{HVf})$ denote the modified hash function for which key generation outputs a random instance of $\mathcal{F}$ together with the original hash key. Then we can include side information about the sample in terms of f, i.e., we too need that $\mathcal{H}$ is $\oplus$-non-malleable with respect to the hint function $\mathsf{hint}_{\mathsf{BR}}((K, \mathsf{f}), r||m) = \mathsf{f}(r)$.

Theorem 3. *Let* $\mathcal{F}$ *be a trapdoor one-way permutation and* $\mathcal{H} = (\mathsf{HK}_{\mathcal{F}}, \mathsf{H}, \mathsf{HVf})$ *be a perfectly one-way hash function with respect to* hint_{BR} *which is also* $\oplus$*-non-malleable with respect to* hint_{BR}*. Then* $BR^{G,\mathcal{H}}[\mathcal{F}] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *for messages of length* $\ell = \omega(\log \lambda)$ *is IND-CCA secure (for random oracle* G*).*

The proof in the full version of the paper proceeds in game hops, starting with the original attack scenario and transforming this into a game where the adversary has no advantage.

Acknowledgments

We thank the anonymous reviewers for valuable comments. The authors are supported by the Emmy Noether Grant Fi 940/2-1 of the German Research Foundation (DFG).

References

1. 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)
2. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: 43rd FOCS, November 16-19, pp. 345–355. IEEE Computer Society Press, Vancouver (2002)
3. 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)
4. 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)
5. 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)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, November 3-5, pp. 62–73. ACM Press, Fairfax (1993)
7. Bellare, M., Sahai, A.: Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 519–536. Springer, Heidelberg (1999)
8. Biham, E.: New types of cryptanalytic attacks using related keys. Journal of Cryptology 7(4), 229–246 (1994)
9. Boldyreva, A., Cash, D., Fischlin, M., Warinschi, B.: Foundations of non-malleable hash and one-way functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 524–541. Springer, Heidelberg (2009)
10. Boldyreva, A., Fischlin, M.: On the security of OAEP. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 210–225. Springer, Heidelberg (2006)
11. Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997)
12. Canetti, R., Dakdouk, R.R.: Extractable perfectly one-way functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 449–460. Springer, Heidelberg (2008)
13. Canetti, R., Halevi, S., Steiner, M.: Mitigating dictionary attacks on password-protected local storage. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 160–179. Springer, Heidelberg (2006)
14. Canetti, R., Varia, M.: Non-malleable obfuscation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 73–90. Springer, Heidelberg (2009)
15. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: 35th ACM STOC, June 9-11, pp. 426–437. ACM Press, San Diego (2003)
16. 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)
17. Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: 30th ACM STOC, May 23-26, pp. 141–150. ACM Press, Dallas (1998)

18. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
19. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The skein hash function family (2008)
20. Fischlin, M.: Security of NMAC and HMAC based on non-malleability. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 138–154. Springer, Heidelberg (2008)
21. Fischlin, M., Fischlin, R.: Efficient non-malleable commitment schemes. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 413–431. Springer, Heidelberg (2000)
22. Knudsen, L.R.: Cryptanalysis of LOKI91. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 196–208. Springer, Heidelberg (1993)
23. Lin, H., Pass, R.: Non-malleability amplification. In: 41st ACM STOC, May 31-June 2, pp. 189–198. ACM Press, Bethesda (2009)
24. Lin, H., Pass, R., Tseng, W.-L.D., Venkitasubramaniam, M.: Concurrent non-malleable zero knowledge proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 429–446. Springer, Heidelberg (2010)
25. Lin, H., Pass, R., Venkitasubramaniam, 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)
26. Lucks, S.: Ciphers secure against related-key attacks. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 359–370. Springer, Heidelberg (2004)
27. Matyas, S., Meyer, C., Oseas, J.: Generating strong one-way functions with cryptographic algorithms. IBM Technical Bulletin (1985)
28. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
29. Ostrovsky, R., Pandey, O., Visconti, I.: Efficiency preserving transformations for concurrent non-malleable zero knowledge. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 535–552. Springer, Heidelberg (2010)
30. Ostrovsky, R., Persiano, G., Visconti, I.: Constant-round concurrent non-malleable zero knowledge in the bare public-key model. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 548–559. Springer, Heidelberg (2008)
31. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 57–74. Springer, Heidelberg (2008)
32. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: 46th FOCS, October 23-25, pp. 563–572. IEEE Computer Society Press, Pittsburgh (2005)
33. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: 37th ACM STOC, May 22-24, pp. 533–542. ACM Press, Baltimore (2005)
34. 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)
35. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS, October 17-19, pp. 543–553. IEEE Computer Society Press, New York (1999)
36. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 239–259. Springer, Heidelberg (2001)

Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols

Douglas Stebila, Lakshmi Kuppusamy, Jothi Rangasamy, Colin Boyd, and Juan Gonzalez Nieto

Information Security Institute, Queensland University of Technology
Brisbane, Queensland, Australia
stebila@qut.edu.au, l.kuppusamy@qut.edu.au, j.rangasamy@qut.edu.au, c.boyd@qut.edu.au, j.gonzaleznieto@qut.edu.au

Abstract. Client puzzles are meant to act as a defense against denial of service (DoS) attacks by requiring a client to solve some moderately hard problem before being granted access to a resource. However, recent client puzzle difficulty definitions (Stebila and Ustaoglu, 2009; Chen et al., 2009) do not ensure that solving n puzzles is n times harder than solving one puzzle. Motivated by examples of puzzles where this is the case, we present stronger definitions of difficulty for client puzzles that are meaningful in the context of adversaries with more computational power than required to solve a single puzzle.

A protocol using strong client puzzles may still not be secure against DoS attacks if the puzzles are not used in a secure manner. We describe a security model for analyzing the DoS resistance of any protocol in the context of client puzzles and give a generic technique for combining any protocol with a strong client puzzle to obtain a DoS-resistant protocol.

Keywords: client puzzles, proof of work, denial of service resistance, protocols.

1 Introduction

Availability of services is an important security property in a network setting. *Denial of service (DoS) attacks* aim to disrupt the availability of servers and prevent legitimate transactions from taking place. One type of DoS attack is *resource depletion*: an attacker makes many requests trying to exhaust the server's resources, such as memory or computational power, leaving the server unavailable to service legitimate requests.

Client puzzles, also called *proofs of work*, can counter resource depletion DoS attacks. Before a server is willing to perform some expensive operation, it demands that the client commit some of its own resources by solving a puzzle. The puzzle should be moderately hard to solve – not as hard as a large factoring problem, for example, but perhaps requiring a few seconds of CPU time. Provided client puzzles are easy for a server to generate and verify, this creates an asymmetry between the amount of work done by a client and the work done by a server.

Although many client puzzle constructions have been proposed, there has been less work in rigourously defining good client puzzles or defining DoS resistance of protocols. The first model for client puzzles was proposed by Jakobsson and Juels [18]. More recently, Stebila and Ustaoglu [31] described a security model for the DoS resistance of key exchange protocols, and Chen et al. [13] proposed a formalization of client puzzles and puzzle difficulty, using a game between a single challenger and a single adversary.

1.1 Contributions and Outline

In this work, we motivate and present stronger notions of security for client puzzles and DoS resistance of protocols and provide several examples satisfying these new definitions.

An Attack on Previous Difficulty Definitions. The main motivation for our stronger notion of security is that it should be hard for an adversary to solve many puzzles, not just one. The existing DoS countermeasure models [18, 13, 31] address the ability of a runtime-bounded adversary to solve a single puzzle, but not of solving multiple puzzles: if one puzzle takes time 2^{20} to solve, for example, will 2^{30} puzzles will take time 2^{50} to solve? This is important in practice, for an adversary will likely have more power than needed to solve a single puzzle.

In order to demonstrate the inadequacy of existing definitions, in Sect. 2 we examine how for some puzzles – the generic puzzle construction of Chen et al. [13], the MicroMint micropayment puzzle scheme [27], and number-theoretic puzzles such as the recent one of Karame-Čapkun [20] – it is hard to solve one instance (satisfying existing definitions [31, 13]), but many instances can be solved without too much more work. This is a weakness in the context of DoS resistance, and so a good puzzle difficulty definition should preclude this.

Stronger Client Puzzles. In Sect. 3, we propose two notions of strong difficulty for client puzzles, one for interactive situations and one for non-interactive situations. These stronger difficulty definitions capture the notion that solving n puzzles should cost about n times the cost of solving one puzzle. We then provide examples of puzzles satisfying these stronger definitions.

DoS-Resistant Protocols. In Sect. 4, we define what it means for a protocol to be DoS-resistant in a multi-user network setting. A server should not perform expensive operations unless a client has done the required work. It should be hard for the work of a legitimate client to be stolen or redirected (avoiding the attack of Mao and Paterson [23]). This generalizes the work of Stebila and Ustaoglu [31] on DoS-resistant key exchange protocols, while also accommodating our stronger notion of security for multiple puzzles as described above. Then, in Sect. 5, we present a theorem that shows how to transform any protocol into a DoS-resistant protocol using a strongly-difficult interactive client puzzle.

We conclude and discuss future work in Sect. 6. Due to length restrictions, proofs of claims appear in the full version [30].

1.2 Related Work

Client Puzzles. Client puzzles were first proposed for protection against DoS attacks (in the form of email spam) by Dwork and Naor [16]. Many client puzzle constructions have subsequently been proposed. There are two main types of client puzzles: *computation-bound* puzzles, which depend on a large number of CPU cycles to solve, and *memory-bound* puzzles [1, 15, 17], which depend on a large number of memory accesses to solve, and which offer more uniform solving time across different CPU speeds compared to computation-bound puzzles. Many computation-bound puzzles are based on the difficulty of inverting a hash function [6, 19, 18, 4, 7, 13], although other techniques (for example, using number-theoretic primitives) exist as well [16, 35, 33, 20]. Puzzle-like constructions also appear in other cryptographic contexts [28, 27, 11] but with a focus on different security properties.

Difficulty of Client Puzzles. Although there have been many puzzle constructions as noted above, only a few of these use any formal notion of security, and there has been little work in developing formal definitions of client puzzle difficulty. The first client puzzle difficulty definition was given by Jakobsson and Juels [18], and another by Canetti et al. [12]. Some memory-bound puzzles [15, 17] include proofs of amortized difficulty.

A richer difficulty definition was given by Chen et al. [13], using two security experiments: unforgeability and puzzle difficulty. Importantly, the difficulty definition only addresses the ability of an adversary to solve a single puzzle. They describe a basic generic client puzzle protocol $\Pi(\mathsf{CPuz})$. Finally, they give a generic client puzzle construction from a pseudorandom function and a one-way function (essentially a MAC and a hash function).

Our definition of puzzle difficulty starts from the Chen et al. [13] definition, but with a number of differences. First, we eliminated the unforgeability property. The unforgeability property is important for their protocol $\Pi(\mathsf{CPuz})$, but is not an essential feature of client puzzles. In fact, to define non-interactive puzzles, in which the client can generate the puzzle itself, we must remove unforgeability. Next, we strengthened the difficulty definition to consider an adversary who solves many puzzles, motivated by our attack in Sect. 2. Our DoS resistance model and protocol is significantly stronger than their protocol $\Pi(\mathsf{CPuz})$, accommodating multiple users in a network setting.

Multiple Puzzles. Our work is motivated by the difficulty of solving multiple puzzles which has not been addressed adequately in previous works. Jakobsson and Juels [18] considered independence of proofs of work, but only in terms of their solvability, not their difficulty. Canetti et al. [12] addressed hardness amplification – the difficulty of solving many instances – of *weakly verifiable puzzles* (WVPs), which are puzzles that need not be publicly verifiable. The adversary for WVPs could not see valid puzzle/solution pairs, so Dodis et al. [14] introduced *dynamic* WVPs that did allow the adversary to see solutions and gave a hardness amplification theorem showing that if solving one dynamic

WVP is hard, then solving an n-wise dynamic WVP is also hard. Still, dynamic WVPs differ from the difficulty definition of Chen et al. [13] and our definition, in that dynamic WVPs generate all puzzle challenges at once, independent of the request, whereas puzzles in the Chen et al. model are generated in response to, and are dependent upon, client requests.

Modelling DoS Attacks on Protocols. Meadows [25] presented a cost-based framework for identifying DoS attacks in network protocols (e.g., Smith et al.'s DoS attack [29] on the JFK key exchange protocol [2]), but can only be used to identify and quantify a DoS attack, not prove that a protocol is DoS-resistant.

Stebila and Ustaoglu [31] gave a provable security model for the DoS resistance of key agreement protocols based on the eCK model for key agreement security [22]. The model splits key exchange into two portions: a presession for the DoS countermeasure, and a session for the key exchange. They give an example protocol using hash function inversions for the DoS countermeasure and building on CMQV [34] for the key exchange protocol. One of their main motivations was to avoid the DoS attack of Mao and Paterson [23] which derived from an authentication failure where messages could be redirected and accepted.

Our definition of DoS resistance for protocols shares some of these characteristics: it uses a presession for the DoS countermeasure and is suitable for a multi-user network setting. It can be used to analyze all protocols, not just key exchange protocols, and it uses a stronger notion of security, considering an adversary who solves many puzzles, not just one. By separating the definition of a puzzle from the definition of a DoS-resistant protocol, we can perform a modular analysis of each component separately and then combine them.

2 Weaknesses in Existing Definitions

In a public network setting, a server will be providing service to many clients at a time. A DoS countermeasure based on client puzzles should require appropriate work to be done for *each* client request: it should not be possible to solve many puzzles easily. While the existing models [18, 31, 13] describe the difficulty of DoS countermeasures when faced with an adversary trying to solve one puzzle, these models do not adequately defend against *powerful adversaries* who can expend more than the effort required to solve a single puzzle.

In this section, we consider some puzzles where a single instance cannot be solved easily by an attacker, satisfying existing difficulty definitions, but where an attacker can solve n puzzles more efficiently than just n times the cost of solving a single puzzle. This motivates our stronger definition of puzzle difficulty in Sect. 3.

While the examples in this section focus on the security definition of Chen et al. [13], they can also be applied to the model of Stebila and Ustaoglu [31].

Generic Puzzle Construction of Chen et al. Chen et al. [13] proposed a generic client puzzle construction based on a pseudorandom function $\mathcal{F}$ and a one-way function ϕ. The challenger selects a secret $s \in \mathcal{K}$ with $|\mathcal{K}| = 2^k$ and public parameters (not relevant to our discussion here), denoted by $*$, to generate a puzzle. The challenger computes $x \leftarrow \mathcal{F}(s, *)$, where $x \in \mathcal{X}$ and $|\mathcal{X}| \geq |\mathcal{K}|$, and then sets $y \leftarrow \phi(x)$. The solver, given the challenge $(y, *)$, has to find a pre-image z such that $\phi(z) = y$.

This generic construction satisfies the puzzle unforgeability and puzzle difficulty security properties provided certain bounds are met: namely, $|\mathcal{X}| \geq |\mathcal{K}|$ and $\frac{|\phi^{-1}(y)|}{|\mathcal{X}|} \leq \frac{1}{2^k}$, for all y. Suppose we have that $|\phi^{-1}(y)| \leq 1$ and $|\mathcal{X}| = 2^k$. Then the bounds in the generic construction are satisfied and solving a single puzzle instance requires approximately 2^k searches in $\mathcal{X}$. But to solve n puzzles, the solver can find the value s with at most 2^k searches and then obtain a solution with one application of $\mathcal{F}$ for each puzzle. That is, solving n puzzles would require $2^k + n$ operations rather than the desired $n \cdot 2^k$ computations.

MicroMint-Based Puzzle. The MicroMint micropayment scheme [27] is effectively a client-puzzle-based micropayment scheme. A coin is a collision in a hash function: it is a pair of values x_1, x_2 such that $H(x_1) = H(x_2)$ for a given hash function H. It is easy to verify the validity of a coin.

Generating coins is harder. If H is a regular (or random) function with ℓ-bit outputs, then to find a collision one must rely on the "birthday paradox" (c.f. [32, §4.2.2]): hash approximately $2^{\ell/2}$ distinct values and search for a collision. This puzzle can be shown to satisfy the puzzle difficulty definition of the Chen et al. model [13] (see the full version of this paper[30]for details).

However, many collisions can be found without too much more work: n collisions can be found with $\sqrt{n} \cdot 2^{\ell/2}$ hash function calls, much less than n times the $2^{\ell/2}$ cost of solving a single puzzle. We emphasize this is not an attack on the MicroMint scheme itself: MicroMint was in fact *designed* so that the amortized cost of generating multiple coins is smaller. While potentially a desirable property in a micropayment scheme, this property is not desirable for client puzzles.

Number-Theoretic Puzzles. Many client puzzles based on number-theoretic constructions have been presented, including the recent scheme of Karame and Čapkun [20], which uses modular exponentiation and argues for security in the Chen et al. model [13] based on the intractability of the RSA problem. Given a puzzle consisting of an RSA modulus N, a challenge x, and a large integer $R >> N$, the solver must compute $x^R \mod N$.

The security argument rests on the assumption that the best known algorithm for this computation requires $O(\log(R))$ modular operations, assuming that factoring N requires more than $O(\log(R))$ operations. For a common puzzle difficulty level of say 2^{20}, a 1024-bit modulus N certainly suffices. But in fact a much smaller N would still suffice and would reduce the computational costs for the verifier, which is important when puzzles are used at extremely low levels in the network stack, such as TCP (e.g., as in [24]).

Even with a smaller N, say 500 bits, the cost of solving a puzzle by computing $x^R \mod N$ is still cheaper than factoring (2^{20} compared to approximately 2^{49} based on the formulas in [5, §6.2]). However, if the adversary wants to solve 2^{30}

puzzles, the best technique is *not* to solve all these puzzles independently (at a cost of $2^{30} \cdot 2^{20} = 2^{50}$ operations) but to first factor N and then use this trapdoor to easily generate solutions (at a cost of $2^{49} + 2^{30}c < 2^{50}$, for some small c which is the cost of easily generating solutions).

3 Strong Client Puzzles

The starting point for our definition of strong client puzzles is the model of Chen et al. [13]. The main differences are as follows.

Firstly, as motivated by Sect. 2, our definition of puzzle difficulty is more robust in that it considers the number of puzzles solved by powerful adversaries.

Secondly, we omit the unforgeability security notion for client puzzles. Inherently, there is no need for puzzles to be unforgeable: in a game played between a challenger and an adversary, the challenger can keep track of all the puzzles issued to detect any forgeries. It is only when *using* puzzles in network protocol that unforgeability sometimes becomes relevant. The main purpose of unforgeability in Chen et al. [13] was to show the DoS resistance of their client puzzle protocol construction $\Pi(\mathsf{Puz})$. We argue in Sect. 4 that a richer notion of DoS resistance is required for a multi-user network setting.

Thirdly, our puzzle definition ensures that the puzzle's semantic meaning – represented by the string str, which may identify the resource the client wishes to access – is the same for both the solver and the verifier. In the model of Chen et al. [13], the server's generation of puz depended on str, but not in a way that the client could verify: puz was an opaque data structure. Thus, a client solving puz could not be certain that this would gain access to the str resource; and similarly, a server receiving a solution for puz could not know that the client solving puz intended to solve a puzzle related to str. This could allow client's work to be stolen by an attacker [31] or redirected [23]. By making a connection between str and puz more transparent, we can incorporate semantic meaning from other protocols or applications into a puzzle.

Fourthly, our security experiment allows for non-publicly verifiable puzzles, as suggested in the notion of weakly verifiable puzzles [12].

Finally, in order to accommodate a variety of puzzle uses, we define two types of difficulty experiments, one for interactive settings and one for non-interactive settings. This accommodates asynchronous applications, such as email, where the client itself generates the puzzle [6, 7]. While the non-interactive definition is more general, it is often convenient to consider the more limited interactive definition because of its simplicity and its more natural use in interactive protocols. We provide examples of puzzles satisfying each type, and interactive puzzles are at the heart of our DoS-resistant protocol construction in Sect. 5.

3.1 Client Puzzles

Definition 1 (Client Puzzle). *A* client puzzle Puz *is a tuple consisting of the following algorithms:*

- $\mathsf{Setup}(1^k)$ *(p.p.t. setup algorithm):*
  1. *Choose the long-term secret key space* sSpace*, puzzle difficulty space* $\mathsf{diffSpace}$*, string space* $\mathsf{strSpace}$*, puzzle space* $\mathsf{puzSpace}$*, and solution space* $\mathsf{solnSpace}$*.*
  2. *Set* $s \leftarrow_R \mathsf{sSpace}$*.*
  3. *Set* $params \leftarrow (\mathsf{sSpace}, \mathsf{puzSpace}, \mathsf{solnSpace}, \mathsf{diffSpace}, \Pi)$*, where* Π *is any additional public information, such as a description of puzzle algorithms, required for the client puzzle.*
  4. *Return* $(params, s)$*.*
- $\mathsf{GenPuz}(s \in \mathsf{sSpace}, d \in \mathsf{diffSpace}, str \in \mathsf{strSpace})$ *(p.p.t. puzzle generation algorithm): Return* $puz \in \mathsf{puzSpace}$*.*
- $\mathsf{FindSoln}(str \in \mathsf{strSpace}, puz \in \mathsf{puzSpace}, t \in \mathbb{N})$ *(probabilistic solution finding algorithm): Return a potential solution* $soln \in \mathsf{solnSpace}$ *after running time at most* t*.*[1]
- $\mathsf{VerSoln}(s \in \mathsf{sSpace}, str \in \mathsf{strSpace}, puz \in \mathsf{puzSpace}, soln \in \mathsf{solnSpace})$ *(d.p.t. puzzle solution verification algorithm): Returns* **true** *or* **false***.*

For *correctness*, we require that if $(params, s) \leftarrow \mathsf{Setup}(1^k)$ and $puz \leftarrow \mathsf{GenPuz}(s, d, str)$, for $d \in \mathsf{diffSpace}$ and $str \in \mathsf{strSpace}$, then there exists $t \in \mathbb{N}$ with $\Pr\left(\mathsf{VerSoln}(s, str, puz, soln) = \mathbf{true} : soln \leftarrow \mathsf{FindSoln}(str, puz, t)\right) = 1$.

3.2 Strong Puzzle Difficulty

A puzzle satisfies strong puzzle difficulty if the probability that a runtime-bounded-adversary can output a list of n fresh, valid puzzle solutions is upper-bounded by a function of the puzzle difficulty parameter and n. This is formalized in the following two experiments for the interactive and non-interactive settings.

We first need to define additional helper oracles as follows:

- $\mathsf{GetPuz}(str)$: Set $puz \leftarrow \mathsf{GenPuz}(s, d, str)$ and record (str, puz) in a list. Return puz.
- $\mathsf{GetSoln}(str, puz)$: If (str, puz) was not recorded by GetPuz, then return $\perp$. Otherwise, find $soln$ such that $\mathsf{VerSoln}(s, str, puz, soln) = \mathbf{true}$. Record $(str, puz, soln)$. Return $soln$.[2]
- $\mathsf{V}(str, puz, soln)$: Return $\mathsf{VerSoln}(s, str, puz, soln)$.

Interactive Strong Puzzle Difficulty. In this setting, we imagine a solver interacting with a challenger: the solver submits a request for a puzzle, the challenger issues a puzzle, the solver sends a solution to the challenger, and the

[1] FindSoln runs in time *at most* t so that a client can stop searching for a puzzle after a specified amount of time; the difficulty definitions in Sect. 3.2 yield that a client must spend *at least* a certain amount of time to find a valid solution.

[2] Note that GetSoln is only obligated to find a solution if puz was actually generated by the challenger. If $\mathcal{A}$ generated puz, then $\mathcal{A}$ may need to employ FindSoln to find a solution. Compared to FindSoln, GetSoln has access to additional secret information that may allow it to find a solution more easily.

challenger checks the solution. The solver can only submit solutions to puzzles that were issued by the challenger: this immediately rules out puzzle forgery or generation of puzzles by the solver. The challenger also allows the solver, via queries, to see solutions to other puzzles.

Let k be a security parameter, let d be a difficulty parameter, let $n \geq 1$, and let $\mathcal{A}$ be an algorithm. The security experiment $\mathsf{Exec}^{\text{INT-STR-DIFF}}_{\mathcal{A},d,\mathsf{Puz}}(k)$ for interactive strong puzzle difficulty of a puzzle Puz is defined as follows:

- $\mathsf{Exec}^{\text{INT-STR-DIFF}}_{\mathcal{A},n,d,\mathsf{Puz}}(k)$:
  1. Set $(params, s) \leftarrow \mathsf{Setup}(1^k)$.
  2. Set $\{(str_i, puz_i, soln_i) : i = 1, \ldots, n\} \leftarrow \mathcal{A}^{\mathsf{GetPuz},\mathsf{GetSoln},\mathsf{V}}(params)$.
  3. If $\mathsf{VerSoln}(s, str_i, puz_i, soln_i) = \textbf{true}$, the tuple (str_i, puz_i) was recorded by GetPuz, and $(str_i, puz_i, soln_i)$ was not recorded by $\mathsf{GetSoln}$ for all $i = 1, \ldots, n$, then return **true**, otherwise return **false**.

Definition 2 (Interactive Strong Puzzle Difficulty). *Let $\epsilon_{d,k,n}(t)$ be a family of functions monotonically increasing in t, where $\epsilon_{d,k,n}(t) \leq \epsilon_{d,k,1}(t/n)$ for all t, n such that $\epsilon_{d,k,n}(t) \leq 1$. Fix a security parameter k and difficulty parameter d. Let $n \geq 1$. Then* Puz *is an* $\epsilon_{d,k,n}(\cdot)$-strongly-difficult interactive client puzzle *if, for all probabilistic algorithms $\mathcal{A}$ running in time at most t,*

$$\Pr\left(\mathsf{Exec}^{\text{INT-STR-DIFF}}_{\mathcal{A},n,d,\mathsf{Puz}}(k) = \textbf{true}\right) \leq \epsilon_{d,k,n}(t) \ .$$

In the random oracle model,[3] To our knowledge, this is the first formal justification for the security of Hashcash. we can define interactive and non-interactive strong puzzle difficulty in terms of the number of oracle queries made by the adversary instead of its running time.

Remark 1. The condition that $\epsilon_{d,k,n}(t) \leq \epsilon_{d,k,1}(t/n)$, for all t and n such that $\epsilon_{d,k,n}(t) \leq 1$, captures the property that solving n puzzles should cost n times the cost of solving one puzzle, at least until the adversary spends enough time t to solve n puzzles with probability 1.

Remark 2. This bound is quite abstract; let us consider a concrete function for $\epsilon_{d,k,n}(t)$. For example, suppose each Puz instance should take approximately 2^d steps to solve. Then we might aim for Puz to be a $\epsilon_{d,k,n}(\cdot)$-strongly-difficult interactive client puzzle, where $\epsilon_{d,k,n}(t) \approx t/2^d n + \text{negl}(k)$.

Remark 3. In the security experiment, the adversary is allowed to request many more than n puzzles using GetPuz. The adversary can then pick which n puzzles it submits as its allegedly solved puzzles $\{(str_i, puz_i, soln_i) : i = 1, \ldots, n\}$. In other words, the adversary could request many puzzles and hope to find some easy-to-solve instances. This means, for example, that puzzles for which 1% of instances are trivially solved could not be proven secure (with a reasonable $\epsilon_{d,k,n}(t)$) according to this difficulty definition.

[3] In the *random oracle model*, a hash function is modelled as an ideal random function accessible to the adversary solely as an oracle. [10].

Relation to Examples from Sect. 2. The Chen et al. generic puzzle construction in Sect. 2 does not satisfy our definition of strong puzzle difficulty. From Theorem 2 of [13], we have that the Chen et al. generic construction is $\epsilon_{d,k}(t)$-difficult, with $\epsilon_{d,k}(t) \lessapprox 2\nu_k(t) + (1 + t/(2^{k-t}))\gamma_d(t)$, where $\nu_k(t)$ is the probability of breaking the pseudorandom function family (with security parameter k) in time t and $\gamma_d(t)$ is the probability of breaking the one-way function (with security parameter d) in time t. By the argument from Sect. 2, there exists an adversary that can win the strongly-difficulty interactive puzzle game with probability at least $\epsilon'_{d,k,n}(t) \gtrapprox \nu_k(t) + \gamma_d(t)/n$, which does not satisfy $\epsilon'_{d,k,n}(t) \leq \epsilon'_{d,k,1}(t/n)$.

Similarly, the MicroMint puzzle from Sect. 2 does not satisfy Definition 2. Finding a single ℓ-bit collision (and thus solving a MicroMint puzzle) requires about $2^{\ell/2}$ hash function calls, but finding n collisions requires only $\sqrt{n} \cdot 2^{\ell/2}$ calls. Let $\epsilon_{k,\ell,n}(q) = \frac{q}{\sqrt{n} \cdot 2^{\ell/2}}$. It is clear that, for $n \geq 2$, $\epsilon_{k,\ell,n}(q) > \epsilon_{k,\ell,1}(q/n)$, and hence MicroMint is not an $\epsilon_{k,\ell,n}(\cdot)$-strongly difficulty interactive puzzle.

Non-Interactive Strong Puzzle Difficulty. Non-interactive strong puzzle difficulty models the case of client-generated puzzles. Besides being useful in their originally proposed setting as an email spam countermeasure [6, 7], they can be useful in protocols that are inherently asynchronous, such as the Internet Protocol (IP), or have a fixed message flow, such as the Transport Layer Security (TLS) protocol.

The technical difference between interactive and non-interactive strongly difficult puzzles is whether the adversary can return solutions only to puzzles generated by the challenger (interactive) or can also return solutions to puzzles it generated itself (non-interactive).

The security experiment $\mathsf{Exec}^{\text{NINT-STR-DIFF}}_{\mathcal{A},n,d,\mathsf{Puz}}(k)$ for non-interactive strong puzzle difficulty is as in the interactive case with a change to line 3 of the experiment:

– $\mathsf{Exec}^{\text{NINT-STR-DIFF}}_{\mathcal{A},n,d,\mathsf{Puz}}(k)$:
   3. If $\mathsf{VerSoln}(s, str_i, puz_i, soln_i) =$ **true** and the tuple $(str_i, puz_i, soln_i)$ was not recorded by $\mathsf{GetSoln}$ for all $i = 1, \ldots, n$, then return **true**, otherwise return **false**.

The definition of $\epsilon_{d,k,n}(\cdot)$-*strongly-difficult non-interactive client puzzles* follows analogously.

Remark 4. If Puz is an $\epsilon_{d,k,n}(\cdot)$-strongly-difficult non-interactive puzzle, then it is also $\epsilon_{d,k,n}(\cdot)$-strongly-difficult interactive puzzle.

3.3 A Strongly-Difficult Interactive Client Puzzle Based on Hash Functions

In this section, we describe a client puzzle based on hash function inversion, similar to the subpuzzle used by Juels and Brainard [19] or the partial inversion proof of work of Jakobsson and Juels [18].

Let $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a hash function. Define $\mathbf{SPuz}_H$ be the following tuple of algorithms:

– $\mathsf{Setup}(1^k)$: Set $\mathsf{sSpace} \leftarrow \{\perp\}$, $\mathsf{diffSpace} \leftarrow \{0, 1, \ldots, k\}$, $\mathsf{strSpace} \leftarrow \{0, 1\}^*$, $\mathsf{puzSpace} \leftarrow \{0, 1\}^* \times \{0, 1\}^k$, $\mathsf{solnSpace} \leftarrow \{0, 1\}^*$, and $s \leftarrow \perp$.
– $\mathsf{GenPuz}(\perp, d, str)$: Set $x \leftarrow_R \{0, 1\}^k$; let x' be the first d bits of x and x'' be the remaining $k - d$ bits of x. Set $y \leftarrow H(x, d, str)$. Return $puz \leftarrow (x'', y)$.
– $\mathsf{FindSoln}(str, (x'', y), t)$: For z from 0 to $\max\{t, 2^d - 1\}$: set $soln \leftarrow z$ (in $\{0, 1\}^d$); if $H(soln||x'', d, str) = y$ then return $soln$.
– $\mathsf{VerSoln}(\perp, str, (x'', y), soln)$: If $H(soln||x'', d, str) = y$ then return **true**, otherwise return **false**.

Theorem 1. *Let H be a random oracle. Let $\epsilon_{d,k,n}(q) = \left(\frac{q+n}{n2^d}\right)^n$. Then* $\mathbf{SPuz}_H$ *is an $\epsilon_{d,k,n}(q)$-strongly-difficult interactive client puzzle, where q is the number of distinct queries to H.*

The proof follows a counting argument and appears in the full version [30].

3.4 Hashcash Is a Strongly-Difficult Non-interactive Client Puzzle

In this section, we show that one of the earliest client puzzles, Hashcash [6, 7], satisfies the definition of a strongly-difficult non-interactive client puzzle in the random oracle model.

While Hashcash was originally proposed to reduce email spam, the current specification (stamp format version 1 [7]) can be applied to any resource. Hashcash is non-interactive: the puzzle is generated by the same person who solves the puzzle. Hence it should be difficult for a client to generate a puzzle that can be easily solved. Hashcash is based on the difficulty of finding a partial preimage of a string starting with a certain number of zeros in the SHA-1 hash function.

A Hashcash *stamp* is a string of the form `ver:bits:date:resource:[ext]:rand:counter`. The field `bits` denotes the "value" of the stamp (the number of zeros at the start of the output) and `counter` is the solution to the puzzle. A stamp is *valid* if $H(stamp)_{[1\ldots\mathtt{bits}]} = 0\ldots0$. In the context of real-world email applications, there may be additional restrictions on the validity of a stamp, such as whether `date` is within a reasonable range and whether the email address (`resource`) specified is acceptable.

Theorem 2. *Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$, where $k \geq d$, be a random oracle. Let $\epsilon_{d,k,n}(q) = \frac{q+n}{n2^d}$. Then Hashcash is an $\epsilon_{d,k,n}(q)$-strongly-difficult non-interactive puzzle, where q is the number of queries made by $\mathcal{A}$ to H.*

The proof follows a counting argument and appears in the full version [30].

4 Denial-of-Service Resistance of Protocols

Although we have defined what a good client puzzle is, it does not immediately follow that using a good client puzzle in a protocol yields DoS resistance. In this section, we describe what it means for a protocol to be DoS-resistant, and in the subsequent section we give a generic construction for DoS-resistant protocols.

Our approach begins similar to that of Stebila and Ustaoglu [31]. We work in an adversary-controlled multi-user communication network.[4] The adversary's goal is to cause a server to commit resources without the adversary itself having done the work to satisfy the denial of service countermeasure.

Protocol Execution. A protocol is a message-driven interaction, taking place among disjoint sets of clients $\mathsf{Clients}$ and servers $\mathsf{Servers}$, where each *party* is a probabilistic polynomial-time Turing machine. An execution of the protocol is called a *presession.* During execution, each party $\hat{U}$ may have multiple instances of the protocol running, with each instance indexed by a value $i \in \mathbb{Z}_+$; these instances are denoted by $\Pi_i^{\hat{U}}$. A protocol consists of the following algorithms:

– $\mathsf{GlobalSetup}(1^k)$ (p.p.t. protocol setup algorithm): Select the long-term secret key space $\rho\mathsf{Space}$. Choose global public parameters Π of the scheme and return $params \leftarrow (\rho\mathsf{Space}, \Pi)$; this is assumed to be an implicit input to all remaining algorithms.
– $\mathsf{ServerSetup}(\hat{S} \in \mathsf{Servers})$ (p.p.t. party setup algorithm): Select $\rho_{\hat{S}} \in \rho\mathsf{Space}$. Perform any additional setup required by *params*.
– $\mathsf{CAction}j(\hat{C} \in \mathsf{Clients}, i \in \mathbb{Z}_+, m_{j-1}, M'_{j-1})$, for $j = 1, \ldots$ (p.p.t. protocol client action algorithm): Instance i of party $\hat{C}$ produces its jth protocol message for the run of the protocol, based on the instance's previous private state m_{j-1} and the received message M'_{j-1}. The output (M_j, m_j) consists of its outgoing message M_j and its new private state m_j.
– $\mathsf{SAction}j(\hat{S} \in \mathsf{Servers}, i \in \mathbb{Z}_+, m'_{j-1}, M_j)$, for $j = 1, \ldots$ (p.p.t. protocol server action algorithm): Instance i of party $\hat{S}$ produces its jth protocol message for this instance, based on $\hat{S}$'s long-term secret, the previous private state m'_{j-1}, and the received message M_j. The output (M'_j, m'_j) consists of its outgoing message M'_j and its new private state m'_j.

The client is assumed to be the initiator. An instance records its current progress through the protocol with the value j of the last completed action.

Presessions. After receiving some sequence of $\mathsf{SAction}j(\hat{S}, i, \ldots)$ calls, a server instance will either *accept* or *reject*; if it accepts, it outputs a *presession* identified by a tuple of the form $[\hat{C}, \hat{S}, \tau]$, where $\hat{C}$ is the *partner* and τ is a sequence of messages. The sequence of messages τ is meant to act like a transcript; however, since in DoS-resistant protocols a server may not store state early in the protocol, portions of τ could have been forged by an adversary. Accepted presessions must be unique within a party. Additionally, since the protocol may be used for another purpose – key agreement, electronic voting, etc. – we do not require that the protocol terminate after accepting, and indeed expect that it may continue to perform some additional application-level functionality.

[4] It is true that, in an adversary-controlled network, the adversary can deny service simply by not relaying messages. Our concern, however, is with resource depletion attacks in which a server is overwhelmed with requests.

Correctness. A protocol is *correct* if, for all $\hat{C} \in \mathsf{Clients}$ and $\hat{S} \in \mathsf{Servers}$ who follow the protocol, there exists a running time t for $\hat{C}$ such that, when messages are relayed faithfully between $\hat{C}$ and $\hat{S}$, $\hat{S}$ will accept with probability 1. In other words, clients can eventually do enough work to make connections.[5]

Denial of Service Countermeasure. To provide DoS resistance, a protocol will typically include some test so the server can decide, based on the proposed presession $[\hat{C}, \hat{S}, \tau]$ and its secret ρ, whether to accept or reject based on some DoS countermeasure in the protocol. It is the adversary's goal to cause a server to accept without the adversary having faithfully followed the protocol.

Adversary's Powers. The adversary controls all communication links and can send, create, modify, delay, and erase messages to any participants. Additionally, the adversary can learn private information from parties or cause them to perform certain actions.

The following queries model how the adversary interacts with the parties:

– $\mathsf{Send}(\hat{U}, i, M)$: The adversary sends message M to instance i of $\hat{U}$ who performs the appropriate protocol action (either $\mathsf{CAction}j(\hat{U}, i, m, M)$ or $\mathsf{SAction}j(\hat{U}, i, m, M)$ based on the instance's last completed action $j - 1$), updates its state, and returns its outgoing message, if any.
– $\mathsf{Expose}(\hat{S})$: The adversary obtains $\hat{S}$'s secret value $\rho_{\hat{S}}$; mark $\hat{S}$ as *exposed*.

Security Definition. The basic idea of the security definition is as follows: the amount of credit the adversary gets in terms of accepted presessions should not be greater than the amount of work the adversary itself did. An important part of the definition below is solutions from legitimate clients.

An instance $\Pi_i^{\hat{S}}$ that has accepted a presession $[\hat{C}, \hat{S}, \tau]$ is said to be *fresh* provided that $\hat{S}$ was not exposed before $\hat{S}$ accepted this presession and there does not exist an instance $\Pi_j^{\hat{C}}$ which has a matching conversation [9] for τ. (Intuitively, a "fresh" instance is an attackable instance, one that has not been trivially solved by exposing the server's private information.)

Let k be a security parameter, let $n \geq 1$, and let $\mathcal{A}$ be a probabilistic algorithm. The security experiment $\mathsf{Exec}^{\text{DOS}}_{\mathcal{A},n,P}(k)$ for DoS resistance of a protocol P is defined as follows:

– $\mathsf{Exec}^{\text{DOS}}_{\mathcal{A},n,P}(k)$: Run $\mathsf{GlobalSetup}(k)$. For each $\hat{S} \in \mathsf{Servers}$, run $\mathsf{ServerSetup}(\hat{S})$. Run $\mathcal{A}(params)$ with oracle access to Send and Expose. If, summing over all servers, the number of fresh instances accepted is n, then return **true**, otherwise return **false**.

A protocol is DoS-resistant if the probability that an adversary with bounded runtime can cause a server to accept n fresh presessions is bounded:

Definition 3 (Denial-of-service-resistant Protocol). *Let $\epsilon_{k,n}(t)$ be a family of functions that are monotonically increasing in t, where $\epsilon_{k,n}(t) \leq \epsilon_{k,1}(t/n)$*

[5] Limits on the amount of work done by the server come later, in Definition 3.

for all t, n *such that* $\epsilon_{k,n}(t) \leq 1$*. Fix a security parameter* k*. Let* $n \geq 1$*. We say that a protocol* P *is* $\epsilon_{k,n}(\cdot)$-denial-of-service-resistant *if*

1. *for all probabilistic algorithms* $\mathcal{A}$ *running in time at most* t*,*

$$\Pr\left(\mathsf{Exec}^{\text{DOS}}_{\mathcal{A},n,P}(k) = \mathbf{true}\right) \leq \epsilon_{k,n}(t) + \text{negl}(k) \ \text{, and}$$

2. *no call to* $\mathsf{SAction}j_P(\hat{S}, i, m, M)$ *results in an expensive operation unless* $\Pi_i^{\hat{S}}$ *has accepted.*

The second aspect addresses the idea that a server should not perform expensive operations unless the countermeasure has been passed. As the notion of "expensive" can vary from setting to setting, we leave it vague, but it can easily be formalized, for example by using Meadows' cost-based framework [25].

Avoiding Client Impersonations. Though a DoS countermeasure does not provide explicit authentication, we still wish to avoid impersonations. For example, suppose a client $\hat{C}$ sends messages meant to prove its legitimate intentions in communicating with server $\hat{S}$. It should not be possible for an adversary to easily use those messages to cause another server $\hat{S}'$ to perform expensive operations, nor should it be possible for an adversary to easily use those messages to convince $\hat{S}$ that a different client $\hat{C}'$ intended to communicate with $\hat{S}$.

This is prevented by the model since party names are included in the presession identifiers. If an adversary observed a presession $[\hat{C}, \hat{S}, \tau]$ and then tried to use that information to construct a presession $[\hat{C}', \hat{S}, \tau']$ of another user $\hat{C}'$ with the same server, then this new presession would be unexposed and the adversary would be prohibited from easily causing a server to accept it by Definition 3. This in effect requires a binding of values in the DoS countermeasure transcript τ to the parties – $\hat{C}$ and $\hat{S}$ – in question.

Avoiding Replay Attacks. We follow the approach of Stebila and Ustaoglu [31] in dealing with replay attacks, where replay attacks are avoided by uniqueness of presession identifiers of accepted presessions. This does mean that the server has to store a table of presession identifiers, but this does not constitute a vector for a DoS attack because the server only stores a presession identifier after it accepts a presession, so it is doing an expensive operation only after the DoS countermeasure has been passed.

5 Building DoS-Resistant Protocols from Client Puzzles

In this section, we present a generic technique that transforms any protocol P into a DoS-resistant protocol $D(P)$. Our technique uses strongly-difficult interactive client puzzles as a DoS countermeasure and message authentication codes for integrity of stateless connections [3]. We prove that the combined protocol $D(P)$ is a DoS-resistant protocol.

The client and server each provide nonces and construct the string *str* using their names, nonces, and any additional information, such as a timestamp or information from a higher-level protocol. The server generates a puzzle from *str*, authenticates the puzzle using the message authentication code (to avoid storing state), and sends it to the client. The client solves the puzzle using its own string *str* and sends the solution to the server. The server checks the message authentication code and the correctness of the solution. Finally, the server checks that the presession is unique and accepts. The messages for the DoS countermeasure are interleaved, where possible, with the messages of the main protocol, and after the countermeasure has accepted the main protocol continues as needed.

Specification. Let P be a protocol such that $\mathsf{SAction1}_P$ does not involve any expensive operations. Let k be a security parameter. Let $\mathsf{MAC} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$ be a family of secure message authentication codes [8]. Let $\mathsf{Puz} = (\mathsf{Setup}, \mathsf{GenPuz}, \mathsf{FindSoln}, \mathsf{VerSoln})$ be a strongly-difficult interactive client puzzle with long-term secret key space $\mathsf{sSpace} = \{\perp\}$ (there is no long-term secret key for puzzles). Although this may seem restrictive, many puzzles satisfy this constraint, including the hash-based puzzle in Sect. 3.3. Fix a DoS difficulty parameter $d \in \mathsf{diffSpace}$.

Let $D(P)_{\mathsf{Puz},d,\mathsf{MAC},k}$ be the protocol consisting of the following algorithms:

– $\mathsf{GlobalSetup}(1^k)$: Set $\rho\mathsf{Space} \leftarrow \{0,1\}^k$ and $\mathsf{NonceSpace} \leftarrow \{0,1\}^k$.
– $\mathsf{ServerSetup}(\hat{S} \in \mathsf{Servers})$: Set $mk_{\hat{S}} \leftarrow_R \{0,1\}^k$ and $\rho_{\hat{S}} \leftarrow mk_{\hat{S}}$.
– $\mathsf{CAction}j_{D(P)}(\dots), \mathsf{SAction}j_{D(P)}(\dots)$: As specified by the protocol in Figure 1.

Remark 5. The construction $D(P)$ requires that $\mathsf{SAction1}_P$ not involve any expensive operations, as $\mathsf{SAction1}_P$ is called by $\mathsf{SAction1}_{D(P)}$ before the server instance has accepted. If $\mathsf{SAction1}_P$ does in fact involve expensive operations, then P would need to be rewritten so that the expensive operation is delayed until $\mathsf{SAction2}_P$. In other words, the $D(P)$ construction may result in an additional round being added before the P protocol is run; this should not be surprising.

Additionally, $\mathsf{SAction1}_P$ may result in a private output m_1' which the server instance needs to store until the next message is received. If state storage is considered an expensive operation (as it could be a vector for a resource depletion DoS attack), then there are two options: use a stateless connection [3] to encrypt m_1' and send it to the client who must return it in the following round, or, as above, rewrite P so as to delay the operation until $\mathsf{SAction2}_P$.

Theorem 3. *Let P be a protocol such that $\mathsf{SAction1}_P$ does not involve any expensive operations. Suppose that Puz is an $\epsilon_{d,k,n}(t)$-strongly-difficult interactive puzzle with long-term secret key space $\mathsf{sSpace} = \{\perp\}$ and that MAC is a family of secure message authentication codes. Then $D(P)_{\mathsf{Puz},d,\mathsf{MAC},k}$ is an $\epsilon'_{d,k,n}(t)$-denial-of-service-resistant protocol, for $\epsilon'_{k,n}(t) = \epsilon_{d,k,n}(t + t_0 q_{\mathsf{Send}}) + \mathrm{negl}(k)$, where q_{Send} is the number of Send queries issued and t_0 is a constant depending on the protocol, assuming $t \in \mathrm{poly}(k)$.*

$D(P)_{\mathsf{Puz},d,\mathsf{MAC},k}$ – $\mathsf{Send}(\hat{U}, i, M)$ protocol specification			
	Client $\hat{C}$		Server $\hat{S}$ long-term secret: $\rho_{\hat{S}} = mk_{\hat{S}}$
	$\underline{\mathsf{CAction1}_{D(P)}}$:		
1.	$N_C \leftarrow_R \mathsf{NonceSpace}$		
2.	$(M_1, m_1) \leftarrow \mathsf{CAction1}_P()$	$\xrightarrow{\hat{C}, N_C, M_1}$	$\underline{\mathsf{SAction1}_{D(P)}}$:
3.			$N_S \leftarrow_R \mathsf{NonceSpace}$
4.			$(M'_1, m'_1) \leftarrow \mathsf{SAction1}_P(M_1)$
5.			$str \leftarrow (\hat{C}, \hat{S}, N_C, N_S, M_1, M'_1)$
6.			$puz \leftarrow \mathsf{GenPuz}(\perp, d, str)$
7.	$\underline{\mathsf{CAction2}_{D(P)}}$:	$\xleftarrow{N_S, M'_1, puz, \sigma}$	$\sigma \leftarrow \mathsf{MAC}_{mk_{\hat{S}}}(str, puz)$
8.	$str \leftarrow (\hat{C}, \hat{S}, N_C, N_S, M_1, M'_1)$		
9.	$soln \leftarrow \mathsf{FindSoln}(str, puz, t)$		
10.	$(M_2, m_2) \leftarrow \mathsf{CAction2}_P(m_1, M'_1)$	$\xrightarrow{str, puz, \sigma, soln}$	$\underline{\mathsf{SAction2}_{D(P)}}$:
11.			reject if $\sigma \neq \mathsf{MAC}_{mk_{\hat{S}}}(str, puz)$
12.			reject if $\neg\mathsf{VerSoln}(\perp, str, puz, soln)$
13.			$\tau \leftarrow (N_C, N_S, M_1, M'_1, puz, soln)$
14.			verify no stored presession $[\hat{C}, \hat{S}, \tau]$
15.			accept and store presession $[\hat{C}, \hat{S}, \tau]$
	continue with $\mathsf{CActionj}_P$		continue with $\mathsf{SActionj}_P$

Fig. 1. $D(P)_{\mathsf{Puz},d,\mathsf{MAC},k}$ DoS countermeasure protocol

The proof of Theorem 3 follows by a sequence of games, first replacing the message authentication code with a MAC challenger, and then replacing the puzzles with a Puz challenger. Fresh accepted presessions correspond to valid solutions to the Puz challenger, yielding the bound relating the protocol and the puzzle. The details appear in the full version [30].

6 Conclusion

Our goal in this work was to improve security definitions for client puzzles and denial-of-service-resistant protocols. We presented a new, stronger definition of puzzle difficulty for client puzzles, motivated by examples considering the effects of an adversary who has enough resources to solve more than one puzzle. This definition is sufficiently general to be useful for analyzing and proving the difficulty of a wide range of computation- and memory-bound puzzle constructions.

Whereas the client puzzle difficulty definition suffices for a simple game between a challenger and an adversary, we need something more advanced for a multi-user network setting. Thus, we introduced a new definition of DoS resistance for network protocols.

Our work can be viewed in part as combining the client puzzles approach of Chen et al. [13] and the DoS-resistant protocols approach of Stebila and Ustaoglu, extending both to provide stronger DoS resistance and better modularity.

To demonstrate the utility of our new definitions, we have included examples of two hash-based client puzzles (including an analysis of the Hashcash client puzzle) and given a generic technique for converting any protocol into a DoS-resistant protocol using an interactive client puzzle.

Future Work. The interactive request-challenge-solution nature of client puzzles in the Chen et al. definition [13] and our Definition 2 is incompatible with the definition of dynamic weakly verifiable puzzles [14], so the hardness amplification theorem from one to many puzzles does not apply. An important theoretical question arising is the development of a hardness amplification theorem for client puzzles that is suitable, and avoids the counterexamples from Sect. 2 when going from the Chen et al. definition [13] to our Definition 3.2.

Key agreement is the most widely deployed cryptographic protocol on the Internet, and, as a computationally-expensive operation, is a possible attack vector for DoS attacks. Some Internet key agreement protocols – such as IKEv2 [21], the Host Identity Protocol (HIP) [26], and Just Fast Keying (JFK) [2] – have been designed with DoS attacks in mind. An important future work to be undertaken is the formal analysis of the DoS resistance of these protocols using an approach such as the one we have presented.

Acknowledgements. The authors are grateful for helpful suggestions from anonymous referees. This work was supported by the Australia-India Strategic Research Fund project TA020002.

References

[1] Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. In: Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 2003, Internet Society, San Diego (2003)

[2] Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A.D., Reingold, O.: Just Fast Keying: Key agreement in a hostile Internet. ACM Transactions on Information and System Security 7(2), 1–30 (2004)

[3] Aura, T., Nikander, P.: Stateless connections. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 87–97. Springer, Heidelberg (1997)

[4] Aura, T., Nikander, P., Leiwo, J.: DOS-resistant authentication with client puzzles. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2000. LNCS, vol. 2133, pp. 170–177. Springer, Heidelberg (2001)

[5] Babbage, S., Catalano, D., Cid, C., Dunkelman, O., Gehrmann, C., Granboulan, L., Lange, T., Lenstra, A., Nguyen, P.Q., Paar, C., Pelzl, J., Pornin, T., Preneel, B., Rechberger, C., Rijmen, V., Robshaw, M., Rupp, A., Smart, N., Ward, M.: ECRYPT yearly report on algorithms and keysizes (2007-2008) (2008), http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf

[6] Back, A.: A partial hash collision based postage scheme (1997), http://www.hashcash.org/papers/announce.txt

[7] Back, A.: Hashcash (2004), http://www.hashcash.org/docs/hashcash.html

[8] Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. Journal of Computer and System Sciences 61(3), 362–399 (2000)
[9] 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)
[10] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proc. 1st ACM Conference on Computer and Communications Security (CCS), pp. 62–73. ACM, New York (1993)
[11] Boyen, X.: Halting password puzzles: Hard-to-break encryption from human-memorable keys. In: Proc. 16th USENIX Security Symposium, pp. 119–134 (2007)
[12] 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)
[13] Chen, L., Morrissey, P., Smart, N.P., Warinschi, B.: Security notions and generic constructions for client puzzles. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 505–523. Springer, Heidelberg (2009)
[14] 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)
[15] Dwork, C., Goldberg, A., Naor, M.: On memory-bound functions for fighting spam. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 426–444. Springer, Heidelberg (2003)
[16] 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)
[17] 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)
[18] Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols (extended abstract). In: Preneel, B. (ed.) Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security. IFIP Conference Proceedings, vol. 152, pp. 258–272. Kluwer, Dordrecht (1999), http://www.rsa.com/rsalabs/node.asp?id=2049
[19] Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 1999, pp. 151–165. Internet Society, San Diego (1999)
[20] Karame, G.O., Capkun, 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)
[21] Kaufman, C.: Internet Key Exchange (IKEv2) protocol, RFC 4306 (2005)
[22] LaMacchia, B., 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)
[23] Mao, W., Paterson, K.G.: On the plausible deniability feature of Internet protocols (2002) (manuscript), http://citeseer.ist.psu.edu/678290.html
[24] McNevin, T.J., Park, J.M., Marchany, R.: pTCP: A client puzzle protocol for defending against resource exhaustion denial of service attacks. Technical Report TR-ECE-04-10, Department of Electrical and Computer Engineering, Virginia Tech (2004), http://www.arias.ece.vt.edu/publications/TechReports/mcNevin-2004-1.pdf

[25] Meadows, C.: A formal framework and evaluation method for network denial of service. In: Proc. 12th IEEE Computer Security Foundations Workshop (CSFW), p. 4. IEEE, Los Alamitos (1999)

[26] Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.R.: Host Identity Protocol, RFC 5201 (2008)

[27] Rivest, R.L., Shamir, A.: PayWord and MicroMint: Two simple micropayment schemes. In: Lomas, M. (ed.) Security Protocols 1996. LNCS, vol. 1189, pp. 69–87. Springer, Heidelberg (1997)

[28] Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Tech. Rep. TR-684, MIT Laboratory for Computer Science (1996), http://people.csail.mit.edu/rivest/RivestShamirWagner-timelock.pdf

[29] Smith, J., González Nieto, J., Boyd, C.: Modelling denial of service attacks on JFK with Meadows's cost-based framework. In: Buyya, R., Ma, T., Safavi-Naini, R., Steketee, C., Susilo, W. (eds.) Proc. 4th Australasian Information Security Workshop – Network Security (AISW-NetSec) 2006. CRPIT, vol. 54, pp. 125–134. Australian Computer Society (2006)

[30] Stebila, D., Kuppusamy, L., Rangasamy, J., Boyd, C., Gonzalez Nieto, J.: Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. Cryptology ePrint Archive (2010) (full version), http://eprint.iacr.org/

[31] Stebila, D., Ustaoglu, B.: Towards denial-of-service-resilient key agreement protocols. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 389–406. Springer, Heidelberg (2009)

[32] Stinson, D.R.: Cryptography: Theory and Practice, 2nd edn. Chapman & Hall, Boca Raton (2002)

[33] Tritilanunt, S., Boyd, C., Foo, E., González Nieto, J.: Toward non-parallelizable client puzzles. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 247–264. Springer, Heidelberg (2007)

[34] Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Designs, Codes and Cryptography 46(3), 329–342 (2008)

[35] Waters, B., Juels, A., Halderman, J.A., Felten, E.W.: New client puzzle outsourcing techniques for DoS resistance. In: Pfitzmann, B., Liu, P. (eds.) Proc. 11th ACM Conference on Computer and Communications Security (CCS), pp. 246–256. ACM, New York (2004)

On Shortening Ciphertexts: New Constructions for Compact Public Key and Stateful Encryption Schemes

Joonsang Baek, Cheng-Kang Chu, and Jianying Zhou

Institute for Infocomm Research
1 Fusionopolis Way, #21-01 Connexis
Singapore 138632
{jsbaek,ckchu,jyzhou}@i2r.a-star.edu.sg

Abstract. We present new constructions of (conventional) public key and stateful public key encryption schemes which produce ciphertexts of compact size while providing both efficiency and strong security. Our public key encryption scheme incurs only one group element ciphertext expansion (defined as the size of the ciphertext minus the size of the plaintext message) but compared with the previous scheme in the literature, its encryption algorithm is more efficient. Our stateful encryption scheme resolves the problem of ciphertext expansion of the existing schemes in the literature and hence can be served as a favorable alternative. Both of our schemes do not depend on the external length-preserving cipher constructed from the expensive strong pseudo random permutation. We provide security analysis of our schemes against chosen ciphertext attack under the well-known computational assumptions, in the random oracle model. We envision that our schemes can serve as efficient public key primitives suitable for implementing on resource-constrained devices.

1 Introduction

Motivation. As the era of post-desktop computing is advancing, the role of small devices such as sensors, mobile phones, or PDAs is becoming crucially important. On the one hand many applications running on those small, inexpensive, and networked devices can benefit us, but on the other hand they have potential to create new vulnerabilities that attackers can exploit. However, protecting small devices by providing cryptographic solutions is not always easy: In tiny devices like sensors, resources for computations and communications are so constrained that cryptographic algorithms that need heavy computations and communications can hardly to be realized. Even in the small devices of moderate size, such as mobile phones, significant amount of computations for security services should be prohibited as this is directly related to the battery life.

In this regard, it was believed that public key cryptographic algorithms are too heavy to be implemented on tiny devices (especially, sensors). In contrast to this general belief, more and more public key algorithms [3,26,27] even including pairing-based algorithms [25,24] have been implemented on wireless sensors recently. Although the advance of hardware capacity of the small devices [15] could be one of the reasons for these successful implementations, the efforts to optimize the underlying algorithms and/or to create a new framework for constructing secure yet efficient public key schemes might be a bigger reason. One notable example of the latter is Bellare, Kohno and Shoup (BKS)'s [5] stateful public key encryption (StPKE) scheme, which is to reduce the cost of public key encryption by allowing a sender to maintain state which is reused across different sessions.

Based on our experiences and the previous work in the literature, we summarize what requirements public key encryption schemes should satisfy to be suitable for resource-constrained environments:

– Low communication overhead: As the small devices are lack of sufficient and frequent supply of energy, reducing communication overhead is as important as reducing computational overhead. According to [26], power to transmit one bit in the "Mica2Dot" sensor is equivalent to approx. 2,090 clock cycles of execution on the microcontroller. In terms of energy consumption, [26] reports that receiving 1 byte of data using the Mica2Dot [14] sensor takes 28.6 μJ of energy while sending 1 byte of data consumes 59.2 μJ. Hence, reducing 10 bytes from the data which are to be sent will save significant energy and, consequently contribute to longer battery life. Therefore, it is essential to design encryption schemes which produce compact ciphertexts to keep the "ciphertext expansion" (defined as the size of the ciphertext minus the size of the plaintext message) as small as possible. As will be discussed in the next subsection, surprisingly few PKE schemes have minimal ciphertext expansion without incurring additional computational overhead. We also note that the StPKE scheme based on DHIES [1] is very efficient in terms of computation, but has inevitable ciphertext expansion.
– Low computational overhead: On sensors, it is quite common to encrypt short messages, say, 10-20 bytes in length, such as password, collected environment data (such as temperature) and so on. If possible, it is preferable to encrypt the short messages without padding or pre-processing, in order not to cause additional overhead for computation. For this reason, avoiding ciphertext expansion by using a length-preserving DEM (Data Encapsulation Mechanism) for KEM (Key Encapsulation Mechanism)/DEM framework [13] is undesirable in resource-constrained environment as the PRP (Pseudo Random Permutation) [19] used to construct such length-preserving DEM is complicated and inefficient [2,10], moreover, it requires the length of plaintext to exceed certain length.
– Efficient encryption algorithm: Although this seems to be an obvious requirement, it should be emphasized that in many situations, efficiency of encryption may be more important than efficiency of decryption. This is because, it is often the case that small devices with limited resources encrypt messages and send the resulting ciphertexts to the base stations with higher computational

capacity, which will decrypt the ciphertexts upon receiving them. (Note that this is one of the typical settings in wireless sensor network.)
– Reasonable but reliable security level: The above requirements could trivially be satisfied by weakening security, say, by reducing the key size or/and security requirements dramatically. We do not consider this as an option to gain efficiency of cryptographic algorithms. In terms of encryption schemes, we still require them to be secure against chosen ciphertext attack (CCA) at least, however, in the random oracle model [7].

Motivated by the above requirements and the current status of providing small devices with strong security based on public key cryptographic algorithms mentioned earlier, in this paper, we focus on designing even more efficient PKE and StPKE schemes that produce *compact* ciphertexts. In what follows, we review the related work on compact PKE schemes.

Related Work. In the literature, there have been several proposals of constructing PKE schemes that produce the reduced length of ciphertexts while providing security against chosen ciphertext attack, which we call "CCA-security". Kurosawa and Matsuo [21] presented a variant of DHIES [1], which produces shorter ciphertexts than DHIES. Their technique is essentially to use the KEM/DEM framework [13] in which a length-preserving DEM is employed. However, as Boyen [10] and, Abe, Kiltz and Okamoto (AKO) [2] pointed out, the problem of using the length-preserving DEM is that the resulting PKE scheme cannot be very efficient as the length-preserving DEM is constructed from strong PRP (Pseudo Random Permutation) [19], which is complicated and inefficient and, importantly, needs the size of plaintext messages to exceed certain length (i.e., the length of messages need to have a lower bound). Boyen [10] also provided a "miniature" encryption scheme which outputs very short ciphertext with tight security against CCA based on Gap Diffie-Hellman assumption. But this scheme does not take plaintexts of arbitrary length. Later Cash et al. [12] proposed a twin Diffie-Hellman KEM scheme which is eventually be reduced to the ordinary Computational Diffie-Hellman (CDH) assumption for its CCA-security. Compared with Fujisaki and Okamoto's [18] method, an advantage of this scheme is that it will produce a PKE scheme that does not incur further ciphertext expansion if it is combined with a length-preserving DEM. But computation cost for encryption/decryption is increased in this scheme. Abe et al.'s [2] PKE scheme proposed very recently is the first PKE scheme, which accepts arbitrary length of plaintext messages and has a ciphertext expansion of *one* group element. This scheme is very attractive to be used in resource-constrained environment.

As briefly mentioned before, the stateful public key encryption (StPKE) reduces the cost of public key encryption significantly by allowing a sender to maintain state that is reused across different encryptions. Due to the efficiency gained from maintaining state, StPKE schemes have potential to be employed in the settings where computational resources are constrained. However, ciphertext expansion of BKS's [5] efficient StPKE scheme based on DHIES [1] is actually bigger than that of the original DHIES as the underlying data (symmetric) encryption of the former scheme is required to be randomized and hence the random string *should be attached* to the symmetric part. (Informally speaking, in StPKE, since the session key (Diffie-Hellman key) becomes deterministic if the state is not changed, additional randomness should come from the symmetric encryption.) This ciphertext expansion problem in StPKE is mentioned in [5]. Our aim is to resolve this problem. We note that although there exist other construction of StPKE, but all suffer from a similar problem [4].

Our Contributions.[1] First, we show that AKO's [2] scheme can be optimized further by proposing a variant of AKO scheme, which is more efficient than the original AKO scheme but does not degrade its security. More precisely, our proposed scheme requires only two exponentiations in encryption (compared with one single exponentiation plus one multiple exponentiation in the AKO scheme) but is still proven secure against CCA in the random oracle model assuming that the Strong Diffie-Hellman (SDH) problem is hard. We also present a modified version of our scheme to provide CCA-security under the weaker CDH assumption using the technique from [12].

Second, we propose a StPKE scheme whose ciphertext expansion is smaller than the most efficient StPKE scheme based on DHIES [5]. Although our scheme is also based on the Diffie-Hellman primitive like DHIES, it changes the DHIES structure in a non-trivially way to reduce the size of ciphertext without degrading security. We show that our StPKE scheme satisfies CCA security in the random oracle model, assuming that the same SDH problem is hard. As a by-product of our StPKE construction, we also present a compact stateful identity-based encryption scheme.

Organization. In the next section, we review the notions of CCA-security for PKE and StPKE. In Section 3, we describe our new PKE schemes. Subsequently in Section 4, we describe our StPKE scheme of compact ciphertext. We compare the proposed schemes with other related ones in the literature in Section 5, which will be followed by conclusion.

2 Preliminaries

In this section, we review the formal definitions and security notions of PKE and StPKE against CCA and computational primitives we will need.

Public Key Encryption. We use the usual definitions of PKE. A PKE scheme consists of three algorithms, "KG (Key Generation)", "Enc (Encryption)" and "Dec (Decryption)". KG generates a public and private key pair and Enc encrypts a plaintext using the generated public key. Dec decrypts a given ciphertext using the private key.

Throughout this paper, we assume $\lambda \in \mathbb{Z}^+$ is a security parameter. Now, let Π_{pke} be a PKE scheme. The security against CCA for the scheme Π_{pke}, "IND-CCA", is defined as follows. Consider the following game in which an adversary $\mathcal{A}$ interacts with the challenger.

[1] Readers are referred to "Design Rationale" of Sections 3 and 4 for more detailed accounts of the technical contributions each proposed scheme.

Phase 1: The challenger runs the key generation algorithm providing λ as input to generate a public/private key pair (pk, sk). Whenever $\mathcal{A}$ makes decryption queries, each of which is denoted by ψ, the challenger runs the decryption algorithm on input ψ and gives the resulting decryption to $\mathcal{A}$.
Challenge: In this phase, $\mathcal{A}$ submits two equal-length plaintexts m_0 and m_1. On receiving this, the challenger computes a challenge ciphertext ψ^* which encrypts m_β where $\beta \in \{0,1\}$ is chosen uniformly at random. It gives ψ^* to $\mathcal{A}$.
Phase 2: Whenever $\mathcal{A}$ makes decryption queries, each of which is denoted by ψ, with a restriction that $\psi \neq \psi^*$, the challenger runs the decryption algorithm on input ψ and gives the resulting decryption to $\mathcal{A}$.
Guess: $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$.

We define $\mathcal{A}$'s advantage $\mathsf{Adv}^{ind-cca}_{\Pi_{pke},\mathcal{A}}(\lambda)$ to be $|\Pr[\beta' = \beta] - 1/2|$.

Stateful Public Key Encryption. We use the usual definition of StPKE given in [5]. A StPKE scheme consists of the following algorithms, "Setup (Setup)", "KG (Key Generation)", "PKCk (Public Key Checking)", "NwSt (New State)", "Enc (Encryption)" and "Dec (Decryption)". Setup generates a system-wide parameter and KG generates a public and private key pair. PKCk is a public key verification algorithm that checks the validity of a given public key. NwSt is an algorithm for generating fresh state. Enc encrypts a plaintext using the generated public key and the current state. Dec decrypts a given ciphertext using the private key. Note that like the original paper [5], the validity check required in this paper is very simple one, for example, checking whether some component of the given public key belongs to the underlying group, which is already exercised in practice [20].

Now, let Π_{StPKE} be a PKE scheme. The security against CCA for the scheme Π_{StPKE}, "IND-CCA", is defined as follows [5]. Consider the following game in which an adversary $\mathcal{A}$ interacts with the game.

Phase 1: The game computes system-wide parameter sp by providing the security parameter λ as input to Setup. It also generates (pk_1, sk_1) by running KG on input sp, and new state st by running NwSt on input sp. Note that (sk_1, pk_1) is the private/public key pair of the honest receiver R_1. $\mathcal{A}$ outputs public keys $pk_2, \ldots, pk_n$ of receivers $R_2, \ldots, R_n$ respectively, all of which are in the range of $\mathsf{KG}(sp)$. (Note that $\mathcal{A}$ may or may not know the private keys corresponding to the public keys pk_2, …, pk_n.) The game sends (sp, pk_1) to $\mathcal{A}$. $\mathcal{A}$ then issues a number of (but polynomially many) queries, each of which is responded by the game. The type of each query and the actions taken by the game are as follows:

- Encryption queries, each of which is denoted by (i, m) where $i \in \{1, \ldots, n\}$: On receiving this, the game computes $(\psi, st) = \mathsf{Enc}(sp, pk_i, st, m)$, where st denotes current state, and sends ψ to $\mathcal{A}$.
- Decryption queries, each of which is denoted by ψ : On receiving this, the game computes $\mathsf{Dec}(sp, sk_1, \psi)$ and sends the resulting decryption to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits a challenge query (m_0, m_1) such that $|m_0| = |m_1|$: On receiving this, the game picks $\beta \in \{0,1\}$ at random, computes $(\psi^*, st) = \mathsf{Enc}(sp, pk_1, st, m_\beta)$, where st denotes current state, and sends ψ^* to $\mathcal{A}$.
Phase 2: $\mathcal{A}$ continues to issue encryption queries (i, m) and decryption queries such that $\psi \neq \psi^*$.
Guess: $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$.

We define $\mathcal{A}$'s advantage $\mathsf{Adv}^{ind-cca}_{\Pi_{StPKE},\mathcal{A}}(\lambda)$ to be $|\Pr[\beta' = \beta] - 1/2|$.

We remark that the above security notion can be considered in the *KSK* (Known Secret Key) or the *USK* (Unknown Secret Key) models [5]. In the KSK model, it is assumed that $\mathcal{A}$ possesses the corresponding private (secret) keys $sk_2 \ldots, sk_n$ of the public keys it outputs in Phase 2 of the attack game. On the other hand, in the USK model, this assumption is not needed[2].

Computational Primitives. Finally we review the computational primitives, which will be used in the paper.

Let $\mathbb{G}$ be a multiplicative group of prime order $p \approx 2^{2\lambda}$, where λ is a security parameter. The Strong Diffie-Hellman (SDH) problem is defined as follows. An adversary $\mathcal{B}$, given $(g, g^a, g^b) \in \mathbb{G}^3$, is to compute g^{ab} with the help of a *restricted* Decisional Diffie-Hellman oracle $\mathsf{DDH}_{g,g^a}(\cdot,\cdot)$, which on input $(g^{\bar{b}}, g^{\bar{c}})$ outputs 1 if and only if $a\bar{b} = \bar{c} \bmod p$. We define $\mathcal{B}$'s advantage on solving the SDH problem to be $\mathsf{Adv}^{sdh}_{\mathcal{B}}(\lambda)$.

We can also define GDH assumption [22], a similar but stronger assumption than the above SDH assumption: An adversary $\mathcal{B}$, given $(g, g^a, g^b) \in \mathbb{G}^3$, is to compute g^{ab} with the help of a *full* Decisional Diffie-Hellman oracle $\mathsf{DDH}_g(\cdot,\cdot,\cdot)$, which on input $(g^{\bar{a}}, g^{\bar{b}}, g^{\bar{c}})$ outputs 1 if and only if $\bar{a}\bar{b} = \bar{c} \bmod p$. We define $\mathcal{B}$'s advantage on solving the GDH problem to be $\mathsf{Adv}^{sdh}_{\mathcal{B}}(\lambda)$.

3 Our Compact Public Key Encryption Scheme

In this section, we present our compact PKE scheme, which we denote by "Π_{SDH+}".

Description. As follows is a description of Π_{SDH+}.

- $\mathsf{KG}(\lambda)$: Generate group parameters $(p, g, \mathbb{G})$, where $p \approx 2^{2\lambda}$ and $g \in \mathbb{G}$ is a generator. Choose hash functions $\mathsf{G} : \mathbb{G} \to \{0,1\}^{\lambda_k}$, where $\lambda_k = 2\lambda$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p^*$. Then pick $x \in \mathbb{Z}_p^*$ uniformly at random and compute $y = g^x$. Return public key $pk = (p, g, y, \mathsf{G}, \mathsf{H})$ and private key $sk = (pk, x)$.
- $\mathsf{Enc}(pk, m)$: Pick $r \in \mathbb{Z}_p^*$ at random and compute $\kappa = g^r$, $k = \mathsf{G}(\kappa)$, $v = k \oplus m$, and $\tau = \mathsf{H}(v)$, where $m \in \{0,1\}^l$ for $l \geq 1$ is a plaintext. Compute $y^{\tau r}$ and then $u = \kappa y^{\tau r} = g^r y^{\tau r}$. Return a ciphertext $\psi = (u, v)$.

[2] In other words, in the KSK model, it may be the case that the trusted third party (like CA) is required to perform a proof of knowledge protocol to confirm whether users have corresponding private keys of their public keys while in the USK model, PKCk should be run to check whether the public keys are valid. Readers are referred to [5].

– $\mathsf{Dec}(sk, \psi)$: Upon receiving $\psi = (u, v)$, compute

$$\kappa = u^{\frac{1}{1+\tau x}},$$

where $\tau = \mathsf{H}(v)$, and return $m = v \oplus k$, where $k = \mathsf{G}(\kappa)$.

In the above construction, when $l > \lambda_k$, we expand k to a bit-string k' using a pseudo-random generator and use k' as an xor-based one-time pad. Whereas, when $l \leq \lambda_k$ ($l = |m|$), we use the first l-bit of k as an xor-based one-time pad. Hence the symmetric encryption part v is "length-preserving [2]".

Design Rationale. Notice that the efficiency gain in our scheme comes from the computation of u. By "moving" $\tau = \mathsf{H}(v)$ from the exponent of g to y, we could avoid g^r is computed *again* through the multi-exponentiation. In other words, we can compute u by performing *exactly two exponentiations*, separately computing g^r and $y^{\tau r}$ and just multiplying them. However, in the original AKO [2] scheme, u is computed as $(g^\tau y)^r$ while using g^r as (encapsulation) key, so two values g^r and $g^{\tau r}$ should be computed as a sequence of exponentiations and $g^{\tau r}$ and y^r are multiplied together, or g^r and $g^{\tau r} y^r$ should be computed separately[3]. In this case, $g^{\tau r} y^r (= (g^\tau y)^r)$ should be computed via multi-exponentiation.) Hence, we could avoid a multi-exponentiation or a sequence of two exponentiations operations, whose cost do not exceed two single exponentiations but more than one exponentiation [8]. (We note that depending on algorithms that can be employed [16], multi-exponentiation can be conducted very efficiently. However, our argument here is that one can in fact avoid multi-exponentiation completely when constructing compact PKE schemes, which is beneficial for lightweight cryptography as some fast algorithms for multi-exponentiation might require much more memory, for example, Pippenger's algorithm [8], and/or more involved implementations.)

Security. Somewhat interestingly, we show that our modification does not degrade security at all. Below, we prove that the scheme Π_{SDH+} is IND-CCA secure in the random oracle model, relative to the SDH problem:

Theorem 1. *For an IND-CCA adversary $\mathcal{A}$ attacking Π_{SDH+} with running time t, there exists an adversary $\mathcal{B}$ solving the SDH problem with running time $t + \mathsf{O}(q_\mathsf{D} q_\mathsf{G})$ such that*

$$\mathsf{Adv}^{ind-cca}_{\Pi_{SDH+},\mathcal{A}}(\lambda) \leq 2\lambda_k \mathsf{Adv}^{sdh}_{\mathcal{B}}(\lambda) + \frac{q_\mathsf{G}}{p} + \frac{q_\mathsf{G} + 2q_\mathsf{H}}{2^{\lambda_k}},$$

where q_G and q_H denote the number of queries to the random oracles G and H respectively; q_D denotes the number of queries to the decryption oracle.

Sketch of the proof. The main idea of the proof follows that of the original AKO scheme [2]. However, due to the change of encryption process, we have

[3] The latter method was suggested in [2]. The former method takes similar amounts of computations [8].

to simulate the public key and challenge ciphertext differently. Basically we *set* $g = z^a$, $y = (zg^{-1})^{1/\tau^*}$, $u^* = z^b$, $v^* = k^* \oplus m_\beta$, and $k^* = \mathsf{G}(\kappa^*)$ for random $k^* \in \{0,1\}^{\lambda_k}$ and $\beta \in \{0,1\}$, where z^a and z^b are the SDH instance. (We do not know the value κ^* yet.) The SDH problem will be solved when the IND-CCA adversary queries G on $\kappa^* = z^{ab}$. The validity of each ciphertext (u, v) is checked as follows: For an entry (κ, k) in the query-answer list for G, check whether $\mathsf{DDH}_{z,g}((u/\kappa^{(1-\frac{\tau}{\tau^*})})^{\frac{\tau^*}{\tau}}, \kappa) = 1$, where $\tau^* = \mathsf{H}(v^*)$ and $\tau = \mathsf{H}(v)$. – Due to the lack of space, the complete proof will be given in the full version of this paper.

CDH Variant. Using the twinning technique from [12], our scheme can easily be modified to provide CCA-security with even the weaker CDH (Computational Diffie-Hellman) assumption. (Note that the CDH assumption can be defined not giving the SDH adversary access to the strong DDH oracle.) As follows is the description of the new scheme, which we denote by "Π_{CDH+}".

– $\mathsf{KG}(\lambda)$: Generate group parameters $(p, g, \mathbb{G})$, where $p \approx 2^{2\lambda}$ and $g \in \mathbb{G}$ is a generator. Choose hash functions $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{\lambda_k}$, where $\lambda_k = 2\lambda$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p^*$. Then pick $x \in \mathbb{Z}_p$ and $z \in \mathbb{Z}_p$ uniformly at random and compute $y = g^x$ and $\tilde{g} = g^z$. Return public key $pk = (p, g, y, \mathsf{G}, \mathsf{H})$ and private key $sk = (pk, x)$.
– $\mathsf{Enc}(pk, m)$: Pick $r \in \mathbb{Z}_q^*$ at random and subsequently compute $\kappa = g^r$, $\tilde{\kappa} = \tilde{g}^r$, $k = \mathsf{G}(\kappa, \tilde{\kappa})$, $v = k \oplus m$, and $\tau = \mathsf{H}(v)$. Compute $y^{r\tau}$ and then $u = \kappa y^{r\tau} = g^r y^{r\tau}$. Return a ciphertext $\psi = (u, v)$.
– $\mathsf{Dec}(sk, \psi)$: Upon receiving $\psi = (u, v)$, compute

$$\kappa = u^{\frac{1}{1+\tau x}} \text{ and } \tilde{\kappa} = u^{\frac{z}{1+\tau x}},$$

where $\tau = \mathsf{H}(v)$, and return $m = v \oplus k$, where $k = \mathsf{G}(\kappa, \tilde{\kappa})$.

We proved that the scheme Π_{CDH+} is IND-CCA secure in the random oracle model, relative to the CDH problem. (The proof will be given in the full version of the paper.)

4 Our Compact Stateful Public Key Encryption Scheme

In this section, we present our compact StPKE scheme.
Description. As follows is a description of our StPKE scheme, which we denote by "Π_{stDH+}".

– $\mathsf{Setup}(\lambda)$: Generate a multiplicative group $\mathbb{G}$ of prime order $p \approx 2^{2\lambda}$. Choose hash functions $\mathsf{G} : \mathbb{G} \times \{0,1\}^\lambda \to \{0,1\}^{\lambda_k}$, where $\lambda_k = 2\lambda$, and $\mathsf{H} : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \times \mathbb{G} \to \{0,1\}^\lambda$. Return system parameter $sp = (\mathbb{G}, p, g, \mathsf{G}, \mathsf{H})$.
– $\mathsf{KG}(sp)$: Pick $x \in \mathbb{Z}_p^*$ uniformly at random and compute $y = g^x$. Return public key $pk = y$ and private key $sk = x$.
– $\mathsf{PKCk}(sp, pk)$: Check if $y \in \mathbb{G}$. If it is, return 1, and 0 otherwise.

– $\mathsf{NwSt}(sp)$: Pick $r \in \mathbb{Z}_p^*$ uniformly at random and compute $u = g^r$. Return state $st = (r, u)$.
– $\mathsf{Enc}(sp, pk, st, m)$: Let $st = (r, u)$. Compute $\kappa = y^r$ and pick $s \in \{0,1\}^\lambda$ uniformly at random. Then subsequently compute $k = \mathsf{G}(u, s)$, $v = k \oplus m$ for plaintext $m \in \{0,1\}^l$ $(l \geq 1)$ and $w = \mathsf{H}(y, u, v, \kappa) \oplus s$. Return ciphertext $\psi = (u, v, w)$.
– $\mathsf{Dec}(sp, sk, \psi)$: Upon receiving $\psi = (u, v, w)$, compute $\kappa = u^x$ and $s = w \oplus \mathsf{H}(y, u, v, \kappa)$. Then return $m = v \oplus k$, where $k = \mathsf{G}(u, s)$.

Like the previous PKE scheme, the symmetric encryption part v is length-preserving: When $l > \lambda_k$, k is expanded to a bit-string k' through a pseudo-random generator k' will be used as an xor-based one-time pad. When $l \leq \lambda_k$ $(l = |m|)$, the first l-bit of k will be used as an xor-based one-time pad.

Design Rationale. For the sake of convenience of discussion, we briefly describe the DHIES-based StPKE scheme from [5], which we denote by Π_{stDH}: In this scheme, a plaintext m is encrypted to $(g^r, \mathsf{E}_k(m))$, $y = g^x$ is public key and $k = \mathsf{H}(y, g^r, y^r)$ is a session key. Here the random r and g^r are served as state, so g^r does not need to be computed each time a new plaintext is encrypted. (y^r can also be a part of state to boost efficiency as suggested in [5] .) Since the session key k essentially becomes deterministic when the same state is reused, the symmetric encryption E needs to be strengthened to provide enough randomness. More precisely, E should not only provide a guarantee that access to decryption oracle should not give any useful information to adversary but also make sure that there is sufficient randomness to attain indistinguishability. Consequently, the symmetric encryption in Π_{stDH} needs to be fully IND-CCA secure unlike the case of DHIES whose symmetric encryption needs to be only one-time CCA-secure [5,13], wihch is weaker. However, cost for strengthening the symmetric encryption is the expansion of the ciphertext [5].

In contrast, we *always randomize* the session key (i.e., k in our description) in our scheme even if the same state is reused. Importantly, the random string used to randomize the session key is xor-ed with the "authentication tag (the output produced by the hash function H)" to reduce the size of the whole ciphertext and to prevent it from revealing. (Unlike the previous scheme [5], the random string should not be revealed.)

Finally, some readers might wonder why one cannot use AKO's [2] public key encryption as a basic primitive to construct compact StPKE. The reason is that the specific algebraic property of the AKO scheme does not allow us to use the same state again as stateful version of the AKO scheme is totally insecure: Suppose that r used to create a ciphertext (u, v) such that $u = (g^{\mathsf{H}(v)}y)^r$ where $y = g^x$ and $v = \mathsf{E}_k(m)$ where $k = \mathsf{G}(g^r)$ is reused. Then one can get (u', v') such that $u' = (g^{\mathsf{H}(v')}y)^r$ and $v' = \mathsf{E}_k(m')$. But, in this case the session key material g^r can easily be extracted by computing $(u/u')^{1/(\mathsf{H}(v)-\mathsf{H}(v'))}$!

Security. We prove that in the random oracle and the USK (Unknown Secret Key) model, the scheme Π_{stDH+} is IND-CCA secure assuming the Gap Diffie-Hellman (GDH) problem is hard:

Theorem 2. *For an IND-CCA adversary $\mathcal{A}$ attacking Π_{stDH+} with running time t, there exists an adversary $\mathcal{B}$ solving the GDH problem with running time $t + \mathsf{O}(q_{\mathsf{D}}(q_{\mathsf{G}} + q_{\mathsf{H}}))$ such that*

$$\mathsf{Adv}^{ind-cca}_{\Pi_{stDH+},\mathcal{A}}(\lambda) \leq \frac{q_{\mathsf{G}}}{2^{2\lambda}} + \frac{q_{\mathsf{H}}}{2^{\lambda}} + \mathsf{Adv}^{gdh}_{\mathcal{B}}(\lambda),$$

where λ denotes the security parameter; q_{G} and q_{H} denote the number of queries to the random oracles G and H respectively; q_{D} denotes the number of queries to the decryption oracle.

Sketch of the proof. Assuming that (g^a, g^b) is an instance of the GDH problem, we set $u^* = g^a$ and $y_1 = g^b$, where y_1 is an honest receiver's public key and u^* is the current state (which reused throughout the game). By using the IND-CCA adversary $\mathcal{A}$, we try to find g^{ab}, which is set as the session key κ^* in the challenge ciphertext. The proof basically uses the programability of the random oracles to answer $\mathcal{A}$'s queries to the oracles G and H. In particular, upon receiving queries to H, we use the DDH oracle to "filter" Diffie-Hellman tuples, i.e., tuples of the form $(g^{\bar{a}}, g^{\bar{b}}, g^{\bar{a}\bar{b}})$. This filtering procedure will play an important role in answering queries to the decryption oracle (That is, we filter out implicitly "right ciphertexts".) Note that for each decryption query, we need to deal with the query-answer lists for *both* G and H as the ciphertext components v and w are inter-related through G and H. Note also that unlike the normal PKE, we need to answer "encryption queries" under the same state, which can be done using the programability of G and H. – The full proof is given in Appendix A.

Extension to Stateful Identity-Based Encryption. The technique used in our StPKE scheme Π_{stDH+} can easily be extended to construct a compact stateful identity-based encryption (StIBE) scheme, whose concept was first introduced by Phong, Matsuoka and Ogata [23]. Compared with their StIBE scheme based on Boneh and Franklin's IBE [9], our scheme will provide shorter ciphertext in the same way as our StPKE scheme Π_{stDH+} does. (Phong et al.'s scheme has a similar structure of the scheme Π_{stDH} in [5], so the same degree of ciphertext expansion will be incurred when the "encrypt-then-mac" construction for IND-CCA secure symmetric encryption [6] is used.)
Description. As follows is a description of our StIBE scheme, which we denote by "Π_{stBDH+}".

– $\mathsf{Setup}(\lambda)$: Generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order $p \approx 2^{2\lambda}$ and a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Pick a generator $g \in \mathbb{G}_1$. Also, pick $x \in \mathbb{Z}_p^*$ uniformly at random and compute $y = g^x$. Then, choose a length-preserving symmetric encryption $\Pi_{sym} = (\mathsf{E}, \mathsf{D})$, and hash functions $\mathsf{G} : \mathbb{G}_1 \times \{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda_k}$, where $\lambda_k = 2\lambda$, $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathsf{H}_2 : \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^* \times \mathbb{G}_2 \rightarrow \{0,1\}^{\lambda}$. Return system parameters $params = (\mathbb{G}, p, g, \hat{e}, \Pi_{sym}, \mathsf{G}, \mathsf{H}, y)$ and master key $mk = x$.
– $\mathsf{Extract}(sp, mk, \mathtt{ID})$: On receiving $\mathtt{ID}$, compute $sk_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})^x$. Return $sk_{\mathtt{ID}}$ as a private key associated with $\mathtt{ID}$.
– $\mathsf{NwSt}(sp)$: Pick $r \in \mathbb{Z}_p^*$ uniformly at random and compute $u = g^r$ and $t = y^r$. Return state $st = (u, t)$.

- $\mathsf{Enc}(sp, \mathtt{ID}, st, m)$: Let $st = (u, t)$. Compute $\phi_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $\kappa = \hat{e}(\phi_{\mathtt{ID}}, t)$. Pick $s \in \{0,1\}^{\lambda}$ uniformly at random. Then subsequently compute $k = \mathsf{G}(u, s)$, $v = \mathsf{E}_k(m)$ for plaintext $m \in \{0,1\}^l$ and $w = \mathsf{H}_2(\phi_{\mathtt{ID}}, u, v, \kappa) \oplus s$. Return a ciphertext $\psi = (u, v, w)$.
- $\mathsf{Dec}(sp, sk_{\mathtt{ID}}, \psi)$: Upon receiving $\psi = (u, v, w)$, compute $\kappa = \hat{e}(sk_{\mathtt{ID}}, u)$ and $s = w \oplus H_2(\phi_{\mathtt{ID}}, u, v, \kappa)$, where $\phi_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$. Then return $m = \mathsf{D}_k(v)$, where $k = \mathsf{G}(u, s)$.

The security of the above scheme Π_{stBDH+} against CCA (as defined in [23]) can be proven as follows. A basic idea of the proof is to first reduce the CCA-security of the above scheme in the selective identity model, where a challenge identity $\mathtt{ID}^*$ is output by adversary beginning of the game, to the security of the normal StPKE (defined in Section 2) in which a session key can be computed as $\kappa = \hat{e}(\phi, y^r)$ where ϕ is now a fixed point rather than a hash output of some identity. By using the selective identity model, one can handle the encryption queries, each of which is denoted by $(\mathtt{ID}, m)$ more effectively. Using a similar technique used in the proof of Π_{stDH+}, it can be shown that the derived StPKE scheme is IND-CCA secure assuming that the Bilinear Diffie-Hellman (BDH) [9] is hard (in the random oracle model). The final step is to convert the selective identity secure version of the scheme into the fully secure version using the result of [23]. (The detailed proof will be provided in the full version of this paper.)

5 Comparisons

In Table 1, we summarize the restriction on the size of plaintext, encryption and decryption cost, ciphertext expansion (bandwidth), and computational assumptions for proving CCA-security of our schemes and the related schemes in the literature.

Table 1. Comparison of our PKE schemes with other schemes. ("λ" denotes a security parameter. e denotes "exponentiation". Note that we assume that one multi-exponentiation equals to 1.5 exponentiation following the general convention [2,8]. "mac" and "hash" denote MAC-ing and hashing of a long message, respectively. "sprp" denotes cost for a strong PRP (pseudo random permutation) computation.

Scheme	Plaintext size	Encryption cost	Decryption cost	Ciphertext expansion	Assumption
DHIES [1]	any	$2e$+mac	$1e$+mac	$\lvert\mathbb{G}\rvert + \lvert\text{mac}\rvert$	SDH
KM [21]	$\lvert m\rvert > \lambda$	$2e$+sprp	$1e$+sprp	$\lvert\mathbb{G}\rvert$	SDH
Twin KM [12]	$\lvert m\rvert > \lambda$	$3e$+sprp	$1.5e$+sprp	$\lvert\mathbb{G}\rvert$	CDH
Boyen [10]	$\lvert m\rvert > 2\lambda$	$2.5e$+hash	$1.5e$+hash	$\lvert\mathbb{G}\rvert$	GDH
AKO [2]	any	$2.5e$+hash	$1e$+hash	$\lvert\mathbb{G}\rvert$	SDH
Twin AKO [2]	any	$3.5e$+hash	$2e$+hash	$\lvert\mathbb{G}\rvert$	CDH
Π_{SDH+} (Ours)	any	$2e$+hash	$1e$+hash	$\lvert\mathbb{G}\rvert$	SDH
Π_{CDH+} (Ours)	any	$3e$+hash	$2e$+hash	$\lvert\mathbb{G}\rvert$	CDH

Like the schemes in [2], our schemes accept arbitrary length of plaintexts, while the KM scheme [21], the Twin KM scheme [12], and Boyen's scheme [10] need the length of plaintext to exceed certain length. Note that the KM and Twin KM scheme are in fact based on the KEM/DEM approach in which a length-preserving DEM is employed. However, this length-preserving DEM will bring security problem when a very short message is encrypted. Hence, as noted in [2], the plaintext length should at least be λ, a security parameter. In terms of encryption and decryption cost, our schemes are highly efficient, more efficient than the AKO and Twin AKO schemes, Boyen's scheme[4], and comparable to DHIES [1] and ElGamal [17]. Our schemes are also more efficient than KM and Twin KM schemes which need complicated strong PRPs as (length-preserving) DEM. Summing up, our schemes are most efficient and compact and accept arbitrary length of plaintexts.

In Table 2 , we summarize the restriction on the size of plaintext, encryption cost in stateful mode when $\kappa = y^r$ is a part of state, and decryption cost, ciphertext expansion (bandwidth), and computational assumptions for proving CCA-security of our scheme Π_{stDH+} and the DHIES-based StPKE scheme in [5], denoted by Π_{stDH}. As can be seen from this table, our scheme reduces the ciphertext expansion by up to the size of the random string used in the scheme Π_{stDH} [5]. If we use the estimation from [5], the size of ciphertext expansion of Π_{stDH} is 4λ while that of our scheme Π_{stDH+} is 3λ. Assuming that $\lambda = 128$, one can save 16 bytes per each ciphertext, which could be significant in resource-constrained computing environment.

Table 2. Comparison of our StPKE scheme with the DHIES-based StPKE scheme in [5]. ("λ" denotes a security parameter. e denotes "exponentiation". "mac" and "hash" denote MAC-ing and hashing, respectively. "rs" denotes random string. "GDH" denotes Gap Diffie-Hellman assumption.) Note that following the convention of [5], we omit the initiation exponentiation (one "e" operation) in the encryption cost.

Scheme	Plaintext size	Encryption cost (stateful)	Decryption cost	Ciphertext expansion	Assumption
Π_{stDH} [5]	any	hash+mac	1e+mac	$\lvert\mathbb{G}\rvert + \lvert\text{mac}\rvert + \lvert\text{rs}\rvert$	GDH
Π_{stDH+} (Ours)	any	2hash	1e+hash	$\lvert\mathbb{G}\rvert + \lvert\text{hash}\rvert$	GDH

6 Conclusion

We presented a compact PKE scheme which is more efficient than the previous one in the literature [2] and an efficient StPKE scheme which outputs short ciphertext. We also presented extensions of these schemes. We showed that our

[4] Although [2] states that the encryption of Boyen's scheme needs three exponentiations, we correct it to 2.5 exponentiations since, as claimed in [10], computation cost for a sequence of exponentiations such as g^r and g^{rf} can be reduced by reusing g^r for the computation of g^{fr}.

schemes satisfy CCA security in the random oracle model under the well-known assumptions, SDH, CDH and GDH respectively. Further reducing the size of ciphertexts of public key encryption and related schemes for resource-constrained devices will be interesting future research. (However, we cautiously state that the PKE scheme in the random oracle model is almost optimized due to AKO's scheme [2] and its further optimization presented in this paper.)

Acknowledgement

The authors are grateful to the anonymous referees of CT-RSA 2011 for their helpful comments. This work is supported by the EMA funded project SecSG-EPD090005RFP(D), Singapore.

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. Abe, M., Kiltz, E., Okamoto, T.: Compact CCA-secure encryption for messages of arbitrary length. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 377–392. Springer, Heidelberg (2009)
3. Baek, J., Tan, H., Zhou, J., Wong, J.: Realizing Stateful Public Key Encryption in Wireless Sensor Network. In: Proc. of the IFIP-SEC 2008, pp. 95–108. Springer, Heidelberg (2008)
4. Baek, J., Zhou, J., Bao, F.: Generic constructions of stateful public key encryption and their applications. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 75–93. Springer, Heidelberg (2008)
5. Bellare, M., Kohno, T., Shoup, V.: Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation. In: ACM-CCS 2006, pp. 380–389. ACM Press, New York (2006)
6. 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)
7. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM-CCS 1993, pp. 62–73. ACM, New York (1993)
8. Bernstein, D.J.: Pippenger's Exponentiation Algorithm (2002) (preprint), http://cr.yp.to
9. Boneh, D., Franklin, M.: Identity Based Encryption from the Weil Pairing. SIAM Journal of Computing 32(3), 586–615 (2003); Entexded abstract in Crypto 2001, LNCS, vol. 2139, pp. 213–229, Springer-Verlag (2001)
10. Boyen, X.: Miniature CCA2 PK Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 485–501. Springer, Heidelberg (2007)
11. Boyen, X.: A Tapestry of Identity-Based Encryption: Practical Frameworks Compared. International Journal of Applied Cryptography 1(1), 3–21 (2008)
12. 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); Full version available on Cryptology ePrint Archive: Report 2008/067

13. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM Journal of Computing 33, 167–226 (2003)
14. Mica2Dot Wireless Sensor Mote, MEMSIC Inc., http://www.memsic.com
15. MicaZ Wireless Sensor Network Platform, Crossbow Technology, http://www.xbow.com/
16. Müller, B.: Algorithms for Multi-Exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
17. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Trans. Information Theory 31, 469–472 (1985)
18. 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)
19. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
20. ISO CD 18033-2. Encryption Algorithms Part 2: Asymmetric Ciphers (2004)
21. Kurosawa, K., Matsuo, T.: How to Remove MAC from DHIES. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 236–247. Springer, Heidelberg (2004)
22. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
23. Phong, L., Matsuoka, H., Ogata, W.: Stateful Identity-Based Encryption Scheme: Faster Encryption and Decryption. In: AsiaCCS 2008, pp. 381–388. ACM, New York (2008)
24. Shirase, M., Miyazaki, Y., Takagi, T., Han, D., Choi, D.: Efficient Implementation of Pairing-Based Cryptography on a Sensor Node. IEICE Transactions 92-D(5), 909–917 (2009)
25. Szczechowiak, P., Kargl, A., Scott, M., Collier, M.: On the Application of Pairing Based Cryptography to Wireless Sensor Networks. In: ACM-WISEC 2009, pp. 1–12. ACM, New York (2009)
26. Wander, A., Gura, N., Eberle, H., Gupta, V., Shantz, S.: Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. In: IEEE International Conference on Pervasive Computing and Communication 2005 (PerCom 2005), pp. 324–328. IEEE Computer Society, Los Alamitos (2005)
27. Watro, R., Kong, D., Fen Cuti, S., Gardiner, C., Lynn, C., Kruus, P.: TinyPK: securing sensor networks with public key technology. In: ACM Workshop on Security of ad hoc and Sensor Networks 2004 (SASN 2004), pp. 59–64. ACM Press, New York (2004)

A Proof of Theorem 2

Proof. The proof consists of the following sequence of games.

Game G_0: This game is identical to the IND-CCA game played by an attacker $\mathcal{A}$ against the scheme Π_{StPKE}. We repeat this game to clean up the notations. Let sp be a system parameter. Let $sk_1 = x_1$ and $pk_1 = y_1 (= g^{x_1})$ be private and public keys of the honest receiver respectively. Let $pk_2, \ldots, pk_n$ be the public keys output by $\mathcal{A}$. Let $st = (r^*, u^*)$, where $u^* = g^{r^*}$ for random $r^* \in \mathbb{Z}_p^*$, be the sender's state, fixed throughout each game. We denote a challenge ciphertext by $\psi^* = (u^*, v^*, w^*)$, where $v^* = k^* \oplus m_\beta$ for random $\beta \in \{0,1\}$ such that $k^* = \mathsf{G}(u^*, s^*)$, and $w^* = \sigma^* \oplus s^*$ where $s^* \in \{0,1\}^\lambda$ is chosen at random, $\sigma^* = \mathsf{H}(y_1, u^*, v^*, \kappa^*)$ and $\kappa^* = (u^*)^{x_1}$.

We denote by S_0 the event $\beta' = \beta$, where β' is a bit output by $\mathcal{A}$ at the end of the game. (We use a similar notation $S_1, S_2, \ldots$ for all modified games $G_1, G_2, \ldots$ respectively). Since G_0 is the same as the real attack game of IND-CCA, we have

$$\mathsf{Adv}^{\text{ind-cca}}_{\mathcal{A},\Pi_{stDH+}}(\lambda) = \left|\Pr[S_0] - \frac{1}{2}\right|.$$

Game G_1: Starting from this game, we use query-answer lists, $\mathsf{EncList}$ and GList for the encryption oracle and the random oracle G respectively. We also use two types of lists $\mathsf{H_0List}$ and $\mathsf{H_1List}$ for the random oracle H. Basically the $\mathsf{H_1List}$ will record queries which consist of group elements which have right Diffie-Hellman distribution and answers for the queries. $\mathsf{H_0List}$ will record the list of queries that do not have right Diffie-Hellman distribution. Note that in these lists, queries and answers are divided by the symbol "|".

In this game, if $\mathcal{A}$ submits public keys, the game checks the validity of each public key via running PKCk. We denote the public keys that have passed the validity test by $y_2, \ldots, y_n$. Then, add $[(u^*, s^*)|k^*]$ to GList and $[(y_1, u^*, v^*, \kappa^*)|\sigma^*]$ to $\mathsf{H_1List}$. Whenever $\mathcal{A}$ submits encryption queries with regards to the public keys $y_1, \ldots y_n$, each of which is denoted by "(i, m)" where $i \in [1, n]$, the game runs the following encryption oracle simulator.

```
$\mathsf{EncSim}(i, m)$
  Pick $s \in \{0,1\}^\lambda$ and $k \in \{0,1\}^{\lambda_K}$ uniformly at random.
  Set $k \overset{\text{def}}{=} \mathsf{G}(u^*, s)$ and add $[(u^*, s)|k]$ to $\mathsf{GList}$.
  Compute $v = k \oplus m$.
  Pick $\sigma \in \{0,1\}^\lambda$ uniformly at random.
  Compute $w = \sigma \oplus s$.
  Set $\sigma \overset{\text{def}}{=} \mathsf{H}(y_i, u^*, v, ?)$ and add $[(y_i, u^*, v, ?)|\sigma]$ to $\mathsf{H_1List}$.
  Return $(u^*, v, w)$.
```

Notice in the above simulation that, since s is chosen freshly at random, there can be more than one s and hence v that correspond to the same plaintext m. Since the above encryption oracle simulator perfectly simulates Game G_0, we have

$$\Pr[S_1] = \Pr[S_0].$$

Game G_2: In this game, if $\mathcal{A}$ queries some (u^*, s) such that $\mathsf{G}(u^*, s) = \mathsf{G}(u^*, s^*) = k^*$, or some (y_1, u^*, v, κ) such that $\mathsf{H}(y_1, u^*, v, \kappa) = \mathsf{H}(y_1, u^*, v^*, \kappa^*) = \sigma^*$, the simulator aborts the game. Note that games G_0 and G_1 are equivalent until such event happens. Since G and H are assumed as random oracles whose ranges are $\{0,1\}^{\lambda_k}$ and $\{0,1\}^{\lambda_k}$ respectively, and there are up to q_G and q_H such queries, we have

$$\left|\Pr[S_2] - \Pr[S_1]\right| \le \frac{q_\mathsf{G}}{2^{\lambda_k}} + \frac{q_\mathsf{H}}{2^\lambda},$$

where $\lambda_k = 2\lambda$

Game G_3: In this game, whenever $\mathcal{A}$ submits queries to the random oracle G and H the game runs the following simulators GSim and HSim respectively.

```
$\mathsf{GSim}(u, s)$

  If $[(u, s)|k]$ exists in $\mathsf{GList}$, return $k$.
  Pick $k \in \{0,1\}^{\lambda_k}$ uniformly at random, set $k \overset{\text{def}}{=} \mathsf{G}(u, s)$ and return $k$.
  Add $[(u, s)|k]$ to $\mathsf{GList}$.

$\mathsf{HSim}(y, u, v, \kappa)$

  If $[(y, u, v, \kappa)|\sigma]$ exists in $\mathsf{H_1List}$ or $\mathsf{H_0List}$, return $\sigma$.
  If $\mathsf{DDH}_g(y, u, \kappa) = 1$ do the following:
    If $u = u^*$, $v = v^*$ and $y = y_1$, extract $[(y_1, u^*, v^*, \kappa^*)|\sigma^*] \in \mathsf{H_1List}$ and return $\sigma^*$.
    Otherwise, do the following:
      If $[(y, u, v, ?)|\sigma]$ or $[(y, u, v, \kappa)|\sigma]$ in $\mathsf{H_1List}$, return $\sigma$. Then replace the symbol "?" with $\kappa$.
      Otherwise, pick $\sigma \in \{0,1\}^{\lambda}$ uniformly at random, return $\sigma$ and add $[(y, u, v, \kappa)|\sigma]$ to $\mathsf{H_1List}$.
  If $\mathsf{DDH}_g(y, u, \kappa) \neq 1$ do the following:
    If $[(y, u, v, \kappa)|\sigma]$ exists in $\mathsf{H_0List}$, return $\sigma$. Otherwise, pick $\sigma \in \{0,1\}^{\lambda}$ uniformly at random, return $\sigma$ and add $[(y, u, v, \kappa)|\sigma]$ to $\mathsf{H_0List}$.
```

Using the random oracle simulators described above, this game answers $\mathcal{A}$'s each decryption query $\psi = (u, v, w)$ as follows.

```
$\mathsf{DecSim}(u, v, w)$

  If $u = u^*$ and $v = v^*$ (and hence $w \neq w^*$), do the following:
    Extract $[(y_1, u^*, v^*, \kappa^*)|\sigma^*]$ from $\mathsf{H_1List}$, compute $s = w \oplus \sigma^*$, search $[(u^*, s)|k] \in \mathsf{GList}$. If such entry exists, return $m = v \oplus k$. Otherwise, run $\mathsf{GSim}$ on input $(u, s)$ to get $k$ and return $m = v \oplus k$.
  Otherwise, do the following:
    If $[(y_1, u, v, \kappa)|\sigma]$ or $[(y_1, u, v, ?)|\sigma]$ exists in $\mathsf{H_1List}$, compute $s = w \oplus \sigma$ and search $[(u, s), k] \in \mathsf{GList}$. If such an entry exists in $\mathsf{GList}$, return $m = v \oplus k$. Otherwise (such a pair does not exist in $\mathsf{GList}$), run $\mathsf{GSim}$ on input $(u, s)$ to get $k$ and return $m = v \oplus k$.
    Otherwise, that is, if $[(y_1, u, v, \kappa)|\sigma]$ exists in $\mathsf{H_0List}$ or such an entry simply does not exist, pick $\sigma' \in \{0,1\}^{\lambda}$ uniformly at random and add $[(y_1, u, v, ?)|\sigma']$ to $\mathsf{H_1List}$. (That is, if $[(y_1, u, v, \kappa)|\sigma]$ exists in $\mathsf{H_0List}$, the value $\sigma$ is ignored.) Then compute $s = w \oplus \sigma'$ and search $[(u, s), k] \in \mathsf{GList}$. If such an entry exists in $\mathsf{GList}$, return $m = v \oplus k$. Otherwise (such a pair does not exist in $\mathsf{GList}$), run $\mathsf{GSim}$ on input $(u, s)$ to get $k$ and return $m = v \oplus k$.
```

Note that the simulators presented above perfectly simulates the random oracles, challenge ciphertexts and encryption oracle. Hence we have

$$\Pr[S_2] = \Pr[S_1].$$

Game G_4: In this game, we do not pick up σ^* and set $\sigma^* = \mathsf{H}(y_1, u^*, v^*, \kappa^*)$ any longer. Now, let F be an event that $\mathcal{A}$ queries $(y_1, u^*, v^*, \kappa^*)$ (which satisfies $\mathsf{DDH}_g(y_1, u^*, \kappa^*) = 1$) to the random oracle H. Notice that unless F occurs, this game and the previous game proceed identically. Thus we get

$$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F].$$

Note that one can now construct an attacker $\mathcal{B}$ that can solve the GDH problem using the StPKE attacker $\mathcal{A}$. – $\mathcal{B}$, given (g, g^a, g^b) and access to the DDH oracle, sets $y_1 = g^b$ and $u^* = g^a$ can simulate the random oracles G and H and can respond to $\mathcal{A}$'s various queries in the exactly the same way as this game does. Especially, when $\mathcal{A}$ queries $(y_1, u^*, v^*, \kappa^*)$ to H, $\mathcal{B}$ outputs $\kappa^*(= g^{ab})$ and halts. Consequently, we get $\Pr[F] \leq \mathsf{Adv}_{\mathcal{B}}^{\mathrm{gdh}}(\lambda)$.

Note also that in this game, the challenge ciphertext $\psi^* = (u^*, v^*, w^*)$ does not leak any information on $\beta \in \{0, 1\}$. This is because, the value $k^* = \mathsf{G}(\sigma^* \oplus w^*)$ is used only to "mask" m_β in v^*. (That is, k^* is used like a perfect one-time pad.) Hence, we get

$$\Pr[S_4] = 1/2.$$

Thus, we get the bound in the theorem statement.

Better Key Sizes (and Attacks) for LWE-Based Encryption

Richard Lindner[1,⋆] and Chris Peikert[2,⋆⋆]

[1] Technische Universität Darmstadt
rlindner@cdc.informatik.tu-darmstadt.de
[2] Georgia Institute of Technology
cpeikert@cc.gatech.edu

Abstract. We analyze the concrete security and key sizes of theoretically sound lattice-based encryption schemes based on the "learning with errors" (LWE) problem. Our main contributions are: (1) a new lattice attack on LWE that combines basis reduction with an enumeration algorithm admitting a time/success tradeoff, which performs better than the simple distinguishing attack considered in prior analyses; (2) concrete parameters and security estimates for an LWE-based cryptosystem that is more compact and efficient than the well-known schemes from the literature. Our new key sizes are up to 10 times smaller than prior examples, while providing even stronger concrete security levels.

Keywords: lattice-based cryptography, basis reduction, learning with errors.

1 Introduction

Recent years have seen significant progress in theoretically sound lattice-based cryptography, resulting in solutions to many tasks of wide applicability. In the realm of encryption alone, for example, we now have public-key cryptosystems [3, 32, 33] with chosen-ciphertext security [31, 28], identity-based encryption [17, 13, 1], and a fully homomorphic cryptosystem [16]. Much of this progress has been greatly aided by the use of simple and flexible average-case problems — namely, the *short integer solution* (SIS) introduced by Ajtai [2] and the *learning with errors* (LWE) problem of Regev [33] — that are provably as hard as certain lattice problems in the *worst case*, and appear to require time exponential in the main security parameter to solve.

For *practical* parameters, however, the concrete hardness of the SIS and LWE problems against algorithmic attacks is still far from a settled issue. This makes it difficult to assess the actual security and efficiency of cryptographic schemes that are based on these problems. The purpose of this paper is to shed further light on this issue, by considering new variants of known schemes and attacks, and analyzing their consequences in terms of key sizes and estimated security.

⋆ This work was supported by CASED (www.cased.de).
⋆⋆ This material is based upon work supported by the National Science Foundation under Grant CNS-0716786.

1.1 Our Contributions

We analyze the concrete security and efficiency of modern lattice-based cryptographic schemes, with a focus on LWE and public-key encryption. To start, we describe an LWE-based cryptosystem that has substantially smaller keys and ciphertexts than the more well-known systems in the literature (namely, the original system of Regev [33] and its more efficient amortized variants [30, 17]). Our scheme incorporates several techniques and perspectives from recent works; in particular, it is an instance of an abstract system described by Micciancio [24] that generalizes all the schemes of [33, 30, 17], and the system's design and security proof (under the LWE assumption) combine a variety of techniques from recent works [6, 25, 21, 29] to yield asymptotic and concrete improvements in key size. While there are not any new techniques involved, to our knowledge the literature lacks a full description and analysis of the system, despite it now being an important target of study.

Our second main contribution is a new and stronger way of using existing algorithmic attack tools, such as lattice basis reduction and bounded-distance decoding with preprocessing, to analyze the concrete security of recent lattice-based cryptosystems. Our attack is directed specifically at the LWE problem, and exploits some of its structural properties in ways that have not been attempted before in a cryptanalytic context. (Our attack also does not seem immediately applicable to other lattice problems, such as the unique shortest vector problem, that have been used for public-key encryption [3, 32, 4].) Therefore, we believe that our analysis gives a more accurate assessment of LWE's concrete hardness than estimates derived from prior lattice attacks.

Applying our attack to the improved cryptosystem, we then propose concrete parameters and (conservative) runtime estimates for modern commodity hardware. Despite our improved attacks, the resulting key sizes are still smaller than prior example parameters by factors as large as 10, even for stronger security levels. (See Section 6 for full details.) For example, using parameters that can encrypt a 128-bit payload and appear to be at least as secure as AES-128, we obtain public key sizes of about $1,120$ kilobits, or about 400 kilobits assuming a public source of trusted randomness.

Clearly, the above key sizes are still too large for many applications, but this is a consequence of the quadratic overhead inherent to the use "standard" LWE. By using the compact "ring-based" variant of LWE and cryptosystem from [22] (which is related to the heuristic NTRU scheme [18] and the theoretically sound line of works initiated in [23]), we can immediately shrink the above key sizes by a factor of at least 200. The resulting sizes of 2-5 kilobits are comparable to modern recommendations for RSA, and the cryptosystem itself is many times faster on modern hardware.

Our methodology. Here we briefly summarize our methods and main conclusions. Our approach involves a dedicated study of basis reduction for a certain family of random lattices, and a post-reduction decoding algorithm that to our knowledge have not been considered in prior analyses. (For a discussion of our approach in relation to prior works, see Section 1.2.)

Lattice-based cryptosystems in the line of works started by Ajtai [2] involve a family of so-called *q-ary lattices*, which are m-dimensional integer lattices that contain $q\mathbb{Z}^m$ as a sublattice, for some modulus $q \geq 2$. We study how basis reduction performs, in terms of its running time and the global properties of its output basis, on random lattices from this family. Our experiments yield reliable and theoretically well-behaved predictions about the basis quality that may be obtained using various amounts of computational effort.

Complementing our analysis of lattice basis reduction, we describe a new post-reduction attack on the *search* version of the LWE problem, and provide precise trade-offs between time and adversarial advantage (i.e., success probability) in terms of the given basis quality. Even though we attack the search-LWE problem, which is not strictly necessary to break the semantic security of most LWE-based cryptosystems, our full attack turns out to be strictly preferable (for a very wide range of parameters used in cryptography) to the natural distinguishing attack on *decision*-LWE that has been considered in prior analyses [25, 34]. Specifically, our attack can solve a search-LWE instance, and hence decrypt a ciphertext, with the same or better advantage than the distinguishing attack, while using lattice vectors of lower quality and hence much less total runtime. The improvement is especially pronounced in the high-advantage regime, where the adversary needs relatively high confidence in the decrypted plaintext, such as might be required for breaking hybrid encryption.

Our post-reduction attack involves a simple extension of Babai's "nearest-plane" algorithm [8] that allows us to trade basis quality against decoding time, which to our knowledge has not been explored in a cryptanalytic context. The extension is related to Klein's (de)randomized algorithm [19] for bounded-distant decoding, but is simpler and specifically tailored to the known Gaussian distribution of the error vector. As we have already indicated, the quality/time trade-off dramatically affects the quality of basis required to solve an LWE instance, and hence the running time of the attack.

Finally, we note that our analysis is entirely modular, and allows for substituting improved basis reduction algorithms (and their accompanying runtime and quality predictions) into the post-reduction attack.

1.2 Related Work

Several papers contain studies of the concrete hardness of lattice problems. Here we mention the ones most closely related to our work, which are aimed at calculating secure parameters for lattice-based cryptosystems, and describe the most important distinctions.

Gama and Nguyen [14] performed a comprehensive study of the behavior of basis reduction for various families of lattices. Their analysis is primarily focused on the best obtainable solutions to the Hermite-, Unique-, and Approximate-Shortest Vector Problems. The Hermite SVP is in particular an important problem in our work and other cryptanalyses. While Gama and Nguyen did not attempt to document the behavior of basis reduction on random q-ary lattices (aside from the closely related Goldstein-Mayer distribution for enormous q), our experiments confirmed several of their findings for this family (as did the experiments in [25]). Gama and Nguyen's study was aimed mainly at predicting the behavior of basis reduction, but did not include runtime predictions, nor did it investigate the *use* of a reduced basis to solve bounded-distance decoding problems such as LWE, where additional algorithmic trade-offs are possible.

The survey by Micciancio and Regev [25] proposed example parameters for various lattice-based schemes from the contemporary literature (which have larger keys than the one we describe here). Their parameters were derived using Gama and Nguyen's conclusions about the (in)feasibility of obtaining various Hermite factors, and as such do not include concrete estimates of attack runtimes or success probabilities. Their security estimates are calculated using the natural distinguishing attack on LWE by finding one relatively short vector in an associated lattice; our attack succeeds with lower-quality vectors, making it even more effective. (It should be noted that the example parameters given in [25] were already known to offer moderate security at best.)

Rückert and Schneider [34] recently gave concrete estimates of "symmetric bit security" for many recent lattice-based schemes, incorporating concrete runtime estimates for various Hermite factors in random q-ary lattices. Their analysis uses a permissive form of the distinguishing attack described in [25], in which the adversarial advantage is about 2^{-72}. This small advantage is not incorporated into their final bit security estimates, so the estimates are more conservative than ours, even without taking into account the superior decoding attack on search-LWE.

Finally, we note that the best distinguishing attack against LWE used in [25, 34] may not always apply to our cryptosystem, because its parameters can be set so that relatively few LWE samples are published, and thus the attack is forced to use a suboptimal lattice dimension. We give further details in Sections 5.1 and 6.

2 Preliminaries

For a positive integer k, we use $[k]$ to denote the set $\{1, \ldots, k\}$. The base-2 logarithm is denoted lg. We use bold lower-case letters (e.g., $\mathbf{x}$) to denote vectors over the real $\mathbb{R}$. We use bold upper-case letters (e.g., $\mathbf{B}$) for ordered sets of vectors, and identify the set with the matrix having the vectors as its columns. We let $\|\mathbf{B}\| := \max_i \|\mathbf{b}_i\|$, where $\|\cdot\|$ denotes the Euclidean norm.

For an (ordered) set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subset \mathbb{R}^n$, its *Gram-Schmidt orthogonalization* $\widetilde{\mathbf{B}}$ is defined iteratively as $\widetilde{\mathbf{b}_1} = \mathbf{b}_1$, and $\widetilde{\mathbf{b}}_i$ is the component of $\mathbf{b}_i$ orthogonal to span$(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ for $i = 2, \ldots, k$. In matrix notation, it corresponds to the (unique) decomposition $\mathbf{B} = \mathbf{QR}$, where the columns of $\mathbf{Q} \in \mathbb{R}^{n \times k}$ are orthonormal (i.e., $\mathbf{Q}^t\mathbf{Q} = \mathbf{I}$) and $\mathbf{R} \in \mathbb{R}^{k \times k}$ is right-triangular with positive diagonal entries; the Gram-Schmidt vectors are then $\widetilde{\mathbf{b}}_i = \mathbf{q}_i \cdot r_{i,i}$. For a set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$, its *fundamental parallelepiped* is $\mathcal{P}_{1/2}(\mathbf{B}) := \mathbf{B} \cdot [-\frac{1}{2}, \frac{1}{2})^k$.

A *lattice* Λ in $\mathbb{R}^m$ is a discrete additive subgroup. In this work we are concerned only with *q-ary* integer lattices, which are contained in $\mathbb{Z}^m$ and contain $q\mathbb{Z}^m$, i.e., $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$. Such a lattice is generated by a (non-unique) *basis* $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \subset \mathbb{Z}^m$ of linearly independent integer vectors, as $\Lambda = \mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^m = \{\sum_{i \in [m]} z_i \cdot \mathbf{b}_i : z_i \in \mathbb{Z}\}$. The determinant $\det(\Lambda)$ of such a lattice is its index as a subgroup of $\mathbb{Z}^m$, i.e., $\det(\Lambda) = |\mathbb{Z}^m : \Lambda|$. Equivalently, it is $|\det(\mathbf{B})|$ for any basis $\mathbf{B}$ of Λ.

Discrete Gaussians. For a lattice Λ and a positive real $s > 0$, the *discrete Gaussian* distribution $D_{\Lambda,s}$ over Λ with parameter s is the probability distribution having support Λ that assigns a probability proportional to $\exp(-\pi\|\mathbf{x}\|^2/s^2)$ to each $\mathbf{x} \in \Lambda$. For $\Lambda = \mathbb{Z}^n$, it is easy to see (by orthonormality of its standard basis) that the discrete Gaussian $D_{\mathbb{Z}^n,s}$ is simply the product distribution of n independent copies of $D_{\mathbb{Z},s}$. There are efficient algorithms for sampling from a distribution within negligible statistical distance of $D_{\mathbb{Z},s}$, given any $s > 0$. (See, e.g., [17]: for arbitrary s there is a rejection sampling algorithm, and for small s one can compute a close approximation to the cumulative distribution function.).

We will need two tail bounds on discrete Gaussians.

Lemma 1 ([9, Lemma 1.5]). *Let $c \geq 1$ and $C = c \cdot \exp(\frac{1-c^2}{2}) < 1$. Then for any real $s > 0$ and any integer $n \geq 1$, we have*

$$\Pr\left[\|D_{\mathbb{Z}^n,s}\| \geq c \cdot \tfrac{1}{\sqrt{2\pi}} \cdot s\sqrt{n}\right] \leq C^n.$$

Lemma 2 ([10, Lemma 2.4]). *For any real $s > 0$ and $T > 0$, and any $\mathbf{x} \in \mathbb{R}^n$, we have*

$$\Pr\left[|\langle \mathbf{x}, D_{\mathbb{Z}^n,s}\rangle| \geq T \cdot s\|\mathbf{x}\|\right] < 2\exp(-\pi \cdot T^2).$$

Learning with errors. The *learning with errors* (LWE) problem was introduced by Regev [33] as a generalization of the well-known 'learning parity with noise' problem, to larger moduli. The problem is parameterized by a dimension $n \geq 1$ and an integer modulus $q \geq 2$, as well as an error distribution χ over $\mathbb{Z}$ (or its induced distribution over $\mathbb{Z}_q$). In this work we will be concerned only with discrete Gaussian error distributions $\chi = D_{\mathbb{Z},s}$ over the integers, where $\alpha := s/q \in (0,1)$ is often called the (relative) *error rate.*

For an $\mathbf{s} \in \mathbb{Z}_q^n$, the LWE distribution $A_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing a uniformly random $\mathbf{a} \in \mathbb{Z}_q^n$ and error term $e \leftarrow \chi$, and outputting the pair $(\mathbf{a}, t = \langle \mathbf{a}, \mathbf{s}\rangle + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The *search* version of the LWE problem is, given any desired number of independent samples $(\mathbf{a}_i, t_i) \leftarrow A_{\mathbf{s},\chi}$, to find $\mathbf{s}$. The *decision* version of LWE is to distinguish, with non-negligible advantage, between any desired number of independent samples $(\mathbf{a}_i, t_i) \leftarrow A_{\mathbf{s},\chi}$ (for a uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$), and the same number of independent samples drawn from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. It is often convenient to write these problems in matrix form as follows: collecting the vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$ as the columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and the (implicit) error terms $e_i \in \mathbb{Z}$ and values $t_i \in \mathbb{Z}_q$ as the entries of vectors $\mathbf{e} \in \mathbb{Z}^m$, $\mathbf{t} \in \mathbb{Z}_q^m$ respectively, we are given the input

$$\mathbf{A}, \quad \mathbf{t} = \mathbf{A}^t\mathbf{s} + \mathbf{e} \bmod q$$

and are asked to find $\mathbf{s}$, or to distinguish the input from a uniformly random $(\mathbf{A}, \mathbf{t})$. The LWE problem may also be viewed as an average-case 'bounded-distance decoding' problem on a certain family of lattices: for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, define the lattice

$$\Lambda(\mathbf{A}^t) = \{\mathbf{z} \in \mathbb{Z}^m : \exists\, \mathbf{s} \in \mathbb{Z}_q^n \text{ such that } \mathbf{z} = \mathbf{A}^t\mathbf{s} \bmod q\}.$$

Then the $\mathbf{t}$ component of the LWE input may be seen as a perturbed lattice point in $\Lambda(\mathbf{A}^t)$, to be decoded.

Hardness of LWE. We recall several facts from the literature about the provable hardness of LWE. The first is that for error distribution $\chi = D_{\mathbb{Z},\alpha\cdot q}$ where $\alpha \cdot q \geq 2\sqrt{n}$, the search version of LWE is at least as hard as *quantumly* approximating certain worst-case problems on n-dimensional lattices to within $\tilde{O}(n/\alpha)$ factors [33].[1] Moreover, for similar parameters and large enough q, search-LWE is at least as hard as *classically* approximating the decision shortest vector problem and variants [28]. For moduli q that are sufficiently 'smooth' (i.e., products of small enough primes), the decision form of LWE is at least as hard as the search form [33, 28].

A particularly important fact for our purposes is that decision-LWE becomes no easier to solve even if the secret $\mathbf{s}$ is chosen from the error distribution χ, rather than uniformly at random [25, 7]. This may be seen as follows: given access to $A_{\mathbf{s},\chi}$, we can draw many samples to obtain

$$\mathbf{A}^t = \begin{bmatrix}\mathbf{A}_1^t \\ \mathbf{A}_2^t\end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix}\mathbf{t}_1 \\ \mathbf{t}_2\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1^t \\ \mathbf{A}_2^t\end{bmatrix}\mathbf{s} + \begin{bmatrix}\mathbf{e}_1 \\ \mathbf{e}_2\end{bmatrix} = \mathbf{A}^t\mathbf{s} + \mathbf{e} \bmod q,$$

where $\mathbf{A}_2$ is uniform, $\mathbf{e}$ is drawn from χ, and $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times n}$ is *square* and *invertible.* (This follows by forming $\mathbf{A}_1$ by greedily drawing samples that can form an invertible matrix, and disposing of any others until $\mathbf{A}_1$ is complete.) We can then transform $\mathbf{A}$ and $\mathbf{t}$ into

$$\bar{\mathbf{A}}^t := -\mathbf{A}_2^t \cdot \mathbf{A}_1^{-t} \bmod q, \quad \bar{\mathbf{t}} := \bar{\mathbf{A}}^t\mathbf{t}_1 + \mathbf{t}_2 = \bar{\mathbf{A}}^t\mathbf{e}_1 + \mathbf{e}_2 \bmod q,$$

where $\bar{\mathbf{A}}$ is uniform; therefore, we have effectively replaced $\mathbf{s}$ with the error vector $\mathbf{e}_1$. On the other hand, when $\mathbf{A}, \mathbf{t}$ are uniformly random, then so are $\bar{\mathbf{A}}, \bar{\mathbf{t}}$.

In terms of lattices, the above may be interpreted as follows: using the bijection $\mathbf{s} \mapsto \mathbf{A}_1^t\mathbf{s}$ from $\mathbb{Z}_q^n$ to itself, we can see that the lattice $\Lambda(\mathbf{A}^t)$ defined above has as a basis the matrix

$$\mathbf{H} = \begin{bmatrix} & \mathbf{I} \\ q\mathbf{I} & -\bar{\mathbf{A}}^t\end{bmatrix}.$$

(This basis $\mathbf{H}$ is a canonical representation of $\Lambda(\mathbf{A}^t)$ known as the Hermite normal form. We have ordered the basis vectors so that the Gram-Schmidt vectors

[1] It is important to note that the original hardness result of [33] is for a *continuous* Gaussian error distribution, which when rounded naively to the nearest integer does not produce a true discrete Gaussian. Fortunately, a suitable randomized rounding method does so [29].

of $\mathbf{H}$ are integer multiples of the standard basis vectors, where the first several have length q, and the remainder have length 1.) Because $\mathbf{A}^t\mathbf{s} \bmod q \in \Lambda(\mathbf{A}^t)$, we have $\mathbf{t} = \mathbf{A}^t\mathbf{s} + \mathbf{e} = \mathbf{e} \bmod \mathbf{H}$, which is

$$\mathbf{e} - \mathbf{H}\mathbf{e}_1 = \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_2 + \bar{\mathbf{A}}^t\mathbf{e}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \bar{\mathbf{t}} \end{bmatrix} \bmod q.$$

In conclusion, $\bar{\mathbf{t}} = \bar{\mathbf{A}}^t\mathbf{e}_1 + \mathbf{e}_2$ is the unique canonical representative of $\mathbf{e}$ modulo the lattice $\Lambda(\mathbf{A}^t)$.

Finally, assuming hardness of decision-LWE, a standard hybrid argument over the columns of $\mathbf{E}$ (see, e.g., [31]) shows that $(\bar{\mathbf{A}}, \bar{\mathbf{A}}^t\mathbf{E}_1 + \mathbf{E}_2)$ is indistinguishable from uniform, where the entries of $\mathbf{E} = \left[\begin{smallmatrix}\mathbf{E}_1 \\ \mathbf{E}_2\end{smallmatrix}\right]$ are chosen independently from χ.

3 LWE-Based Encryption

Here we describe an LWE-based cryptosystem that is more space-efficient than the ones commonly known in the literature. It is an instance of an abstract system described by Micciancio [24] that generalizes all the schemes of [33, 30, 17], though a full description and analysis of the generalized system has not appeared in the literature. The security proof combines a number of techniques and perspectives from recent works [25, 21, 29] for the purpose of improved efficiency and a tight analysis. An efficient ring-based analogue of the system is described in the full version of [22].

Despite being a generalization of prior LWE-based cryptosystems, the present scheme can actually be instantiated to have keys and ciphertexts that are smaller by a factor of about $\lg q$, while simultaneously *improving* the concrete security! The improved security comes from the smaller keys (for given security parameter n), which allows for a relatively larger noise rate that makes the LWE problem harder. The smaller keys come from a different style of security proof, which is very similar to the proofs for the coding-based cryptosystem of Alekhnovich [6] and the subset sum-based cryptosystem of Lyubashevsky, Palacio, and Segev [21]. In brief, the proof uses the LWE assumption *twice* (first on the public key, and then again on the ciphertext) to show that the adversary's view in a passive attack is indistinguishable from uniformly random. By contrast, the proofs for prior LWE-based schemes involve a statistical argument on either the public key or ciphertext, but this requires larger keys. We point out that statistical arguments still appear necessary for many advanced applications of LWE, such as identity-based encryption [17] and others that use a 'trapdoor basis,' and we do not know whether comparably small keys and ciphertexts can be obtained for these schemes.

Cryptosystem. The cryptosystem involves a few parameters: an integer modulus $q \geq 2$ and integer dimensions $n_1, n_2 \geq 1$, which relate to the underlying LWE problems; Gaussian parameters s_k and s_e for key generation and encryption, respectively; and a message alphabet Σ (for example, $\Sigma = \{0, 1\}$) and message length $\ell \geq 1$.

We also require a simple error-tolerant encoder and decoder, given by functions $\mathsf{encode}\colon \Sigma \to \mathbb{Z}_q$ and $\mathsf{decode}\colon \mathbb{Z}_q \to \Sigma$, such that for some large enough threshold $t \geq 1$, $\mathsf{decode}(\mathsf{encode}(m) + e \bmod q) = m$ for any integer $e \in [-t, t)$. For example, if $\Sigma = \{0, 1\}$, then we can define $\mathsf{encode}(m) := m \cdot \lfloor \frac{q}{2} \rfloor$, and $\mathsf{decode}(\bar{m}) := 0$ if $\bar{m} \in \left[-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor\right) \subset \mathbb{Z}_q$, and 1 otherwise. This method has error tolerance $t = \lfloor \frac{q}{4} \rfloor$. We also extend encode and decode to vectors, component-wise.

To get the smallest public keys, our system makes use of a uniformly random public matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n_1 \times n_2}$ that is generated by a trusted source, and is used by all parties in the system. If there is no trusted source, then $\bar{\mathbf{A}}$ may be chosen by the user herself as part of key generation, and included in the public key.

– $\mathsf{Gen}(\bar{\mathbf{A}}, 1^\ell)$: choose $\mathbf{R}_1 \leftarrow D_{\mathbb{Z}, s_k}^{n_1 \times \ell}$ and $\mathbf{R}_2 \leftarrow D_{\mathbb{Z}, s_k}^{n_2 \times \ell}$, and let $\mathbf{P} = \mathbf{R}_1 - \bar{\mathbf{A}} \cdot \mathbf{R}_2 \in \mathbb{Z}_q^{n_1 \times \ell}$. The public key is $\mathbf{P}$ (and $\bar{\mathbf{A}}$, if needed), and the secret key is $\mathbf{R}_2$.
In matrix form, the relationship between the public and secret keys is:

$$\left[\bar{\mathbf{A}}\ \mathbf{P}\right] \cdot \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix} = \mathbf{R}_1 \bmod q. \tag{1}$$

– $\mathsf{Enc}(\bar{\mathbf{A}}, \mathbf{P}, \mathbf{m} \in \Sigma^\ell)$: choose $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \times \mathbb{Z}^\ell$ with each entry drawn independently from $D_{\mathbb{Z}, s_e}$. Let $\bar{\mathbf{m}} = \mathsf{encode}(\mathbf{m}) \in \mathbb{Z}_q^\ell$, and compute the ciphertext

$$\mathbf{c}^t = \left[\mathbf{c}_1^t\ \mathbf{c}_2^t\right] = \left[\mathbf{e}_1^t\ \mathbf{e}_2^t\ \mathbf{e}_3^t + \bar{\mathbf{m}}^t\right] \cdot \begin{bmatrix} \bar{\mathbf{A}} & \mathbf{P} \\ \mathbf{I} & \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{1 \times (n_2 + \ell)}. \tag{2}$$

– $\mathsf{Dec}(\mathbf{c}^t = [\mathbf{c}_1^t, \mathbf{c}_2^t], \mathbf{R}_2)$: output $\mathsf{decode}(\mathbf{c}_1^t \cdot \mathbf{R}_2 + \mathbf{c}_2^t)^t \in \Sigma^\ell$.
Using Equation (2) followed by Equation (1), we are applying decode to

$$\left[\mathbf{c}_1^t\ \mathbf{c}_2^t\right] \cdot \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix} = \left(\mathbf{e}^t + \left[\mathbf{0}\ \mathbf{0}\ \bar{\mathbf{m}}^t\right]\right) \cdot \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix} = \mathbf{e}^t \cdot \mathbf{R} + \bar{\mathbf{m}}^t,$$

where $\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix}$. Therefore, decryption will be correct as long as each $|\langle \mathbf{e}, \mathbf{r}_j \rangle| < t$, the error threshold of decode. (We give a formal analysis below.)

For another perspective on this scheme as an (approximate) key-agreement mechanism, let $\ell = 1$ for simplicity. By the discussion in Section 2, we can interpret key generation as reducing a Gaussian error vector $\mathbf{r}$ modulo a lattice defined by $\bar{\mathbf{A}}$, and publishing the result $\bar{\mathbf{A}}\mathbf{r}_2 - \mathbf{r}_1 \bmod q$. Likewise, we can view encryption as reducing a Gaussian error vector $\mathbf{e}$ modulo the *dual* of the same lattice, and publishing the result $\mathbf{e}_1^t \bar{\mathbf{A}} + \mathbf{e}_2^t \bmod q$. Using their respective private error vectors and the other party's public message, the sender and receiver can both (approximately) compute $\mathbf{e}_1^t \bar{\mathbf{A}} \mathbf{r}_2 \in \mathbb{Z}_q$, whereas a passive adversary cannot. A formal proof of security appears below.

Due to space constraints, we defer the description of an analogous scheme based on the decision *ring*-LWE problem [22] to the full version. For messages of length any $\ell \leq n = n_1 = n_2$, and using the same values of n and q as above, the public and secret keys are about an n factor smaller than in the above system, namely $n \lg q$ or $2n \lg q$ bits at most, depending on the availability of a common trusted string. (The ciphertext size is the same, namely $2n \lg q$ bits.)

Parameters for correctness. Here we give an upper bound on the Gaussian parameters s_k, s_e in terms of the desired per-symbol error probability δ. For reasonably small values of δ, correctness for the entire message can effectively be guaranteed by way of a simple error-correcting code.

One small subtlety is that if a portion of the random vector $\mathbf{e}$ used for encryption happens to be 'too long,' then the probability of decryption error for every symbol can be unacceptably large. We address this by giving a bound on $\mathbf{e}$, in Equation (4) below, which is violated with probability at most $2^{-\kappa}$ for some statistical parameter κ (say, $\kappa = 40$ for concreteness). We then calculate the error probabilities assuming that the bound holds; the overall decryption error probability is then no more than $2^{-\kappa}$ larger. One can also modify the Enc algorithm to reject and resample any $\mathbf{e}$ that violates Equation (4); the adversary's advantage can increase by at most $2^{-\kappa}$.

Lemma 3 (Correctness). *In the cryptosystem from Section 3, the error probability per symbol (over the choice of secret key) is bounded from above by any desired $\delta > 0$, as long as*

$$s_k \cdot s_e \leq \frac{\sqrt{2}\pi}{c} \cdot \frac{t}{\sqrt{(n_1 + n_2) \cdot \ln(2/\delta)}}. \tag{3}$$

Here $c \geq 1$ is a value that depends (essentially) only on $n_1 + n_2$; representative values are given in Figure 1.

Proof. As shown above in the specification of the decryption algorithm, the jth symbol of the message decrypts correctly if $|\langle \mathbf{e}, \mathbf{r}_j \rangle| < \lfloor \frac{q}{4} \rfloor$. Recall that the entries of $\mathbf{e} \in \mathbb{Z}^{n_1+n_2+\ell}$ are independent and have distribution $D_{\mathbb{Z},s_e}$, and $\mathbf{r}_j \in \mathbb{Z}^{n_1+n_2+\ell}$ is the jth column of $\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix}$, where the entries of $\mathbf{R}_1$ and $\mathbf{R}_2$ are drawn independently from $D_{\mathbb{Z},s_k}$.

To bound the error probability, let $\bar{\mathbf{e}} \in \mathbb{Z}^{n_1+n_2}$ consist of the first $n_1 + n_2$ entries of $\mathbf{e}$. Then by Lemma 1, there is a $c \geq 1$ such that

$$\|\bar{\mathbf{e}}\| \leq c \cdot \frac{1}{\sqrt{2\pi}} \cdot s_e \sqrt{n_1 + n_2} \tag{4}$$

except with very small probability (concrete values of c are given in Figure 1). For any fixed $\bar{\mathbf{e}}$ satisfying the above bound, observe that each $\langle \mathbf{e}, \mathbf{r}_j \rangle$ is independent and distributed essentially as $\langle \bar{\mathbf{e}}, D_{\mathbb{Z},s_k}^{n_1+n_2} \rangle$. By Lemma 2, for any $T \geq 0$ we have

$(n_1 + n_2)$	$c \geq$	$(s_k \cdot s_e)/t \leq$
256	1.35	0.08936
384	1.28	0.07695
512	1.25	0.06824
640	1.22	0.06253

Fig. 1. Bounds on parameters for Lemma 3 using a per-symbol error probability of $\delta = 0.01$, where c is determined so that the probability of choosing a 'bad' encryption vector $\mathbf{e}$ is at most 2^{-40}.

$$\Pr\left[\left|\langle \bar{\mathbf{e}}, D_{\mathbb{Z},s_k}^{n_1+n_2}\rangle\right| \geq T \cdot s_k \|\bar{\mathbf{e}}\|\right] < 2\exp(-\pi \cdot T^2).$$

Letting $T = t/(s_k\|\bar{\mathbf{e}}\|)$, where t is the error tolerance of our message encoding, and using the bound on $\|\bar{\mathbf{e}}\|$ from above, we get the bound on $s_k \cdot s_e$ from the lemma statement.

Theorem 1. *The cryptosystem describe above is CPA-secure, assuming the hardness of decision-*LWE *with modulus q for: (i) dimension n_2 with error distribution $D_{\mathbb{Z},s_k}$, and (ii) dimension n_1 with error $D_{\mathbb{Z},s_e}$.*

Proof. It suffices to show that the entire view of the adversary in an IND-CPA attack is computationally indistinguishable from uniformly random, for any encrypted message $\mathbf{m} \in \Sigma^\ell$. The view consists of $(\bar{\mathbf{A}}, \mathbf{P}, \mathbf{c})$, where $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n_1 \times n_2}$ is uniformly random, $\mathbf{P} \leftarrow \mathsf{Gen}(\bar{\mathbf{A}}, 1^\ell)$, and $\mathbf{c}^t \leftarrow \mathsf{Enc}(\bar{\mathbf{A}}, \mathbf{P}, \mathbf{m})$. First, $(\bar{\mathbf{A}}, \mathbf{P})$ is computationally indistinguishable from uniformly random $(\bar{\mathbf{A}}, \mathbf{P}^*) \in \mathbb{Z}_q^{n_1 \times (n_2+\ell)}$ under assumption (i) in the lemma statement, because $\mathbf{P} = (\bar{\mathbf{A}}^t)^t \cdot (-\mathbf{R}_2) + \mathbf{R}_1$, and $\bar{\mathbf{A}}^t$ is uniform while the entries of both $-\mathbf{R}_2$ and $\mathbf{R}_1$ are drawn from $D_{\mathbb{Z},s_k}$. So the adversary's view is indistinguishable from $(\mathbf{A}, \mathbf{c})$ where $\mathbf{A} = (\bar{\mathbf{A}}, \mathbf{P}^*)$ is uniformly random and $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{A}, \mathbf{m})$. Now $(\mathbf{A}, \mathbf{c})$ is also computationally indistinguishable from uniformly random $(\mathbf{A}, \mathbf{c}^*)$ under assumption (ii) in the lemma statement, because $\mathbf{c} = (\mathbf{A}^t\mathbf{e}_1 + \left[\begin{smallmatrix}\mathbf{e}_2\\ \mathbf{e}_3\end{smallmatrix}\right]) + \left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{m}\end{smallmatrix}\right]$, and $\mathbf{A}$ is uniform while the entries of $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ are drawn from $D_{\mathbb{Z},s_e}$.

It should be noted that for some settings of the parameters, one of the two assumptions in Theorem 1 may be true *information-theoretically* for the number of LWE samples exposed by the system in an attack. For instance, if $n_2 \geq n_1 \lg q$ and $s_k \geq \omega(\sqrt{\log n_1})$, then the public key $(\bar{\mathbf{A}}, \mathbf{P})$ is within a negligible (in n_1) statistical distance of uniformly random (by a suitable version of the leftover hash lemma), whereas the corresponding ciphertexts are statistically far from uniform. These properties are important in, for example, the 'dual' cryptosystem and identity-based encryption scheme of [17]. Conversely, the applications found in [30, 11, 7] have public keys that are far from uniform, but require that encryption under a 'malformed' (uniformly random) public key produces a ciphertext that is statistically independent of the encrypted message. These properties are achieved when $n_1 \geq n_2 \lg q$ and $s_e \geq \omega(\sqrt{\log n_2})$, again by the leftover hash lemma.

4 Lattice Decoding Attacks

The most promising practical attacks on the cryptosystem from Section 3, and more generally on LWE itself, use lattice-basis reduction followed by a decoding phase using the reduced basis.[2] In this section we analyze the performance of decoding as it relates to the quality of a given reduced basis. Then in Section 5 we analyze the effort required to obtain bases of a desired quality.

Before proceeding, we briefly explain how our *decoding* attack on LWE differs from the *distinguishing* attacks considered in other works [25, 34]. In the latter, the adversary distinguishes (with some noticeable advantage) an LWE instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}^t\mathbf{s} + \mathbf{e})$ from uniformly random, which is typically enough to break the semantic security of an LWE-based cryptosystem with the same advantage. To do this, the adversary finds a short nonzero integral vector $\mathbf{v}$ such that $\mathbf{A}\mathbf{v} = \mathbf{0} \bmod q$, which may be seen as a short vector in the (scaled) dual of the LWE lattice $\Lambda(\mathbf{A}^t)$. (Equivalently, the points of $\Lambda(\mathbf{A}^t)$ may be partitioned into hyperplanes orthogonal to $\mathbf{v}$, successively separated by distance $q/\|\mathbf{v}\|$.) The adversary then simply tests whether the inner product $\langle \mathbf{v}, \mathbf{t} \rangle$ is "close" to zero modulo q. When $\mathbf{t}$ is uniform, the test accepts with probability exactly 1/2, but when $\mathbf{t} = \mathbf{A}^t\mathbf{s}+\mathbf{e}$ for Gaussian $\mathbf{e}$ with parameter s, we have $\langle \mathbf{v}, \mathbf{t} \rangle = \langle \mathbf{v}, \mathbf{e} \rangle \bmod q$, which is essentially a Gaussian (reduced mod q) with parameter $\|\mathbf{v}\| \cdot s$. When this parameter is not much larger than q, the Gaussian (mod q) can be distinguished from uniform with advantage very close to $\exp(-\pi \cdot (\|\mathbf{v}\| \cdot s/q)^2)$. For example, when $\|\mathbf{v}\| = 4q/s$ the distinguishing advantage is about 2^{-72}. However, to distinguish (and hence decrypt a ciphertext) with high confidence, one needs $\|\mathbf{v}\| \leq q/(2s)$ or so, which usually requires a great deal more effort to obtain.

It is customary to include the inverse distinguishing advantage in the total 'cost' of an attack, so the computational effort and advantage need to be carefully balanced. For practical parameters, the optimal total cost of the distinguishing attack typically involves a very small distinguishing advantage (see Section 6), which may not be very useful in some settings, such as hybrid encryption.

Our decoding attack is stronger than the distinguishing attack in that it can actually recover the secret error vector in the LWE instance (and hence decrypt the ciphertext) with the same or better advantage, while using lower-quality vectors. For all the parameter settings that we investigated, our attack yields a better total effort as a ratio of time/advantage, and it is significantly more efficient in the high-advantage regime. (See Section 6 and Figure 3 in particular for details.) The attack works by using an entire reduced basis (not just one vector), and by expending some additional post-reduction effort to find the LWE solution. We also point out that unlike in basis reduction, the post-reduction effort is fully parallelizable.

[2] There are also purely combinatorial attacks on LWE [12, 38] that may perform asymptotically better than lattice reduction, but so far not in practice. Also, these attacks generally require more LWE samples than our cryptosystem exposes, and an exponentially large amount of space.

The attack. Recall that an LWE instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}^t\mathbf{s} + \mathbf{e})$ may be seen as a bounded-distance decoding problem on a certain lattice $\Lambda = \Lambda(\mathbf{A}^t)$, where $\mathbf{A}^t\mathbf{s} \in \Lambda$.

The standard method for solving a bounded-distance decoding problem on lattices is the recursive NearestPlane algorithm of Babai [8]. The input to the algorithm is some lattice basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ (which for best results should be as reduced as possible) and a target point $\mathbf{t} \in \mathbb{R}^m$, and the output is a lattice point $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ that is 'relatively close' to $\mathbf{t}$. The precise guarantee is that for any $\mathbf{t} \in \text{span}(\mathbf{B})$, $\mathsf{NearestPlane}(\mathbf{B}, \mathbf{t})$ returns the unique $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\mathbf{t} \in \mathbf{v} + \mathcal{P}_{1/2}(\widetilde{\mathbf{B}})$. In other words, if $\mathbf{t} = \mathbf{v} + \mathbf{e}$ for some $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, the algorithm outputs $\mathbf{v}$ if and only if $\mathbf{e}$ happens to lie in $\mathcal{P}_{1/2}(\widetilde{\mathbf{B}})$.

The main drawback of this approach in attacking LWE is that in a reduced basis $\mathbf{B}$, the last several Gram-Schmidt vectors of $\mathbf{B}$ are typically very short, whereas the first few are relatively long. In such a case, the parallelepiped $\mathcal{P}_{1/2}(\widetilde{\mathbf{B}})$ is very 'long and skinny,' and so the Gaussian error vector $\mathbf{e}$ is very unlikely to land in it, causing NearestPlane to produce an incorrect answer.

We address this issue by giving a generalized algorithm that admits a time/success tradeoff. It works just as NearestPlane does, except that it can recurse on some $d_i \geq 1$ distinct planes in the ith level of the recursion. In essence, the multiple recursion has the effect of making the parallelepiped $\mathcal{P}_{1/2}(\widetilde{\mathbf{B}})$ wider in the direction of $\widetilde{\mathbf{b}}_i$ by a factor of exactly d_i.[3] To capture the most probability mass of the Gaussian error distribution of $\mathbf{e}$, one should choose the multiples d_i so as to maximize $\min_i(d_i \cdot \|\widetilde{\mathbf{b}}_i\|)$.[4]

The input to our NearestPlanes algorithm is a lattice basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subset \mathbb{R}^m$, a vector $\mathbf{d} = (d_1, \ldots, d_k) \in (\mathbb{Z}^+)^k$ of positive integers, and a target point $\mathbf{t} \in \mathbb{R}^m$. It outputs a set of $\prod_{i\in[k]} d_i$ distinct lattice vectors in $\mathcal{L}(\mathbf{B})$, as follows:

1. If $k = 0$, return $\mathbf{0}$. Else, let $\mathbf{v}$ be the projection of $\mathbf{t}$ onto span$(\mathbf{B})$.
2. Let $c_1, \ldots, c_{d_k} \in \mathbb{Z}$ be the d_k distinct integers closest to $\langle \widetilde{\mathbf{b}_k}, \mathbf{v}\rangle / \langle \widetilde{\mathbf{b}_k}, \widetilde{\mathbf{b}_k}\rangle$.
3. Return $\displaystyle\bigcup_{i\in[d_k]} (c_i \cdot \mathbf{b}_k + \mathsf{NearestPlanes}(\{\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}\}, (d_1, \ldots, d_{k-1}), \mathbf{v} - c_i \cdot \mathbf{b}_k)$.

Note that the recursive calls to NearestPlanes can be run entirely in parallel. The following lemma is an immediate extension of the analysis from [8].

[3] The algorithm of Klein [19] also can recurse on more than one plane per iteration. Klein's algorithm solves the general bounded-distance decoding problem, and selects the planes at each stage probabilistically (though it can also be derandomized); its guarantee is related solely to the shortest Gram-Schmidt vector in the basis. Our algorithm is tailored specifically to the setting where we know the distribution of the offset vector; this allows the algorithm to recurse on exactly those planes that maximize the probability of success (over the choice of the error vector).

[4] One could further generalize the algorithm to search within an approximate *ball* made up of 'bricks' that are copies of $\mathcal{P}_{1/2}(\widetilde{\mathbf{B}})$, thus capturing even more of the Gaussian without adding much more to the search space. However, this would significantly complicate the analysis, and we find that the present approach is already very effective.

Lemma 4. *For* $\mathbf{t} \in \operatorname{span}(\mathbf{B})$*,* $\mathsf{NearestPlanes}(\mathbf{B}, \mathbf{d}, \mathbf{t})$ *returns the set of all* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $\mathbf{t} \in \mathbf{v} + \mathcal{P}_{1/2}(\widetilde{\mathbf{B}} \cdot \mathbf{D})$*, where* $\mathbf{D} = \operatorname{diag}(\mathbf{d})$*. The running time is essentially* $\prod_{i \in [k]} d_i$ *times as large as that of* $\mathsf{NearestPlane}(\mathbf{B}, \mathbf{t})$*.*

Note that the columns of $\widetilde{\mathbf{B}} \cdot \mathbf{D}$ from the lemma statement are the orthogonal vectors $d_i \cdot \widetilde{\mathbf{b}}_i$, so $\mathcal{P}_{1/2}(\widetilde{\mathbf{B}} \cdot \mathbf{D})$ is a rectangular parallelepiped with axis lengths $d_i \cdot \|\widetilde{\mathbf{b}}_i\|$.

Success probability of $\mathsf{NearestPlanes}$. When $\mathbf{t} = \mathbf{v} + \mathbf{e}$ for some $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ and a *continuous* Gaussian $\mathbf{e} \leftarrow D_s$ for some $s > 0$, the probability that $\mathbf{v}$ is in the output set of $\mathsf{NearestPlanes}(\mathbf{B}, \mathbf{d}, \mathbf{t})$ is

$$\prod_{i=1}^{m} \Pr\left[|\langle \mathbf{e}, \widetilde{\mathbf{b}}_i \rangle| < d_i \cdot \langle \widetilde{\mathbf{b}}_i, \widetilde{\mathbf{b}}_i \rangle / 2\right] = \prod_{i=1}^{m} \operatorname{erf}\left(\frac{d_i \cdot \|\widetilde{\mathbf{b}}_i\| \sqrt{\pi}}{2s} \right), \tag{5}$$

which follows by the independence of the values $\langle \mathbf{e}, \widetilde{\mathbf{b}}_i \rangle$, due to the orthogonality of the Gram-Schmidt vectors $\widetilde{\mathbf{b}}_i$. When $\mathbf{e}$ is drawn from a sufficiently wide *discrete* Gaussian over the integer lattice (in practice, a parameter of 6 or more suffices), the above is an extremely close approximation to the true probability.

We conclude this section by giving an informal explanation for why the advantage of the decoding attack can potentially be much larger than that of the distinguishing attack above, given vectors of the same quality. In the distinguishing attack, using a vector $\mathbf{v}$ of length (say) $\|\mathbf{v}\| \approx 4q/s$ implies that $\langle \mathbf{v}, \mathbf{t} \rangle \bmod q$ is distributed roughly as D_{4q} modulo q, whose statistical distance is only about 2^{-72} from uniform. A basis $\mathbf{B}$ of $\Lambda(\mathbf{A}^t)$ of equivalent quality has $\|\widetilde{\mathbf{b}}_m\| = q/\|\mathbf{v}\| = s/4$, because $\Lambda(\mathbf{A}^t)$ lies in hyperplanes orthogonal to $\mathbf{v}$ and separated by distance $q/\|\mathbf{v}\|$. So even without using multiple recursion in $\mathsf{NearestPlanes}$ (i.e., letting every $d_m = 1$), the corresponding term in Equation (5) is $\operatorname{erf}(\sqrt{\pi}/8) \approx 0.25$; moreover, the remaining terms typically approach 1 very rapidly, since $\|\widetilde{\mathbf{b}}_i\|$ usually increases quickly as i decreases. Letting $d_i > 1$ increases the overall success probability even more at little added cost, and allows for obtaining a relatively large advantage without needing higher-quality basis vectors.

5 Basis Reduction and Experiments

In this section we present an analysis of lattice basis reduction on random q-ary lattices arising from LWE, and results of reduction experiments on various parameters. Our goal is to predict a conservative, but still useful, lower bound on the practical runtime of the lattice decoding attack described in Section 4 for a given set of LWE parameters.

We found that the best practical lattice reduction algorithm currently available to us is the BKZ algorithm as implemented by Shoup in the NTL library [37], so this is what we used in our experiments. The BKZ algorithm is parameterized by a blocksize k between 2 and the dimension of the lattice to be reduced. As the blocksize increases, the reduced basis improves in quality (i.e., it contains shorter lattice vectors, whose Gram-Schmidt lengths are closer together), but the runtime of BKZ also rapidly increases, becoming practically infeasible for $k \geq 30$ or so.

There has been some recent progress in the development of algorithms for finding short vectors in lattices, which can be used as subroutines to (or entire replacements of) BKZ reduction. For example, Gama, Nguyen, and Regev [15] recently proposed a new method called "Extreme Enum", which is much faster than its predecessor, the Schnorr-Euchner enumeration [36]. There are also single-exponential time algorithms for the Shortest Vector Problem [5, 27, 26], which can run faster in practice than Schnorr-Euchner enumeration in certain low dimensions; however, these algorithms also require exponential space. We were not able to evaluate the performance and effectiveness of all these approaches, leaving this for future work. The BKZ implementation we use employs Schnorr-Euchner enumeration and, since the BKZ framework uses the enumeration subroutine as a black box, we presume that new algorithms incorporating Extreme Enum and other approaches will soon be available for evaluation. (For a comparison of enumeration algorithms in practice, see the open SVP-challenge website.[5])

In Section 5.1, we analyze the main properties of BKZ-reduced bases for q-ary lattices that are relevant to our decoding attack. In Section 5.2, we use our experiments to estimate the runtime required to obtain bases of a desired quality. We point out that the rest of our analysis is independent of this estimate, and can easily be applied with other runtime estimates for BKZ variants or other approaches.

5.1 Basis Reduction for *q*-ary Lattices

We begin by reviewing some of the prior work on basis reduction, in particular as applied to the q-ary lattices that arise from LWE.

The analysis of lattice reduction algorithms by Gama and Nguyen [14] identified the *Hermite factor* of the reduced basis as the dominant parameter in the runtime of the reduction and the quality of the reduced basis. A basis $\mathbf{B}$ of an m-dimensional lattice Λ has Hermite factor δ^m for $\delta \geq 1$ if $\|\mathbf{b}_1\| = \delta^m \cdot \det(\Lambda)^{1/m}$. For convenience, we call δ the *root-Hermite factor*.

Another important concept is the *Geometric Series Assumption* (GSA), introduced by Schnorr [35]. The GSA says that in a BKZ-reduced basis $\mathbf{B}$, the lengths $\|\widetilde{\mathbf{b}}_i\|$ of the Gram-Schmidt vectors decay geometrically with i, namely, $\|\widetilde{\mathbf{b}}_i\| = \|\mathbf{b}_1\| \cdot \alpha^{i-1}$ for some $0 < \alpha < 1$. Our experiments on random q-ary lattices adhere to the GSA very closely, with the exception that the Gram-Schmidt lengths are always upper- and lower-bounded by q and 1 respectively, owing to the special structure of q-ary lattices. (See the full version for details.)

[5] http://www.latticechallenge.org/svp-challenge/

By combining the notion of Hermite factor with the GSA, we can predict the lengths of *all* Gram-Schmidt vectors in a basis $\mathbf{B}$ (of an m-dimensional lattice Λ) having root-Hermite factor δ. An easy calculation shows that under the GSA,

$$\det(\Lambda) = \prod_{i=1}^{m} \|\widetilde{\mathbf{b}_i}\| = \alpha^{m(m-1)/2} \cdot \delta^{m^2} \cdot \det(\Lambda) \quad \Longrightarrow \quad \alpha = \delta^{-2m/(m-1)} \approx \delta^{-2},$$

where the approximation holds for large m.

We now turn to q-ary lattices that arise from LWE. Recall that LWE is a bounded-distance decoding problem on the m-dimensional lattice

$$\Lambda(\mathbf{A}^t) = \{\mathbf{z} \in \mathbb{Z}^m : \exists\, \mathbf{s} \in \mathbb{Z}_q^n \text{ such that } \mathbf{z} = \mathbf{A}^t\mathbf{s} \bmod q\}$$

for some $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with $m \geq n$. Because the LWE problem allows us to ignore some of the rows of $\mathbf{A}^t$ (and the corresponding noisy inner products), a natural and important question is what 'subdimension' m makes a lattice attack most effective. This question was addressed in [25], where a simple calculation showed that for a desired root-Hermite factor δ, the subdimension $m = \sqrt{n \lg(q)/\lg(\delta)}$ is optimal in the context of the natural distinguishing attack on LWE (as described at the beginning of Section 4). The analysis of [25] actually applies to the lattice

$$\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\},$$

which is the *dual* of $\Lambda(\mathbf{A}^t)$ up to scaling by a q factor, and the optimal subdimension m given above minimizes the length of $\widetilde{\mathbf{d}_1} = \mathbf{d}_1$ in a reduced basis $\mathbf{D}$ of $\Lambda^{\perp}(\mathbf{A})$ having root-Hermite factor δ. In our setting, by duality the same choice of m *maximizes* $\|\widetilde{\mathbf{b}_m}\| = q/\|\widetilde{\mathbf{d}_1}\|$, where the basis $\mathbf{B}$ of $\Lambda(\mathbf{A}^t)$ is the dual basis of $\mathbf{D}$ in *reverse* order.

In our decoding attack (and assuming the GSA), the form of the success probability given in Equation (5) as a product of $\operatorname{erf}(\cdot)$ terms also strongly indicates that we should maximize $\|\widetilde{\mathbf{b}_m}\|$, and hence use the same subdimension $m = \sqrt{n \lg(q)/\lg(\delta)}$ as above. We do not have a fully rigorous proof of this claim, since using a smaller m decreases the number of terms in the product, and hence could potentially increase the success probability. However, it seems unlikely that using a smaller m would improve the success probability by much (if at all). This is because $\|\widetilde{\mathbf{b}_m}\| = q/\|\mathbf{d}_1\|$ decreases rapidly as m decreases (see [25]), and $\|\widetilde{\mathbf{b}_{m-i}}\| \approx \|\widetilde{\mathbf{b}_m}\| \cdot \delta^{2(i-1)}$ is a very close approximation for small i, which are the Gram-Schmidt vectors that largely determine the success probability. Likewise, increasing m also appears counterproductive, since it both decreases $\|\widetilde{\mathbf{b}_m}\|$ *and* increases the number of terms in the product.

All of the above assumes that a cryptosystem exposes enough LWE samples (via its public keys and/or ciphertexts) to use the optimal subdimension. While this is always true of prior cryptosystems [33, 30, 17], it is not necessarily the case for our cryptosystem in Section 3, due to its smaller keys and ciphertexts. In this case, the adversary should use the dimension m corresponding to the actual number of published samples (this rule applies to some of our parameters sets given in Section 6).

5.2 Extrapolating BKZ Runtimes

In order to assign concrete runtimes to the attacks we put forward, we need to predict the runtime required to achieve a given root-Hermite factor δ in random q-ary lattices.

Gama and Nguyen [14] observed that on random lattices generated according to a variety of models, the runtime required to achieve a given root-Hermite factor δ in large dimensions (exceeding 200 or so) is largely determined by δ alone; the lattice dimension and determinant contribute only second-order terms. Our initial experiments confirmed this behavior for random q-ary lattices, and so we extrapolated runtimes using a fixed set of LWE parameters q and n, for a variety of values δ that correspond to sufficiently large optimal subdimensions $m = \sqrt{n \lg(q)/\lg(\delta)} \approx 200$. Our experiments were performed on a single 2.3 GHz AMD Opteron machine, using the single-precision floating-point BKZ implementation from the standard NTL library [37]. (Practical attacks on LWE for parameters beyond toy examples would require using at least quadruple precision, which would increase the running times by at least some constant factor, so our extrapolations are somewhat optimistic and hence conservative from a security point of view.)

Figure 2 shows the results of our experiments and their extrapolations. Using the rule of thumb that obtaining a 2^k approximation to the shortest vector in an m-dimensional lattice takes time $2^{\tilde{O}(m/k)}$ using BKZ, we conclude that the logarithm of the runtime should grow roughly linearly in $1/\lg(\delta)$. Our limited experiments seem consistent with this behavior, though many more would be needed to confirm it with confidence. Using least-square regression, the best linear fit to our data for $t_{\mathsf{BKZ}}(\delta) := \lg(T_{\mathsf{BKZ}}(\delta))$, the log runtime (in seconds, on our machine) of BKZ as a function of δ, is $t_{\mathsf{BKZ}}(\delta) = 1.806/\lg(\delta) - 91$. Since our experiments were limited by resources and available time, and we expect to see further improvements in basis reduction techniques (such as those in [15]), for analyzing concrete hardness we use a conservative lower bound estimate of

$$t_{\mathsf{BKZ}}(\delta) := \lg(T_{\mathsf{BKZ}}(\delta)) = 1.8/\lg(\delta) - 110. \tag{6}$$

Note that in this estimate, the 1.8 factor is very slightly smaller, and the -110 constant term is substantially smaller, than their counterparts in the best-fit function from our experiments. We chose the value 1.8 because our experiments were limited to relatively small block sizes, and the runtimes needed to achieve smaller values of δ very quickly became infeasible, so we believe that the true coefficient on the linear term (even with improved algorithms) is larger than 1.8. Similarly, our choice of -110 provides for some security margin against special-purpose hardware. In conclusion, we believe that our lower bound estimate provides some safety against foreseeable advances in algorithms and hardware, but in any case, our analysis is entirely modular and can be immediately adapted to work with any revised estimator.

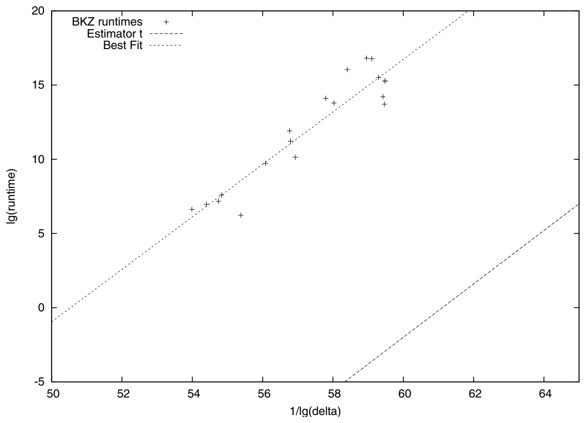


Fig. 2. Runtime of BKZ experiments on random q-ary lattices, with parameters $n = 72$, $q = 1021$, and $m = \sqrt{n \lg(q)/\lg(\delta_0)}$, i.e., the optimal subdimension with respect to a desired root-Hermite factor δ_0. The vertical axis is $t_{\mathsf{BKZ}}(\delta) := \lg(T_{\mathsf{BKZ}}(\delta))$, the logarithmic runtime required to obtain a vector with root-Hermite factor δ when running BKZ with successively increasing blocksizes. The horizontal axis is $1/\lg(\delta)$ for the actual root-Hermite factor δ achieved by the reduction. For comparison, the graph shows the best-fit estimator $t_{\mathsf{BKZ}}(\delta) = 1.086/\lg(\delta) - 91$, and our conservative lower bound estimate $t_{\mathsf{BKZ}}(\delta) = 1.8/\lg(\delta) - 110$.

6 Cryptosystem Parameters

We now estimate the concrete security of, and compute the space requirements for, the LWE-based cryptosystem from Section 3 on a variety of parameters, and compare with the example parameters given in [25] for the cryptosystem described therein (which is essentially due to [30]). Figure 3 provides the security estimates, and Figure 4 gives key and ciphertext sizes.

Instantiating the parameters. We set the cryptosystem's parameters as $n_1 = n_2 = n$ and $s_k = s_e = s$ for some positive integer n and $s > 0$, so that the two LWE hardness assumptions made in Theorem 1 are equivalent. In practice, though, distinguishing the public key and ciphertext from uniform are not equally hard, because the public key exposes fewer LWE samples than the ciphertext does. In particular, the adversary cannot use the optimal subdimension in attacking the public key, making it quite a bit harder to break. This fact could allow us to use slightly smaller s_k and correspondingly larger s_e parameters to get slightly stronger overall security, but we elect not to introduce such complications at this point. (And arguably, the secret key ought to be better-protected than any individual ciphertext.)

We choose the modulus q to be just large enough (according to the bounds in Figure 1) to allow for a Gaussian parameter $s \geq 8$, so that the discrete Gaussian $D_{\mathbb{Z}^m,s}$ approximates the continuous Gaussian D_s extremely well. Increasing the value of q beyond this threshold appears not to increase the concrete security of our cryptosystem, and (somewhat paradoxically) may even slightly decrease it!

			Adv. ε	(Distinguish)		(Decode)		
n	q	s	$\lg(\varepsilon)$	δ	lg(secs)	δ	lg(#enum)	lg(secs)
			≈ 0	*1.0065	83	1.0089	47	32
128	2053	6.77	-32	1.0115	< 0	1.0116	13	< 0
	(toy)		-64	1.0128	< 0	1.0130	1	< 0
			≈ 0	*1.0045	168	1.0067	87	78
192	4093	8.87	-32	1.0079	49	1.0083	54	42
	(low)		-64	1.0087	34	1.0091	44	29
			≈ 0	*1.0034	258	*1.0052	131	132
256	4093	8.35	-32	1.0061	96	1.0063	87	90
	(medium)		-64	1.0067	77	1.0068	73	75
			≈ 0	*1.0027	353	*1.0042	163	189
320	4093	8.00	-32	1.0049	146	1.0052	138	132
	(high)		-64	1.0054	122	1.0055	117	119
			≈ 0	1.0038	219	1.0071	82	68
136	2003	13.01	-32	1.0088	33	1.0092	42	27
	[25]		-64	1.0098	18	1.0102	27	14
			≈ 0	1.0053	126	1.0078	66	52
214	16381	7.37	-32	1.0091	28	1.0094	39	25
	[25]		-64	1.0099	17	1.0102	29	14

Fig. 3. Example parameters and attacks for the LWE-based cryptosystem described in Section 3, for various adversarial advantages. The cryptosystem parameters are $n = n_1 = n_2$, q, $s = s_k = s_e$, and message length $\ell = 128$ bits. For comparison, the last two parameter settings ($n = 136$, $n = 214$) come from the example parameters of [25]. The columns labelled "Distinguish" refer to a distinguishing (i.e., semantic security) attack. These give the root-Hermite factors δ needed to obtain the respective distinguishing advantages (over the random choice of the LWE error vector), and the corresponding logarithmic runtime (in seconds) according to our optimistic estimator from Equation (6). The columns labelled "Decode" refer to our decoding (i.e., message and randomness recovery) attack. These give example root-Hermite factors and number of NearestPlanes enumerations needed to obtain the respective decoding probability, and the corresponding estimated runtime of the attack. Other trade-offs between δ and the number of enumerations are possible (as δ increases, so does #enum); we chose the largest δ for which the estimated enumeration runtime does not exceed that of basis reduction. *An asterisk on a value of δ indicates that for reduced vectors of lengths required by the attack, the cryptosystem reveals too few LWE samples to allow an optimal choice of subdimension and corresponding root-Hermite factor δ. In such cases, we used the value of δ induced by working with the full dimension $m = n_1 + n_2 + \ell = 2n + 128$.

n	q	s	Per-User Key ($\mathbf{P}$)	Full Key ($\mathbf{P}$ & $\bar{\mathbf{A}}$)	Ciphertext ($\mathbf{c}$)	Msg Expansion
128	2053	6.77	1.8×10^5	3.6×10^5	2.8×10^3	22.0
192	4093	8.87	2.9×10^5	7.4×10^5	3.8×10^3	30.0
256	4093	8.35	4.0×10^5	11.2×10^5	4.6×10^3	36.0
320	4093	8.00	4.9×10^5	17.2×10^5	5.4×10^3	42.0
136	2003	13.01	2.8×10^6	5.8×10^6	2.9×10^3	22.6
214	16381	7.37	2.4×10^6	6.4×10^6	4.8×10^3	18.7

Fig. 4. Sizes (in bits) of public keys and ciphertexts for the cryptosystem described in Section 3; for comparison, the last two rows are for parameters given in [25]. In each case, the message size is $\ell = 128$ bits. The "message expansion" factor is the ratio of ciphertext size to plaintext size. Recall that in the ring-based system, the public key sizes are about a factor of n smaller.

This is because the BKZ runtime depends almost entirely on the root-Hermite factor δ, and by the constraints on our parameters (specifically, $s_k = s_e = s = O(\sqrt{q})$), the δ yielding a successful attack on our system grows as $q^{\Theta(1/n)}$, which increases with q (albeit very slowly).

Security estimates and conclusions. We analyze the distinguishing attack and our decoding attack (both described in Section 4), estimating the total runtimes for each of a few representative adversarial advantages. The attacks apply to a single key and ciphertext; by a standard hybrid argument, the advantage increases at most linearly in the number of ciphertexts encrypted under a single key. Our methodology for choosing the appropriate root-Hermite factor for each attack follows from the discussion in Section 4; due to space restrictions, we leave a complete description to the full version.

We highlight a few notable conclusions from our analysis:

1. The decoding attack is always at least as good as the distinguishing attack for all reasonable advantages, and is vastly superior in the high-advantage regime. As an extreme example, decoding is more than 2^{160} times faster when obtaining a large advantage against the "high security" ($n = 320$) set.
2. For the "low security" ($n = 192$) set, our key sizes are about 10 times smaller than those in [25], while offering somewhat better security, and the "high security" parameters are approximately 4-5 times smaller. Note also that the ring-based scheme has key sizes about a factor of n smaller again.
3. For the "medium security" ($n = 256$) set, the best runtime/advantage ratio is approximately 2^{120} seconds, which translates on our machine to about 2^{150} operations. It seems reasonable to conclude that these parameters currently offer security levels at least matching those of AES-128. While we elect not to give precise "symmetric bit security" claims owing to the approximate nature of our predicted runtimes, rough figures could be derived using the heuristics of Lenstra and Verheul [20].

Acknowledgments. We thank Vadim Lyubashevsky, Markus Rückert, and Michael Schneider for helpful discussions, and for pointing out irregularities in our previous security estimates. We also thank the CT-RSA reviewers for their useful comments.

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] Ajtai, M.: Generating hard instances of lattice problems. Quaderni di Matematica 13, 1–32 (2004); Preliminary version in STOC 1996

[3] Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: STOC, pp. 284–293 (1997)

[4] Ajtai, M., Dwork, C.: The first and fourth public-key cryptosystems with worst-case/average-case equivalence. Electronic Colloquium on Computational Complexity (ECCC) 14(97) (2007)

[5] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: STOC, pp. 601–610 (2001)

[6] Alekhnovich, M.: More on average case vs approximation complexity. In: FOCS, pp. 298–307 (2003)

[7] 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)

[8] Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986); Preliminary version in STACS 1985

[9] Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen 296(4), 625–635 (1993)

[10] Banaszczyk, W.: Inequalites for convex bodies and polar reciprocal lattices in R^n. Discrete & Computational Geometry 13, 217–231 (1995)

[11] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)

[12] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)

[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] Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)

[15] Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010)

[16] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)

[17] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)

[18] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)

[19] Klein, P.N.: Finding the closest lattice vector when it's unusually close. In: SODA, pp. 937–941 (2000)
[20] Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. J. Cryptology 14(4), 255–293 (2001)
[21] Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 382–400. Springer, Heidelberg (2010)
[22] 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)
[23] Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Computational Complexity 16(4), 365–411 (2007); Preliminary version in FOCS 2002
[24] Micciancio, D.: Duality in lattice cryptography. In: Public Key Cryptography (2010) (invited talk)
[25] Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post Quantum Cryptography, pp. 147–191. Springer, Heidelberg (February 2009)
[26] Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. In: STOC, pp. 351–358 (2010)
[27] Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: SODA, pp. 1468–1480 (2010)
[28] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC, pp. 333–342 (2009)
[29] Peikert, C.: An efficient and parallel gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010)
[30] Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
[31] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC, pp. 187–196 (2008)
[32] Regev, O.: New lattice-based cryptographic constructions. J. ACM 51(6), 899–942 (2004); Preliminary version in STOC 2003
[33] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), 1–40 (2009); Preliminary version in STOC 2005
[34] Rückert, M., Schneider, M.: Selecting secure parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2010/137 (2010), `http://eprint.iacr.org/`
[35] Schnorr, C.-P.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003)
[36] Schnorr, C.-P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathmatical Programming 66, 181–199 (1994)
[37] Shoup, V.: Number theory library 5.5.2 (NTL) for C++, `http://www.shoup.net/ntl/`
[38] Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)

Binary Huff Curves

Julien Devigne and Marc Joye

Technicolor, Security & Content Protection Labs
1 avenue de Belle Fontaine, 35576 Cesson-Sévigné Cedex, France
{julien.devigne,marc.joye}@technicolor.com

Abstract. This paper describes the addition law for a new form for elliptic curves over fields of characteristic 2. Specifically, it presents explicit formulæ for adding two different points and for doubling points. The case of differential point addition (that is, point addition with a known difference) is also addressed. Finally, this paper presents unified point addition formulæ; i.e., point addition formulæ that can be used for doublings. Applications to cryptographic implementations are discussed.

Keywords: Elliptic curves, Huff curves, binary fields, cryptography.

1 Introduction

Elliptic curve cryptography, introduced in the mid-eighties [10,15], is a technology of choice for the implementation of secure systems. Its main advantage is the absence of sub-exponential algorithms to solve the underlying hard problem, the elliptic curve discrete logarithm problem (ECDLP). Elliptic curve cryptosystems therefore feature smaller key sizes, which results in smaller memory and processor requirements. They are especially well suited to memory-constrained devices like smart cards.

Two types of finite fields are mainly used to implement elliptic curve cryptosystems: large prime fields and fields of characteristic 2. This paper focuses on the binary case. Further, to prevent the so-called MOV reduction [14], only non-supersingular elliptic curves are considered.

An elliptic curve over a field $\mathbb{K}$ is a smooth projective algebraic curve of genus 1 with a specified $\mathbb{K}$-rational point. More explicitly, when $\mathbb{K} = \mathbb{F}_{2^m}$, a (non-supersingular) elliptic curve can be written as the locus in the affine plane of the Weierstraß equation

$$E_{/\mathbb{F}_{2^m}} : y^2 + xy = x^3 + a_2x^2 + a_6 \qquad (a_6 \neq 0)$$

together with the extra point at infinity $\boldsymbol{O} = (0 : 1 : 0)$. It is well known that the points on an elliptic curve given by a Weierstraß equation over any field of definition form a group under the 'chord-and-tangent' addition [19, Chapter III]. The identity element is $\boldsymbol{O}$. The inverse of a point $\boldsymbol{P_0} = (x_0, y_0) \in E \setminus \{\boldsymbol{O}\}$ is given by $-\boldsymbol{P_0} = (x_0, y_0 + x_0)$. Hence, $(0, \sqrt{a_6})$ is a rational point of order 2. (Remember that square roots always exist and are unique in characteristic two.)

Now let $\boldsymbol{P_1} + \boldsymbol{P_2} = \boldsymbol{P_3}$ with $\boldsymbol{P_i} = (x_i, y_i) \in E \setminus \{\boldsymbol{O}\}$ and $\boldsymbol{P_1} \neq -\boldsymbol{P_2}$. Then

$$x_3 = \lambda^2 + \lambda + a_2 + x_1 + x_2 \quad \text{and} \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$

where

$$\begin{cases} \lambda = \dfrac{y_1 + y_2}{x_1 + x_2}, & \text{if } x_1 \neq x_2 \\ \lambda = x_1 + \dfrac{y_1}{x_1}, & \text{if } x_1 = x_2 \end{cases} .$$

There are other known models to represent elliptic curves (see e.g. [4,5]). Having different models at one's disposal is useful as this gives rise to a different arithmetic, with different properties. An expected outcome is of course an improved efficiency. This can be observed at different levels: speed, memory and processor requirements, bandwidth, etc. Another possible benefit, which should not be overlooked, is the ease of implementation. Of particular interest are the unified and complete addition laws, which reduce the number of cases to handle. Furthermore, this can help to prevent certain attacks, including those based on side-channel analysis [11] or exceptional procedure attacks [8]. Yet another application is batch computing [1].

We study in this paper a special model of elliptic curves known as *Huff's model* and generalizations thereof. More precisely, we detail the arithmetic on an extension of Huff's model to binary fields, as recently introduced in [9]. Almost all formulæ required for implementing modern discrete-log based cryptographic protocols with state-of-the-art scalar multiplication algorithms are considered: addition, doubling, special cases of addition, unified point addition formulæ, all in affine and projective versions, differential point addition. We also present a generalized form of the curve that allows for more flexibility in choosing the coefficients —in particular, every ordinary elliptic curve over $\mathbb{F}_{2^m}$ ($m \geq 4$) is birationally equivalent over $\mathbb{F}_{2^m}$ to a generalized binary Huff curve.

As a result, we obtain that (generalized) binary Huff curves can even perform better than binary Edwards curves in some cases, with the only problem that the unified formulæ do *not* really apply to all cases, and there are exceptional situations. Nevertheless, we show that one has *complete* (i.e., fully unified) addition formulæ in certain proper subgroups, which can be used for most cryptographic applications.

The rest of this paper is organized as follows. In the next section, we detail the group law on binary Huff curves. In Section 3, we provide unified addition formulæ for doubling and adding points. Section 4 presents differential point addition formulæ. In Section 5, we propose a generalized model for binary Huff curves. Finally, we conclude the paper in Section 6.

2 Binary Huff Curves

While studying a diophantine problem, Huff introduced a new model for elliptic curves [7] (see also [18]). Huff's model was recently revisited in [9]. The case of fields of odd characteristic is fully covered. For binary fields, an extended model is also proposed but no details are provided. In this section, we fill the gap. In particular, we specify the group law on binary Huff curves.

We start with the definition.

Definition 1 ([9]). *A* binary Huff curve *is the set of projective points* $(X : Y : Z) \in \mathbb{P}^2(\mathbb{F}_{2^m})$ *satisfying the equation*

$$E_{/\mathbb{F}_{2^m}} : aX(Y^2 + YZ + Z^2) = bY(X^2 + XZ + Z^2)\,, \tag{1}$$

where $a, b \in \mathbb{F}^*_{2^m}$ *and* $a \neq b$.

There are three points at infinity satisfying the curve equation, namely $(a : b : 0)$, $(1 : 0 : 0)$, and $(0 : 1 : 0)$. The affine model corresponding to the binary Huff curve given by Eq. (1) is

$$ax(y^2 + y + 1) = by(x^2 + x + 1) \ .$$

As stated in [9, § 3.4], this curve is birationally equivalent to the Weierstraß elliptic curve

$$v(v + (a + b)u) = u(u + a^2)(u + b^2)$$

under the inverse maps

$$(x, y) \leftarrow \left(\tfrac{b(u+a^2)}{v}, \tfrac{a(u+b^2)}{v+(a+b)u}\right) \quad \text{and} \quad (u, v) \leftarrow \left(\tfrac{ab}{xy}, \tfrac{ab(axy+b)}{x^2y}\right) \ . \tag{2}$$

The set of points on a binary Huff curve forms a group. The identity element is $\boldsymbol{o} = (0, 0)$. While the above maps are not line-preserving, the group law on a binary Huff curve satisfies the chord-and-tangent rule. Let $\ell_{\boldsymbol{P},\boldsymbol{Q}}$ be the line joining points $\boldsymbol{P}$ and $\boldsymbol{Q}$. This line intersects the curve in a third point (counting multiplicities) that we denote by $\boldsymbol{P} * \boldsymbol{Q}$. Let also $\ell_{\boldsymbol{R},\boldsymbol{o}}$ be the line through $\boldsymbol{R} := \boldsymbol{P} * \boldsymbol{Q}$ and $\boldsymbol{o}$. The addition of $\boldsymbol{P}$ and $\boldsymbol{Q}$ is defined as the third point of intersection of $\ell_{\boldsymbol{R},\boldsymbol{o}}$ with the curve, that is, $\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{R} * \boldsymbol{o}$. The correctness follows by observing that $\operatorname{div}(\ell_{\boldsymbol{R},\boldsymbol{o}}/\ell_{\boldsymbol{P},\boldsymbol{Q}}) = (\boldsymbol{R} * \boldsymbol{o}) + (\boldsymbol{R}) + (\boldsymbol{o}) - (\boldsymbol{R}) - (\boldsymbol{P}) - (\boldsymbol{Q}) \sim 0$, which implies $(\boldsymbol{P}) + (\boldsymbol{Q}) \sim (\boldsymbol{R} * \boldsymbol{o}) + (\boldsymbol{o})$.

Point inverse. Identity element $\boldsymbol{o}$ is not an inflection point. The inverse of a point $\boldsymbol{P}$ is therefore not defined as $\boldsymbol{P} * \boldsymbol{o}$. From the previous description, we have that $-\boldsymbol{P} = \boldsymbol{P} * \boldsymbol{A}$, where $\boldsymbol{A} = \left(\frac{b}{a+b}, \frac{a}{a+b}\right)$ is the third point of intersection of the tangent line at $\boldsymbol{o}$ with the curve.

Another way to get the inverse is to use the birational equivalence with the Weierstraß form. Let $\boldsymbol{P} = (x_1, y_1) \in E$ be a finite point. Then, whenever defined, we so obtain $-\boldsymbol{P} = (\bar{x_1}, \bar{y_1})$ with

$$\bar{x_1} = \frac{y_1(b + ax_1y_1)}{a + bx_1y_1} \quad \text{and} \quad \bar{y_1} = \frac{x_1(a + bx_1y_1)}{b + ax_1y_1} \ . \tag{3}$$

If $a + bx_1y_1 = 0$ (i.e., $\boldsymbol{P} = \left(\frac{a+b}{b}, \frac{a}{a+b}\right)$) then $-\boldsymbol{P} = (1 : 0 : 0)$. If $b + ax_1y_1 = 0$ (i.e., $\boldsymbol{P} = \left(\frac{b}{a+b}, \frac{a+b}{a}\right)$) then $-\boldsymbol{P} = (0 : 1 : 0)$.

For the points at infinity, we obtain $-(a : b : 0) = (a : b : 0)$, $-(1 : 0 : 0) = (\frac{a+b}{b}, \frac{a}{a+b})$, and $-(0 : 1 : 0) = (\frac{b}{a+b}, \frac{a+b}{a})$. Observe that $(a : b : 0)$ is of order 2.

Point doubling. Here too, the birational equivalence could be used to derive the doubling formula. The calculation, however, quickly becomes tricky. We instead directly rely on the geometric interpretation of the addition law.

Let $\boldsymbol{P} = (x_1, y_1) \in E$ be a finite point. The third point of intersection of the tangent line to the curve at $\boldsymbol{P}$ is $\boldsymbol{S} = \boldsymbol{P} * \boldsymbol{P}$. Then $[2]\boldsymbol{P} = \boldsymbol{S} * \boldsymbol{o}$. After some algebra, whenever defined, we get $[2]\boldsymbol{P} = (x_3, y_3)$ with

$$x_3 = \frac{(a+b){x_1}^2(1+y_1)^2}{b(1+x_1)^2(1+x_1y_1)^2} \quad \text{and} \quad y_3 = \frac{(a+b){y_1}^2(1+x_1)^2}{a(1+y_1)^2(1+x_1y_1)^2} \ . \tag{4}$$

If $x_1 = 1$ then $[2]\boldsymbol{P} = (1:0:0)$. If $y_1 = 1$ then $[2]\boldsymbol{P} = (0:1:0)$. If $x_1y_1 = 1$ (i.e., $\boldsymbol{P} = (\zeta, \zeta^{-1})$ with $\zeta^2 + \zeta + 1 = 0$) then $[2]\boldsymbol{P} = (a:b:0)$. (Note that $(1,1)$ is not a point on E.)

Remark 1. Since a solution to $\zeta^2 + \zeta + 1 = 0$ is a non-trivial cubic root of unity, (ζ, ζ^{-1}) is a rational point if and only the curve is defined over a binary extension field of even degree (i.e., over $\mathbb{F}_{2^m}$ with m even). Note also that (ζ, ζ^{-1}) is of order 4.

For the points at infinity, we have $[2](a:b:0) = \boldsymbol{o}$ (since $(a:b:0)$ is of order 2) and $[2](1:0:0) = [2](0:1:0) = \boldsymbol{A}$, where $\boldsymbol{A} = \left(\frac{b}{a+b}, \frac{a}{a+b}\right)$. Indeed, we have $[2](1:0:0) = -[2]\left(\frac{a+b}{b}, \frac{a}{a+b}\right) = -\left(\frac{b^3}{a^2(a+b)}, \frac{a^3}{b^2(a+b)}\right) = \left(\frac{b}{a+b}, \frac{a}{a+b}\right)$. This can also be seen from $\operatorname{div}(\ell_{\boldsymbol{o},\boldsymbol{o}}/y) = 2(\boldsymbol{o}) + (\boldsymbol{A}) - (\boldsymbol{o}) - 2(\boldsymbol{T_1}) = (\boldsymbol{o}) + (\boldsymbol{A}) - 2(\boldsymbol{T_1}) \sim 0$ and so $[2]\boldsymbol{T_1} = \boldsymbol{A}$, where $\ell_{\boldsymbol{o},\boldsymbol{o}}$ is the tangent line at $\boldsymbol{o}$, $\boldsymbol{A} = \boldsymbol{o} * \boldsymbol{o}$ and $\boldsymbol{T_1} = (1:0:0)$. Similarly, for $[2](0:1:0)$, the result follows from $\operatorname{div}(\ell_{\boldsymbol{o},\boldsymbol{o}}/x) = (\boldsymbol{o}) + (\boldsymbol{A}) - 2(\boldsymbol{T_2}) \sim 0$, where $\boldsymbol{T_2} = (0:1:0)$.

Dedicated point addition. Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2) \in E$ be two finite points with $\boldsymbol{P} \neq \boldsymbol{Q}$. As afore explained, the addition of $\boldsymbol{P}$ and $\boldsymbol{Q}$ is given by $\boldsymbol{P} + \boldsymbol{Q} = (\boldsymbol{P} * \boldsymbol{Q}) * \boldsymbol{o}$. Then, whenever defined, we obtain $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ with

$$x_3 = \frac{(x_1y_1 + x_2y_2)(1 + y_1y_2)}{(y_1 + y_2)(1 + x_1x_2y_1y_2)} \quad \text{and} \quad y_3 = \frac{(x_1y_1 + x_2y_2)(1 + x_1x_2)}{(x_1 + x_2)(1 + x_1x_2y_1y_2)} \ . \tag{5}$$

If $x_1 = x_2$ then $\boldsymbol{P} + \boldsymbol{Q} = (0:1:0)$. If $y_1 = y_2$ then $\boldsymbol{P} + \boldsymbol{Q} = (1:0:0)$.

Remark 2. When $x_1 = x_2$, we can assume that $y_1 \neq 0$ (as otherwise we would have $x_1 = x_2 = 0$ and thus $\boldsymbol{P} = \boldsymbol{Q} = \boldsymbol{o}$). We then have $(x_1, y_1) + \left(x_1, \frac{1}{y_1}\right) = (0:1:0)$. Similarly, for $x_1 \neq 0$, we have $(x_1, y_1) + \left(\frac{1}{x_1}, y_1\right) = (1:0:0)$.

Suppose now $x_1 \neq x_2$ and $y_1 \neq y_2$. If $x_1x_2y_1y_2 = 1$ and $x_1x_2 = 1$ then $\boldsymbol{P} + \boldsymbol{Q} = [2]\boldsymbol{P} + (a:b:0)$. If $x_1x_2y_1y_2 = 1$ and $x_1x_2 \neq 1$ then $\boldsymbol{P} + \boldsymbol{Q} = (a:b:0)$.

When $\boldsymbol{P} = (x_1, y_1)$ is finite and $\boldsymbol{Q}$ is at infinity, whenever defined, we have

$$\begin{cases} (x_1, y_1) + (a:b:0) = \left(\dfrac{1}{x_1}, \dfrac{1}{y_1}\right) \\ (x_1, y_1) + (1:0:0) = \left(\dfrac{a + bx_1y_1}{y_1(b + ax_1y_1)}, \dfrac{x_1(a + bx_1y_1)}{b + ax_1y_1}\right) \\ (x_1, y_1) + (0:1:0) = \left(\dfrac{y_1(b + ax_1y_1)}{a + bx_1y_1}, \dfrac{b + ax_1y_1}{x_1(a + bx_1y_1)}\right) \end{cases} \ ,$$

and similarly when $\boldsymbol{P}$ is at infinity and $\boldsymbol{Q}$ is finite. If $x_1 = 0$ or $y_1 = 0$ (i.e., $\boldsymbol{P} = \boldsymbol{o}$) then $(x_1, y_1) + \boldsymbol{Q} = \boldsymbol{Q}$. If $a + bx_1y_1 = 0$ (i.e., $\boldsymbol{P} = \left(\frac{a+b}{b}, \frac{a}{a+b}\right)$) then $(x_1, y_1) + (0 : 1 : 0) = (a : b : 0)$. If $b + ax_1y_1 = 0$ (i.e., $\boldsymbol{P} = \left(\frac{b}{a+b}, \frac{a+b}{a}\right)$) then $(x_1, y_1) + (1 : 0 : 0) = (a : b : 0)$.

We also have $(a : b : 0) + (1 : 0 : 0) = (0 : 1 : 0)$, $(a : b : 0) + (0 : 1 : 0) = (1 : 0 : 0)$, and $(1 : 0 : 0) + (0 : 1 : 0) = (\frac{a+b}{b}, \frac{a+b}{a})$.

The correctness of the addition law for the exceptional cases (i.e., when the addition formula is not defined) is easily verified. We start with the points at infinity. As before, define $\boldsymbol{A} = \left(\frac{b}{a+b}, \frac{a}{a+b}\right)$, $\boldsymbol{T_1} = (1 : 0 : 0)$, and $\boldsymbol{T_2} = (0 : 1 : 0)$. Define also $\boldsymbol{T_0} = (a : b : 0)$. From $\operatorname{div}(x) = (\boldsymbol{o}) + (\boldsymbol{T_2}) - (\boldsymbol{T_0}) - (\boldsymbol{T_1}) \sim 0$, we get $\boldsymbol{T_2} = \boldsymbol{T_0} + \boldsymbol{T_1}$, that is, $(a : b : 0) + (1 : 0 : 0) = (0 : 1 : 0)$. Since $\boldsymbol{T_0}$ is of order 2, we also get $\boldsymbol{T_1} = \boldsymbol{T_2} - \boldsymbol{T_0} = \boldsymbol{T_0} + \boldsymbol{T_2}$, that is, $(a : b : 0) + (0 : 1 : 0) = (1 : 0 : 0)$. Letting $\ell_{\boldsymbol{o},\boldsymbol{o}}$ the tangent line at $\boldsymbol{o}$, since $\boldsymbol{A} = \boldsymbol{o} * \boldsymbol{o}$, we have $\operatorname{div}(\ell_{\boldsymbol{o},\boldsymbol{o}}) = 2(\boldsymbol{o}) + (\boldsymbol{A}) - (\boldsymbol{T_0}) - (\boldsymbol{T_1}) - (\boldsymbol{T_2}) \sim 0$, which yields $\boldsymbol{T_1} + \boldsymbol{T_2} = \boldsymbol{A} - \boldsymbol{T_0} = \boldsymbol{A} + \boldsymbol{T_0}$, that is, $(1 : 0 : 0) + (0 : 1 : 0) = \left(\frac{b}{a+b}, \frac{a}{a+b}\right) + (a : b : 0) = \left(\frac{a+b}{b}, \frac{a+b}{a}\right)$. The last equality holds because for a finite point $(x_1, y_1) \neq \boldsymbol{o}$, we have $(x_1 : y_1 : 1) + (a : b : 0) = (y_1 : x_1 : x_1y_1)$, that is, $(x_1, y_1) + (a : b : 0) = \left(\frac{1}{x_1}, \frac{1}{y_1}\right)$. Furthermore, for a finite point $(x_1, y_1) \neq \boldsymbol{o}, \left(\frac{a+b}{b}, \frac{a}{a+b}\right)$, the dedicated addition formula yields $\left(\frac{a+bx_1y_1}{y_1(b+ax_1y_1)}, \frac{x_1(a+bx_1y_1)}{b+ax_1y_1}\right) - (1 : 0 : 0) = \left(\frac{a+bx_1y_1}{y_1(b+ax_1y_1)}, \frac{x_1(a+bx_1y_1)}{b+ax_1y_1}\right) + \left(\frac{a+b}{b}, \frac{a}{a+b}\right) = (x_1, y_1)$, that is, $(x_1, y_1) + (1 : 0 : 0) = \left(\frac{a+bx_1y_1}{y_1(b+ax_1y_1)}, \frac{x_1(a+bx_1y_1)}{b+ax_1y_1}\right)$. The relation $(x_1, y_1) + (0 : 1 : 0)$, for a finite point $(x_1, y_1) \neq \boldsymbol{o}, \left(\frac{b}{a+b}, \frac{a+b}{a}\right)$, is obtained from $(x_1, y_1) + (a : b : 0) + (1 : 0 : 0) = \left(\frac{1}{x_1}, \frac{1}{y_1}\right) + (1 : 0 : 0) = \left(\frac{y_1(b+ax_1y_1)}{a+bx_1y_1}, \frac{b+ax_1y_1}{x_1(a+bx_1y_1)}\right)$. If $(x_1, y_1) = \left(\frac{a+b}{b}, \frac{a}{a+b}\right)$ then $(x_1, y_1) + (1 : 0 : 0) = (a : b : 0)$ since $(x_1, y_1) = -\boldsymbol{T_2} = \boldsymbol{T_0} - \boldsymbol{T_1}$. If $(x_1, y_1) = \left(\frac{b}{a+b}, \frac{a+b}{a}\right)$ then $(x_1, y_1) + (0 : 1 : 0) = (a : b : 0)$ since $(x_1, y_1) = -\boldsymbol{T_1} = \boldsymbol{T_0} - \boldsymbol{T_2}$.

The next cases consider finite points. Suppose that $x_1 = x_2$, that is, $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_1, y_2)$. We can write $\operatorname{div}((x - x_1)/x) = (\boldsymbol{P}) + (\boldsymbol{Q}) + (\boldsymbol{T_2}) - (\boldsymbol{o}) - 2(\boldsymbol{T_2}) = (\boldsymbol{P}) + (\boldsymbol{Q}) - (\boldsymbol{T_2}) - (\boldsymbol{o}) \sim 0$. Therefore, we get $(x_1, y_1) + (x_1, y_2) = (0 : 1 : 0)$. The case $y_1 = y_2$ is similar by considering $\operatorname{div}((y - y_1)/y)$. Lastly, suppose that $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2)$ with $x_1 \neq x_2$ and $y_1 \neq y_2$. If $x_1x_2y_1y_2 = 1$ and $x_1x_2 = 1$ then $y_1y_2 = 1$; we thus have $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = \left(\frac{1}{x_1}, \frac{1}{y_1}\right)$. (Note that $x_1x_2 = 1$ implies $\boldsymbol{P} \neq \boldsymbol{o}$.) As already shown, we then have $\boldsymbol{Q} = \left(\frac{1}{x_1}, \frac{1}{y_1}\right) = (x_1, y_1) + (a : b : 0)$. We so obtain $\boldsymbol{P} + \boldsymbol{Q} = (x_1, y_1) + \left(\frac{1}{x_1}, \frac{1}{y_1}\right) = [2]\boldsymbol{P} + (a : b : 0)$. If $x_1x_2y_1y_2 = 1$ and $x_1x_2 \neq 1$ then $y_1y_2 \neq 1$ and also $x_1y_1 \neq x_2y_2$. Indeed, $x_1y_1 = x_2y_2$ and $x_1x_2y_1y_2 = 1$ would imply $x_1y_1 = x_2y_2 = 1$, that is, $\boldsymbol{P} = \left(x_1, \frac{1}{x_1}\right)$ and $\boldsymbol{Q} = \left(x_2, \frac{1}{x_2}\right)$. This in turn would imply $x_1 = \zeta$ and $x_2 = \zeta^{-1}$ (since $\boldsymbol{P} \neq \boldsymbol{Q}$) and so $x_1x_2 = 1$, a contradiction. Consequently, if $x_1x_2y_1y_2 = 1$ and $x_1x_2 \neq 1$, the result follows since then $(x_1, y_1) + (x_2, y_2) = \big((x_1y_1 + x_2y_2)(1 + y_1y_2)(x_1 + x_2) : (x_1y_1 + x_2y_2)(1 + x_1x_2)(y_1 + y_2) : 0\big) = (a : b : 0)$, provided that $x_1 \neq x_2$ and $y_1 \neq y_2$.

Projective formulæ. We now present the projective version of the addition formulæ. It is useful to introduce some notation. When analyzing the performance, we will let M and D respectively denote the cost of a multiplication and of a multiplication by a constant, in $\mathbb{F}_{2^m}$. The cost of additions and squarings in $\mathbb{F}_{2^m}$ will be neglected.

For the point doubling (Eq. (4)) of $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$, we get

$$\begin{cases} X_3 = \alpha \cdot {X_1}^2 {Z_1}^2 (Y_1 + Z_1)^4 \\ Y_3 = \beta \cdot {Y_1}^2 {Z_1}^2 (X_1 + Z_1)^4 \\ Z_3 = (X_1 Y_1 + {Z_1}^2)^2 (X_1 + Z_1)^2 (Y_1 + Z_1)^2 \end{cases} ,$$

where $\alpha = \frac{a+b}{b}$ and $\beta = \frac{a+b}{a}$. In more detail, this can be evaluated as

$$\begin{gathered} m_1 = X_1 Y_1 + {Z_1}^2, \quad m_2 = X_1 Z_1, \quad m_3 = Y_1 Z_1, \\ X_3 = \alpha \cdot [m_2 (Y_1 + Z_1)^2]^2, \quad Y_3 = \beta \cdot [m_3 (X_1 + Z_1)^2]^2, \\ Z_3 = [m_1 (m_1 + m_2 + m_3)]^2, \end{gathered}$$

that is, with $6\mathsf{M} + 2\mathsf{D}$ (here $2\mathsf{D}$ represents the cost of the two multiplications by constants α and β).

For the dedicated point addition (Eq. (5)) of $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$, we get

$$\begin{cases} X_3 = (X_1 Y_1 {Z_2}^2 + X_2 Y_2 {Z_1}^2)(Y_1 Y_2 + Z_1 Z_2)(X_1 Z_2 + X_2 Z_1) \\ Y_3 = (X_1 Y_1 {Z_2}^2 + X_2 Y_2 {Z_1}^2)(X_1 X_2 + Z_1 Z_2)(Y_1 Z_2 + Y_2 Z_1) \\ Z_3 = (X_1 Z_2 + X_2 Z_1)(Y_1 Z_2 + Y_2 Z_1)(X_1 X_2 Y_1 Y_2 + {Z_1}^2 {Z_2}^2) \end{cases} .$$

This can be evaluated with $15\mathsf{M}$ as

$$\begin{gathered} m_1 = X_1 X_2, \quad m_2 = Y_1 Y_2, \quad m_3 = Z_1 Z_2, \\ m_4 = (X_1 + Z_1)(X_2 + Z_2) + m_1 + m_3, \quad m_5 = (Y_1 + Z_1)(Y_2 + Z_2) + m_2 + m_3, \\ m_6 = m_4 (m_2 + m_3), \quad m_7 = m_5 (m_1 + m_3), \quad m_8 = m_1 m_2 + {m_3}^2, \\ m_9 = m_8 + (X_1 Y_1 + {Z_1}^2)(X_2 Y_2 + {Z_2}^2), \\ X_3 = m_6 m_9, \quad Y_3 = m_7 m_9, \quad Z_3 = m_4 m_5 m_8 \ . \end{gathered}$$

The cost can be reduced to $14\mathsf{M}$ with extended coordinates (X_i, Y_i, Z_i, T_i) where $T_i = X_i Y_i$ $(i = 1, 2, 3)$:

$$\begin{gathered} m_1 = X_1 X_2, \quad m_2 = Y_1 Y_2, \quad m_3 = Z_1 Z_2, \\ m_4 = (X_1 + Z_1)(X_2 + Z_2) + m_1 + m_3, \quad m_5 = (Y_1 + Z_1)(Y_2 + Z_2) + m_2 + m_3, \\ m_6 = m_4 (m_2 + m_3), \quad m_7 = m_5 (m_1 + m_3), \quad m_8 = m_1 m_2 + {m_3}^2, \\ m_9 = m_8 + (T_1 + {Z_1}^2)(T_2 + {Z_2}^2), \\ X_3 = m_6 m_9, \quad Y_3 = m_7 m_9, \quad Z_3 = m_4 m_5 m_8, \quad T_3 = X_3 Y_3 \ . \end{gathered}$$

3 Unified Point Addition

The formulæ presented in the previous section for evaluating $\boldsymbol{P} + \boldsymbol{Q}$ distinguish two cases: $\boldsymbol{P} = \boldsymbol{Q}$ (doubling) and $\boldsymbol{P} \neq \boldsymbol{Q}$ (dedicated addition). In this section, we develop addition formulæ that can be used in both cases. The corresponding operation is referred to as *unified point addition*.

The starting point is our formula for the dedicated point addition. Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2) \in E$ be two finite points, with $\boldsymbol{P} \neq \boldsymbol{Q}$. Equation (5) says that, whenever defined, $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ where

$$x_3 = \frac{(x_1y_1 + x_2y_2)(1 + y_1y_2)}{(y_1 + y_2)(1 + x_1x_2y_1y_2)} \quad \text{and} \quad y_3 = \frac{(x_1y_1 + x_2y_2)(1 + x_1x_2)}{(x_1 + x_2)(1 + x_1x_2y_1y_2)} \ .$$

The point addition is defined up to the curve equation. Using the additional relations $ax_i({y_i}^2 + y_i + 1) = by_i({x_i}^2 + x_i + 1)$, $i \in \{1, 2\}$, we get after a lengthy and tedious calculation:

$$\begin{cases} x_3 = \dfrac{b(x_1 + x_2)(1 + x_1x_2y_1y_2) + (a + b)x_1x_2(1 + y_1y_2)}{b(1 + x_1x_2)(1 + x_1x_2y_1y_2)} \\ y_3 = \dfrac{a(y_1 + y_2)(1 + x_1x_2y_1y_2) + (a + b)y_1y_2(1 + x_1x_2)}{a(1 + y_1y_2)(1 + x_1x_2y_1y_2)} \end{cases} . \tag{6}$$

The following Sage script [21] verifies that the two formulations for x_3 are equivalent. The equivalence for y_3 follows by symmetry.

```
R.<a,b,x1,y1,x2,y2>=GF(2)[]
S=R.quotient([
  a*x1*(y1^2+y1+1)+b*y1*(x1^2+x1+1),
  a*x2*(y2^2+y2+1)+b*y2*(x2^2+x2+1)
])
verif = b*(x1*y1+x2*y2)*(1+x1*x2)*(1+y1*y2)+
      (y1+y2)*((a+b)*x1*x2*(1+y1*y2)+b*(x1+x2)*(1+x1*x2*y1*y2))
0 == S(verif)
```

Remarkably, this new expression works for doubling a point $\boldsymbol{P} = (x_1, y_1)$. Indeed, if we replace (x_2, y_2) with (x_1, y_1) in Eq. (6), we obtain

$$x_3 = \frac{(a + b){x_1}^2(1 + {y_1}^2)}{b(1 + {x_1}^2)(1 + {x_1}^2{y_1}^2)} \quad \text{and} \quad y_3 = \frac{(a + b){y_1}^2(1 + {x_1}^2)}{a(1 + {y_1}^2)(1 + {x_1}^2{y_1}^2)} ,$$

namely, our previous doubling formula (Eq. (4)).

The unified addition law given by Eq. (6) is defined when the denominators $b(1 + x_1x_2)(1 + x_1x_2y_1y_2)$ and $a(1 + y_1y_2)(1 + x_1x_2y_1y_2)$ are non-zero. If $x_1x_2y_1y_2 = 1$, $x_1x_2 \neq 1$ and $y_1y_2 \neq 1$ then $\boldsymbol{P} + \boldsymbol{Q} = (a : b : 0)$. If $x_1x_2 = 1$ or $y_1y_2 = 1$ — namely, $\boldsymbol{P} = (x_1, y_1)$ ($\neq \boldsymbol{o}$) and $\boldsymbol{Q} \in \left\{\left(\frac{1}{x_1}, y_1\right), \left(x_1, \frac{1}{y_1}\right), \left(\frac{1}{x_1}, \frac{1}{y_1}\right)\right\}$, then

$$\boldsymbol{P} + \boldsymbol{Q} = \begin{cases} (1 : 0 : 0) & \text{if } x_1x_2 = 1 \text{ and } y_1 = y_2 \\ (0 : 1 : 0) & \text{if } y_1y_2 = 1 \text{ and } x_1 = x_2 \\ \left(\frac{b(1+{x_1}^2)(1+{x_1}^2{y_1}^2)}{(a+b){x_1}^2(1+{y_1}^2)}, \frac{a(1+{y_1}^2)(1+{x_1}^2{y_1}^2)}{(a+b){y_1}^2(1+{x_1}^2)}\right) & \text{otherwise} \end{cases} .$$

Proof. Observe that since $\boldsymbol{Q} = (x_2, y_2)$ belongs to the curve so do $\big(\frac{1}{x_2}, y_2\big)$, $\big(x_2, \frac{1}{y_2}\big)$, and $\big(\frac{1}{x_2}, \frac{1}{y_2}\big)$. Suppose first that $x_1x_2 = 1 \iff x_2 = \frac{1}{x_1}$. Now using the fact that (x_1, y_1) and $\big(\frac{1}{x_1}, y_2\big)$ are on the curve implies that $(y_1 + y_2)(y_1 + y_2 + 1) = (y_1 + y_2)\big(y_1 + 1 + \frac{1}{y_1}\big) \iff (y_1 + y_2)(\frac{1}{y_1} + y_2) = 0$. This means that $(x_2, y_2) \in \big\{\big(\frac{1}{x_1}, y_1\big), \big(\frac{1}{x_1}, \frac{1}{y_1}\big)\big\}$.

Analogously, it can be shown that $y_1y_2 = 1$ implies $(x_1 + x_2)\big(\frac{1}{x_1} + x_2\big) = 0$, that is, $(x_2, y_2) \in \big\{\big(x_1, \frac{1}{y_1}\big), \big(\frac{1}{x_1}, \frac{1}{y_1}\big)\big\}$. □

The cases where the points $\boldsymbol{P}$ and/or $\boldsymbol{Q}$ are at infinity are detailed in the previous section.

To sum up, noting that the case $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} \in \big\{\big(\frac{1}{x_1}, y_1\big), \big(x_1, \frac{1}{y_1}\big), \big(\frac{1}{x_1}, \frac{1}{y_1}\big)\big\}$ corresponds to $\boldsymbol{P}+\boldsymbol{Q} = (1:0:0)$, $\boldsymbol{P}+\boldsymbol{Q} = (0:1:0)$ or $\boldsymbol{Q} = \boldsymbol{P} + (a:b:0)$, we see that the exceptional cases always involve the points at infinity. This leads to the following result.

Proposition 1. *Let E be a binary Huff curve over $\mathbb{F}_{2^m}$ and let $\mathbb{G} \subset E(\mathbb{F}_{2^m})$ be a subgroup such that $(a:b:0), (1:0:0), (0:1:0) \notin \mathbb{G}$. Then the unified addition formula given by Eq.* (6) *is complete.* □

In particular, the addition formula is complete in a subgroup of odd order, provided that $(1:0:0)$ and $(0:1:0)$ are both of even order.

Projective version. Here we give the projective version of Eq. (6). For $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$, we get $\boldsymbol{P}+\boldsymbol{Q} = (X_3 : Y_3 : Z_3)$ with

$$\begin{cases} X_3 = (Z_1Z_2 + Y_1Y_2)\big((X_1Z_2 + X_2Z_1)({Z_1}^2{Z_2}^2 + X_1X_2Y_1Y_2) + \\ \qquad\qquad \alpha X_1X_2Z_1Z_2(Z_1Z_2 + Y_1Y_2)\big) \\ Y_3 = (Z_1Z_2 + X_1X_2)\big((Y_1Z_2 + Y_2Z_1)({Z_1}^2{Z_2}^2 + X_1X_2Y_1Y_2) + \\ \qquad\qquad \beta Y_1Y_2Z_1Z_2(Z_1Z_2 + X_1X_2)\big) \\ Z_3 = (Z_1Z_2 + X_1X_2)(Z_1Z_2 + Y_1Y_2)({Z_1}^2{Z_2}^2 + X_1X_2Y_1Y_2) \end{cases} \tag{7}$$

where $\alpha = \frac{a+b}{b}$ and $\beta = \frac{a+b}{a}$. This can be evaluated as

$$\begin{gathered} m_1 = X_1X_2, \quad m_2 = Y_1Y_2, \quad m_3 = Z_1Z_2, \\ m_4 = (X_1 + Z_1)(X_2 + Z_2) + m_1 + m_3, \quad m_5 = (Y_1 + Z_1)(Y_2 + Z_2) + m_2 + m_3 \\ m_6 = m_1m_3, \quad m_7 = m_2m_3, \quad m_8 = m_1m_2 + {m_3}^2, \\ m_9 = m_6(m_2 + m_3)^2, \quad m_{10} = m_7(m_1 + m_3)^2, \\ m_{11} = m_8(m_2 + m_3), \quad m_{12} = m_8(m_1 + m_3), \\ X_3 = m_4m_{11} + \alpha \cdot m_9, \quad Y_3 = m_5m_{12} + \beta \cdot m_{10}, Z_3 = m_{11}(m_1 + m_3), \end{gathered}$$

that is, with $15\mathsf{M} + 2\mathsf{D}$ (again $2\mathsf{D}$ represents the cost of the two multiplications by constants α and β).

More formulæ. There are other unified addition formulæ similar to Eq. (6). For instance, whenever defined, we also have

$$\begin{cases} x_3 = \dfrac{(1+y_1y_2)\big(b(x_1+x_2)+x_1x_2(a+b+ay_1+ay_2)\big)}{b(1+x_1x_2)(1+x_1x_2y_1y_2)} \\ y_3 = \dfrac{(1+x_1x_2)\big(a(y_1+y_2)+y_1y_2(a+b+bx_1+bx_2)\big)}{a(1+y_1y_2)(1+x_1x_2y_1y_2)} \end{cases} .$$

Alternate unified formulæ can be obtained by selecting another neutral element. This results in translating the group law. For instance, defining $\boldsymbol{o}' = (a : b : 0)$ as the neutral element yields, whenever defined, the following unified point addition formula:

$$\begin{cases} x_3 &= \dfrac{b(1+x_1x_2)(1+x_1x_2y_1y_2)}{b(x_1+x_2)(1+x_1x_2y_1y_2)+(a+b)x_1x_2(1+y_1y_2)} \\ &= \dfrac{b(1+x_1x_2)^2(1+x_1x_2y_1y_2)y_1y_2}{\big[b(x_1+x_2)(1+x_1x_2y_1y_2)+(a+b)x_1x_2(1+y_1y_2)\big](1+x_1x_2)y_1y_2} \\ y_3 &= \dfrac{a(1+y_1y_2)(1+x_1x_2y_1y_2)}{a(y_1+y_2)(1+x_1x_2y_1y_2)+(a+b)y_1y_2(1+x_1x_2)} \\ &= \dfrac{a(1+y_1y_2)^2(1+x_1x_2y_1y_2)x_1x_2}{\big[b(x_1+x_2)(1+x_1x_2y_1y_2)+(a+b)x_1x_2(1+y_1y_2)\big](1+x_1x_2)y_1y_2} \end{cases} .$$

Advantageously, the corresponding projective version can be evaluated with $13\mathsf{M}+2\mathsf{D}$ as

$$\begin{gathered} m_1 = X_1X_2, \quad m_2 = Y_1Y_2, \quad m_3 = Z_1Z_2, \quad m_4 = m_1m_2, \\ m_5 = m_2m_3, \quad m_6 = (X_1+Z_1)(X_2+Z_2)+m_1+m_3 \\ X_3 = (m_1+m_3)(m_4+m_5)({m_3}^2+m_4), \quad Y_3 = \rho m_1(m_2+m_3)^2({m_3}^2+m_4), \\ Z_3 = (m_4+m_5)\big[(\alpha m_1({m_3}^2+m_5)+m_6(m_4+{m_3}^2)\big], \end{gathered}$$

where $\alpha = \frac{a+b}{b}$ and $\rho = \frac{a}{b}$.

4 Differential Point Addition

The so-called Montgomery representation was developed in [16] in order to speed up the implementation of the elliptic curve factoring method (ECM) [12]. It was subsequently adapted to (ordinary) Weierstraß elliptic curves over binary fields in [13,20] (see also [6]). The idea stems from the observation that the x-coordinate of the sum of two points can be evaluated from two x-coordinates of the input points and the x-coordinate of their difference. More generally, a *differential point addition* consists in calculating $w(\boldsymbol{P}+\boldsymbol{Q})$ from $w(\boldsymbol{P})$, $w(\boldsymbol{Q})$ and $w(\boldsymbol{Q}-\boldsymbol{P})$, for a certain coordinate function w. When such an operation is available, the value of $w([k]\boldsymbol{P})$ can be efficiently computed from the binary expansion of scalar k, $k = \sum_{i=0}^{\ell-1} k_i\, 2^i$ with $k_i \in \{0,1\}$ and $k_{\ell-1} = 1$. Defining $\kappa_j = \sum_{i=j}^{\ell-1} k_i\, 2^i$, $\boldsymbol{P_j} = [\kappa_j]\boldsymbol{P}$, and $\boldsymbol{Q_j} = \boldsymbol{P_j} + \boldsymbol{P}$, we have

$$\big(w(\boldsymbol{P_{\ell-1}}), w(\boldsymbol{Q_{\ell-1}})\big) = \big(w(\boldsymbol{P}), w(\boldsymbol{P}+\boldsymbol{P})\big)$$

and

$$\big(w(\boldsymbol{P_j}), w(\boldsymbol{Q_j})\big) = \begin{cases} \big(w(\boldsymbol{P_{j+1}} + \boldsymbol{P_{j+1}}), w(\boldsymbol{P_{j+1}} + \boldsymbol{Q_{j+1}})\big) & \text{if } k_j = 0 \\ \big(w(\boldsymbol{P_{j+1}} + \boldsymbol{Q_{j+1}}), w(\boldsymbol{Q_{j+1}} + \boldsymbol{Q_{j+1}})\big) & \text{if } k_j = 1 \end{cases},$$

for $j = \ell - 2, \ldots, 0$. Remarking that $\kappa_0 = k$, the value of $w([k]\boldsymbol{P})$ is obtained at the end of the recursion as $w(\boldsymbol{P_0})$.

As shown in Section 2, the inverse of a point $\boldsymbol{P} = (x_1, y_1)$ on a binary Huff curve is given by $-\boldsymbol{P} = (\bar{x_1}, \bar{y_1})$ with $\bar{x_1} = \frac{y_1(b+ax_1y_1)}{a+bx_1y_1}$ and $\bar{y_1} = \frac{x_1(a+bx_1y_1)}{b+ax_1y_1}$. A natural choice for coordinate function $w : \boldsymbol{P} \mapsto w(\boldsymbol{P})$ is therefore to define it as the product of the x- and y-coordinates, or as a function thereof. Doing so, we see that $w(\boldsymbol{P}) = w(-\boldsymbol{P})$.

Specifically, for a finite point $\boldsymbol{P} = (x_1, y_1)$, we define $w(\boldsymbol{P}) = x_1y_1$. For the points at infinity, we define $w(1 : 0 : 0) = \frac{a}{b}$, $w(0 : 1 : 0) = \frac{b}{a}$, and $w(a : b : 0) =$ "∞" $= (1 : 0)$. Hence, for the differential doubling, we immediately obtain from Eq. (4),

$$w([2]\boldsymbol{P}) = \frac{\gamma \cdot {w_1}^2}{(1 + w_1)^4} \quad \text{with } \gamma = \tfrac{(a+b)^2}{ab}\,, \tag{8}$$

and where $w_1 = w(\boldsymbol{P})$, provided that $w_1 \neq 1$. If $w_1 = 1$ then $w([2]\boldsymbol{P}) = (1 : 0)$.

Let $\boldsymbol{Q}$ be a second point, different from $\boldsymbol{P}$. We let w_1, w_2, and $\bar{w}$ denote the w-coordinate of $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{Q} - \boldsymbol{P}$, respectively. In principle, it could be possible to derive the formula for the differential addition from Eq. (5). A much simpler way is to rely on the connection between our choice of the w-coordinate and the birational map with the Weierstraß equation; cf. Eq. (2).

Montgomery representation comes in two flavors: additive method and multiplicative method. From the additive formula in [13, Lemma 1] (slightly generalized), whenever defined, we get

$$w(\boldsymbol{P} + \boldsymbol{Q}) = \frac{\bar{w} \cdot (w_1 + w_2)^2}{(w_1 + w_2)^2 + (\gamma\bar{w}) \cdot w_1w_2} \quad \text{with } \gamma = \tfrac{(a+b)^2}{ab}\ .$$

An application of the multiplicative formula in [20, § 3.2] yields after simplification

$$w(\boldsymbol{P} + \boldsymbol{Q}) = \frac{(w_1 + w_2)^2}{\bar{w} \cdot (1 + w_1w_2)^2}\,, \tag{9}$$

provided that $w_1w_2 \neq 1$. (Note that $\bar{w} \neq 0$ since $\boldsymbol{P} \neq \boldsymbol{Q}$.) The case $w_1w_2 = 1$ corresponds to the case $x_1x_2y_1y_2 = 1$; see Section 2. If $w_1w_2 = 1$ then $w(\boldsymbol{P} + \boldsymbol{Q}) = (1 : 0)$.

Projective version. To have projective formulæ, we represent the w-coordinate of a point $\boldsymbol{P}$ by the pair $(W : Z) = (\theta\, w(\boldsymbol{P}) : \theta)$ if $\boldsymbol{P} \neq (a : b : 0)$, and $(W : Z) = (\theta : 0)$ if $\boldsymbol{P} = (a : b : 0)$, for some non-zero $\theta \in \mathbb{F}_{2^m}$. Letting $w_i = (W_i : Z_i)$ for $i = 1, 2$, and $\bar{w} = (\bar{W} : \bar{Z})$, we obtain from Eqs (8) and (9)

$$\begin{cases} W([2]\boldsymbol{P}) = \gamma \cdot (W_1Z_1)^2 \\ Z([2]\boldsymbol{P}) = (W_1 + Z_1)^4 \end{cases}$$

and

$$\begin{cases} W(\boldsymbol{P}+\boldsymbol{Q}) = \bar{Z}(W_1Z_2 + W_2Z_1)^2 \\ Z(\boldsymbol{P}+\boldsymbol{Q}) = \bar{W}(W_1W_2 + Z_1Z_2)^2 \end{cases} .$$

The differential point doubling requires $1\mathsf{M} + 1\mathsf{D}$ (where D is the cost of a multiplication by γ) and, by computing $W_1Z_2 + W_2Z_1$ as $(W_1 + Z_1)(W_2 + Z_2) + (W_1W_2 + Z_1Z_2)$, the differential point addition requires $5\mathsf{M}$. The cost of the differential point addition reduces to $4\mathsf{M}$ when $\bar{Z} = 1$. Using the above scalar multiplication algorithm, the computation of $w([k]\boldsymbol{P})$ can therefore be evaluated with $5\mathsf{M} + 1\mathsf{D}$ per bit of scalar k. Furthermore, we note that over a binary field of even extension (i.e., over $\mathbb{F}_{2^m}$ with m even), selecting the ratio $a/b = \zeta$ (with $\zeta^2 + \zeta + 1 = 0$) leads to $\gamma = 1$ and so reduces the cost of the differential point doubling to $1\mathsf{M}$.

5 Generalized Binary Huff Curves

The transformations given by Eq. (2) show how to express any binary Huff curve as a Weierstraß curve. The reverse direction is however not always possible. Not all ordinary elliptic curves over $\mathbb{F}_{2^m}$ can be expressed in the Huff form as defined by Eq. (1). Worse, none of the NIST-recommended curves [17] can be written in this model. In this section, we generalize the definition of binary Huff curves to cover all isomorphism classes of ordinary curves over $\mathbb{F}_{2^m}$, for $m \geq 4$.

In [9, Section 3], Huff's model is generalized to $ax\mathcal{P}(y) = by\mathcal{P}(x)$ for some monic polynomial $\mathcal{P} \in \mathbb{F}_{2^m}[t]$, of degree 2, with non-zero discriminant, and such that $\mathcal{P}(0) \neq 0$. Binary Huff curves correspond to the choice $\mathcal{P}(t) = t^2 + t + 1$. We consider below a more general polynomial, namely, $\mathcal{P}(t) = t^2 + ft + 1$ with $f \neq 0$.

Definition 2. *A* generalized binary Huff curve *is the locus in* $\mathbb{P}^2(\mathbb{F}_{2^m})$ *of the equation*

$$aX(Y^2 + fYZ + Z^2) = bY(X^2 + fXZ + Z^2)\,, \tag{10}$$

where $a, b, f \in \mathbb{F}_{2^m}^*$ *and* $a \neq b$.

Remark 3. There are other suitable generalizations of binary Huff curves, like $\mathcal{P}(t) = t^2 + t + e$ with $e \neq 0$. We selected $\mathcal{P}(t) = t^2 + ft + 1$ since the arithmetic looked slightly simpler. Note also that polynomial $\mathcal{P}(t) = t^2 + ft + e$ (with $e, f \neq 0$) is not more general since the change of variables $(x, y) \leftarrow (\sqrt{e}\,\bar{x}, \sqrt{e}\,\bar{y})$ transforms the equation $ax(y^2 + fy + e) = by(x^2 + fx + e)$ into $a\bar{x}(\bar{y}^2 + f'\bar{y} + 1) = b\bar{y}(\bar{x}^2 + f'\bar{x} + 1)$ where $f' = f/\sqrt{e}$.

The affine model of Eq. (10) is $ax(y^2 + fy + 1) = by(x^2 + fx + 1)$. It is birationally equivalent to the Weierstraß elliptic curve

$$v(v + (a + b)fu) = u(u + a^2)(u + b^2)$$

under the inverse maps

$$(x, y) \leftarrow \left(\tfrac{b(u+a^2)}{v}, \tfrac{a(u+b^2)}{v+(a+b)fu}\right) \quad \text{and} \quad (u, v) \leftarrow \left(\tfrac{ab}{xy}, \tfrac{ab(axy+b)}{x^2y}\right) .$$

The points at infinity are $(a : b : 0)$, $(1 : 0 : 0)$ and $(0 : 1 : 0)$. Point $\boldsymbol{A}$, namely the point of intersection of the tangent line at $\boldsymbol{o}$ with the curve, becomes $\boldsymbol{A} = \left(\frac{bf}{a+b}, \frac{af}{a+b}\right)$.

Before stating the universality of the generalized model, we summarize some elementary results on binary fields that are needed to understand the proof. We let Tr denote the trace function defined as the linear function given by $\mathrm{Tr} : \mathbb{F}_{2^m} \to \mathbb{F}_2, \theta \mapsto \sum_{j=0}^{m-1} \theta^{2^j}$. We recall that the quadratic equation $x^2+Ax+B=0$ with $A \neq 0$ has a solution in $\mathbb{F}_{2^m}$ if and only if $\mathrm{Tr}(B/A^2) = 0$. If x_0 is a solution then the other solution is $x_0 + A$. Finally, it is easy to see that $\mathrm{Tr}(A^2) = \mathrm{Tr}(A)$ for $A \in \mathbb{F}_{2^m}$.

Proposition 2. *Let $m \geq 4$. Each ordinary elliptic curve over $\mathbb{F}_{2^m}$ is birationally equivalent over $\mathbb{F}_{2^m}$ to a generalized binary Huff curve.*

Proof. Each ordinary elliptic curve over $\mathbb{F}_{2^m}$ is isomorphic to $v^2 + uv = u^3 + a_2u^2 + a_6$ for some $a_2, a_6 \in \mathbb{F}_{2^m}$, $a_6 \neq 0$. Further, from [19, Proposition 3.1(b) and Table 1.2], the Weierstraß curves $v^2 + uv = u^3 + a_2u^2 + a_6$ and $v(v + (a+b)fu) = u(u+a^2)(u+b^2)$ are isomorphic under the admissible change of variables $(u, v) \leftarrow (\mu^2 u, \mu^3(v + su + \sqrt{a_6}))$ with $\mu = (a+b)f$ if and only if the curve parameters satisfy

$$s^2 + s + a_2 + f^{-2} = 0 \quad \text{and} \quad (a+b)^4 f^4 \sqrt{a_6} = a^2b^2$$

for some s. The equation $s^2 + s + a_2 + f^{-2} = 0$ has a solution $s \in \mathbb{F}_{2^m}$ if and only if $\mathrm{Tr}(a_2 + f^{-2}) = 0 \iff \mathrm{Tr}(f^{-1}) = \mathrm{Tr}(a_2)$. Dividing the second equation by b^4 and letting $t = \frac{a^2}{b^2}$ yields $t^2 + \frac{1}{f^4\sqrt{a_6}}t + 1 = 0$, which has a solution $t \in \mathbb{F}_{2^m}$ if and only if $\mathrm{Tr}(f^8 a_6) = \mathrm{Tr}(f\sqrt[8]{a_6}) = 0$. Consequently, given an ordinary binary elliptic curve $v^2 + uv = u^3 + a_2u^2 + a_6$, one can obtain an isomorphic curve $v(v + (a+b)fu) = u(u+a^2)(u+b^2)$ — and so the birationally equivalent generalized binary Huff curve $ax(y^2 + fy + 1) = by(x^2 + fx + 1)$, by choosing parameter f such that $\mathrm{Tr}(f^{-1}) = \mathrm{Tr}(a_2)$ and $\mathrm{Tr}(f\sqrt[8]{a_6}) = 0$, and parameters a and b such that $\frac{a^2}{b^2}$ is a solution to $t^2 + \frac{1}{f^4\sqrt{a_6}}t + 1 = 0$.

It remains to show that such an f always exists. The proof proceeds analogously to the one offered in [2, Theorem 4.3]. Fix $a_2 \in \mathbb{F}_{2^m}$ and $a_6 \in \mathbb{F}_{2^m}^*$. For each $\delta, \epsilon \in \mathbb{F}_2$, define

$$D_{\delta,\epsilon} = \{f \in \mathbb{F}_{2^m}^* : \mathrm{Tr}(f^{-1}) = \delta, \mathrm{Tr}(f\sqrt[8]{a_6}) = \epsilon\} \ .$$

We have to show that the set $D_{\mathrm{Tr}(a_2),0}$ is non-empty.

We have $\#D_{0,0} + \#D_{1,0} = 2^{m-1} - 1$. Indeed, as f runs through $\mathbb{F}_{2^m}^*$, so does $f\sqrt[8]{a_6}$. Hence, $\#D_{0,0} + \#D_{1,0}$ is the number of $f \in \mathbb{F}_{2^m}^*$ with $\mathrm{Tr}(f) = 0$. We also have $\#D_{1,0} + \#D_{1,1} = 2^{m-1}$. Indeed, for the same reason, $\#D_{1,0} + \#D_{1,1}$ is the number of $f \in \mathbb{F}_{2^m}^*$ with $\mathrm{Tr}(f) = 1$.

We compute $\#D_{0,0} + \#D_{1,1}$, which is equal to the number of $f \in \mathbb{F}_{2^m}^*$ with $\mathrm{Tr}(f^{-1} + f\sqrt[8]{a_6}) = 0$. For each f with $\mathrm{Tr}(f^{-1} + f\sqrt[8]{a_6}) = 0$, there are exactly two choices of $x \in \mathbb{F}_{2^m}$ such that $x^2 + x + f^{-1} + f\sqrt[8]{a_6} = 0 \iff f^2x^2 + f^2x = f + f^3\sqrt[8]{a_6}$, producing two points (f, fx) on the elliptic curve $v^2 + uv = \sqrt[8]{a_6}u^3 + u$. Hasse's theorem says that this curve has $2^m + 1 + \tau$ points for some integer τ in the interval $[-2\sqrt{2^m}, 2\sqrt{2^m}]$. Since all points of this curve but $(0, 0)$ and the point at infinity appear as (f, fx), it follows that $\#D_{0,0} + \#D_{1,1} = 2^{m-1} + (\tau - 1)/2$. Finally, we have $4\#D_{1,0} = 2(\#D_{0,0} + \#D_{1,0}) + 2(\#D_{1,0} + D_{1,1}) - 2(\#D_{0,0} + \#D_{1,1}) = 2^m - (\tau + 1)$ and $4\#D_{0,0} = 4(2^{m-1} - 1) - 4\#D_{1,0} = 2^m + (\tau - 3)$. Since $\tau \in [-2\sqrt{2^m}, 2\sqrt{2^m}]$, this implies $\#D_{1,0} \geq \frac{2^m - 2\sqrt{2^m} - 1}{4}$ and $\#D_{0,0} \geq \frac{2^m - 2\sqrt{2^m} - 3}{4}$. The result now follows by remarking that $2^m - 2\sqrt{2^m} - 1 > 2^m - 2\sqrt{2^m} - 3 > 0$ for $m \geq 4$. This is true since $(\sqrt{2^m} - 1)^2 \geq (\sqrt{2^4} - 1)^2 > 4$, which yields $2^m - 2\sqrt{2^m} + 1 > 4 \iff 2^m - 2\sqrt{2^m} - 3 > 0$. □

The arithmetic on generalized binary Huff curves is easily derived from the formulæ given in Section 2. Let $E^\dagger$ be a generalized binary Huff curve as per Eq. (10). Let also $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2) \in E^\dagger$ be two finite points. The formula given by Eq. (3) remains valid for computing the inverse of $\boldsymbol{P}$. An alternate expression for $-\boldsymbol{P} = (\bar{x_1}, \bar{y_1})$ is, whenever defined,

$$\bar{x_1} = \frac{y_1(\hat{\alpha}x_1 + 1)}{\hat{\beta}y_1 + 1} \quad \text{and} \quad \bar{y_1} = \frac{x_1(\hat{\beta}y_1 + 1)}{\hat{\alpha}x_1 + 1} \ . \tag{11}$$

where $\hat{\alpha} = \frac{a+b}{bf}$ and $\hat{\beta} = \frac{a+b}{af}$. The exceptional points of Eq. (11) are $\boldsymbol{P} = \left(\frac{a+b}{bf}, \frac{af}{a+b}\right)$, $\boldsymbol{P} = \left(\frac{bf}{a+b}, \frac{af}{a+b}\right)$, $\boldsymbol{P} = \left(\frac{bf}{a+b}, \frac{a+b}{af}\right)$, and $\boldsymbol{P} = \left(\frac{a+b}{bf}, \frac{a+b}{af}\right)$, the inverses of which are $-\boldsymbol{P} = (1 : 0 : 0)$, $-\boldsymbol{P} = \left(\frac{bf(a+b+af)^2}{(a+b)(a+b+bf)^2}, \frac{af(a+b+bf)^2}{(a+b)(a+b+af)^2}\right)$, $-\boldsymbol{P} = (0 : 1 : 0)$, and $-\boldsymbol{P} = \left(\frac{(a+b)(a+b+bf)^2}{bf(a+b+af)^2}, \frac{(a+b)(a+b+af)^2}{af(a+b+bf)^2}\right)$, respectively.

The doubling formula (Eq. (4)) becomes $[2]\boldsymbol{P} = (x_3, y_3)$ with

$$x_3 = \frac{f(a+b){x_1}^2(1+y_1)^2}{b(1+x_1)^2(1+x_1y_1)^2} \quad \text{and} \quad y_3 = \frac{f(a+b){y_1}^2(1+x_1)^2}{a(1+y_1)^2(1+x_1y_1)^2} \ . \tag{12}$$

The exceptional cases are handled in the same way as in Section 2. We note that the condition $x_1y_1 = 1$ is only possible when $\operatorname{Tr}(f^{-1}) = 0$.

The formula for dedicated point addition (Eq. (5)) is unchanged. For the unified point addition, we get, whenever defined, $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ with

$$\begin{cases} x_3 = \dfrac{b(x_1 + x_2)(1 + x_1x_2y_1y_2) + f(a+b)x_1x_2(1 + y_1y_2)}{b(1 + x_1x_2)(1 + x_1x_2y_1y_2)} \\ y_3 = \dfrac{a(y_1 + y_2)(1 + x_1x_2y_1y_2) + f(a+b)y_1y_2(1 + x_1x_2)}{a(1 + y_1y_2)(1 + x_1x_2y_1y_2)} \end{cases} . \tag{13}$$

The exceptional cases are the same as for the (regular) binary Huff curves. In particular, Proposition 1 remains valid for generalized binary Huff curves: unified addition is complete in any subgroup that does not contain the points at infinity.

Furthermore, the performance is unchanged. This is easily seen by comparing the generalized formulæ of this section with the previous ones (i.e., for $f = 1$). The cost of a point doubling, dedicated point addition, and unified point addition is of $6\mathsf{M} + 2\mathsf{D}$, $15\mathsf{M}$ (or $14\mathsf{M}$ with extended coordinates), and $15\mathsf{M} + 2\mathsf{D}$, respectively. This is better than with binary Edwards curves [5] but, contrary to Edwards' form, unified addition is guaranteed to be complete only in certain proper subgroups.

To sum up, the generalized model presented in this section can be used to represent any ordinary elliptic curve over a finite field of characteristic two; this includes all NIST-recommended curves. It offers a competitive arithmetic leading to efficient implementations. Further, they can be made secure against certain side-channel attacks [11] for cryptographic applications. When the operations of point addition and point doubling make use of different formulæ, they may produce different power traces revealing the secret value of scalar k in the computation of $\boldsymbol{Q} = [k]\boldsymbol{P}$. There are basically three known approaches to circumvent the leakage: *(i)* unifying the addition formulæ, *(ii)* inserting dummy operations, and *(iii)* using regular scalar multiplication algorithms [3, Chapter V]. We note that with the second approach the resulting implementations become vulnerable to safe-error attacks [22]. The so-called Montgomery ladder [16] is an example of a regular scalar multiplication algorithm (third approach). It can be used with the differential point addition formulæ given in Section 4. Given their connection with those of the Weierstraß model, the so-obtained, Huff-based, implementations are equally efficient as the fastest implementations of the Montgomery ladder. But the main advantage of (generalized) Huff curves is that they are equipped with unified point addition formulæ: the same formulæ can be used for doubling or adding points, as required by the first approach against side-channel leakage. The formulæ are even complete — in a subgroup that does not contain the points at infinity. Very few models are known to feature a complete addition law. The Edwards model, as introduced by Bernstein *et al.* in [2], has a such addition law and without any restriction. But this comes at a price: $18\mathsf{M} + 7\mathsf{D}$ (or $21\mathsf{M} + 4\mathsf{D}$) are needed for the complete addition of two points on a binary Edwards curve. Therefore, whenever applicable, the (generalized) binary Huff model should be preferred since it offers faster complete addition formulæ. Other cryptographic applications of complete addition law include protection against exceptional procedure attacks [8] and batch computing [1].

6 Conclusion

This paper studied in detail the addition law for a new model for elliptic curves. While much attention has been paid to elliptic curves over fields of large characteristic, fewer models are known for elliptic curves over binary fields. Our results add the Huff model to the cryptographer's toolbox for the implementation of elliptic curve cryptography in characteristic two. Its distinct arithmetic features may offer an interesting alternative in a number of applications.

Acknowledgments

We are very grateful to the anonymous referees for their useful comments and suggestions.

References

1. Bernstein, D.J.: Batch binary edwards. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 317–336. Springer, Heidelberg (2009)
2. Bernstein, D.J., Lange, T., Farashahi, R.R.: Binary Edwards curves. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 244–265. Springer, Heidelberg (2008)
3. Blake, I.F., Seroussi, G., Smart, N.P. (eds.): Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, vol. 317. Cambridge University Press, Cambridge (2005)
4. Chudnovsky, D.V., Chudnovsky, G.V.: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Advances in Applied Mathematics 7(4), 385–434 (1986)
5. Explicit-formulas database (EFD), http://www.hyperelliptic.org/EFD/
6. Gaudry, P., Lubicz, D.: The arithmetic of characteristic 2 Kummer surfaces and of elliptic Kummer lines. Finite Fields and Applications 15, 246–260 (2009)
7. Huff, G.B.: Diophantine problems in geometry and elliptic ternary forms. Duke Math. J. 15, 443–453 (1948)
8. Izu, T., Takagi, T.: Exceptional procedure attack on elliptic curve cryptosystems. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 224–239. Springer, Heidelberg (2002)
9. Joye, M., Tibouchi, M., Vergnaud, D.: Huff's model for elliptic curves. In: Hanrot, G., Morain, F., Thomé, E. (eds.) ANTS-IX. LNCS, vol. 6197, pp. 234–250. Springer, Heidelberg (2010)
10. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48, 203–209 (1987)
11. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
12. Lenstra Jr., H.W.: Factoring integers with elliptic curves. Annals of Mathematics 126(2), 649–673 (1987)
13. López, J., Dahab, R.: Fast multiplication on elliptic curves over $GF(2^{\mathrm{m}})$ without precomputation. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 316–327. Springer, Heidelberg (1999)
14. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve discrete logaritms to a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)
15. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
16. Montgomery, P.L.: Speeding up the Pollard and elliptic curve methods of factorization. Mathematics of Computation 48(177), 243–264 (1987)
17. National Institute of Standards and Technology: Recommended elliptic curves for federal government use (July 1999), http://csrc.nist.gov/CryptoToolkit/dss/ecdsa/NISTReCur.pdf
18. Peeples Jr., W.D.: Elliptic curves and rational distance sets. Proc. Am. Math. Soc. 5, 29–33 (1954)

19. Silverman, J.H.: The Arithmetic of Elliptic Curves. In: Graduate Texts in Mathematics, vol. 106, ch. III. Springer, Heidelberg (1986)
20. Stam, M.: On montgomery-like representationsfor elliptic curves over $GF(2^k)$. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 240–253. Springer, Heidelberg (2002)
21. Stein, W.A., et al.: Sage Mathematics Software (Version 4.5.1). The Sage Development Team (2010), http://www.sagemath.org
22. Yen, S.M., Joye, M.: Checking before output not be enough against fault-based cryptanalysis. IEEE Transactions on Computers 49(9), 967–970 (2000)

A Variant of the F4 Algorithm

Antoine Joux[1,2] and Vanessa Vitse[2]

[1] Direction Générale de l'Armement (DGA)
[2] Université de Versailles Saint-Quentin, Laboratoire PRISM, 45 av. des États-Unis, 78035 Versailles cedex, France
`antoine.joux@m4x.org`, `vanessa.vitse@prism.uvsq.fr`

Abstract. Algebraic cryptanalysis usually requires to find solutions of several similar polynomial systems. A standard tool to solve this problem consists of computing the Gröbner bases of the corresponding ideals, and Faugère's F4 and F5 are two well-known algorithms for this task. In this paper, we adapt the "Gröbner trace" method of Traverso to the context of F4. The resulting variant is a heuristic algorithm, well suited to algebraic attacks of cryptosystems since it is designed to compute with high probability Gröbner bases of a set of polynomial systems having the same shape. It is faster than F4 as it avoids all reductions to zero, but preserves its simplicity and its efficiency, thus competing with F5.

Keywords: Gröbner basis, Gröbner trace, F4, F5, multivariate cryptography, algebraic cryptanalysis.

1 Introduction

The goal of algebraic cryptanalysis is to break cryptosystems by using mathematical tools coming from symbolic computation and modern algebra. More precisely, an algebraic attack can be decomposed in two steps: first the cryptosystem and its specifics have to be converted into a set of multivariate polynomial equations, then the solutions of the obtained polynomial system have to be computed. The security of a cryptographic primitive thus strongly relies on the difficulty of solving the associated polynomial system. These attacks have been proven to be very efficient for both public key or symmetric cryptosystems and stream ciphers (see [2] for a thorough introduction to the subject).

In this article, we focus on the polynomial system solving part. It is well known that this problem is very difficult (NP-hard in general). However, for many instances coming from algebraic attacks, the resolution is easier than in the worst-case scenario. Gröbner bases, first introduced in [6], are a fundamental tool for tackling this problem. Historically, one can distinguish two families of Gröbner basis computation algorithms: the first one consists of developments of Buchberger's original algorithm [8,14,15,19], while the second can be traced back to the theory of elimination and resultants and relies on Gaussian elimination of Macaulay matrices [10,24,25,26]. Which algorithm to use depends on the shape and properties of the cryptosystem and its underlying polynomial system (base field, degrees of the polynomials, number of variables, symmetries...).

Faugère's F4 algorithm [14] combines ideas from both families. It is probably the most efficient installation of Buchberger's original algorithm, and uses Gaussian elimination to speed up the time-consuming step of "critical pair" reductions. It set new records in Gröbner basis computation when it was published a decade ago, and its implementation in Magma [5] is still considered as a major reference today. However, F4 shares the main drawback of Buchberger's algorithm: it spends a lot of time computing useless reductions. This issue was addressed by Faugère's next algorithm, F5 [15], which first rose to fame with the cryptanalysis of the HFE Challenge [16]. Since then, it has been successfully used to break several other cryptosystems (e.g. [4,17]), increasing considerably the popularity of algebraic attacks. It is often considered as the most efficient algorithm for computing Gröbner bases over finite fields and its performances are for the main part attributable to the use of an elaborate criterion. Indeed, the F5 criterion allows to skip much more unnecessary critical pairs than the classical Buchberger's criteria [7]; actually it eliminates a priori all reductions to zero under the mild assumption that the system forms a semi-regular sequence [3]. Nevertheless, this comes at the price of degraded performances in the reduction step: during the course of the F5 algorithm, many reductions are forbidden for "signature" compatibility conditions, giving rise to polynomials that are either redundant (not "top-reduced" [13]), or whose tails are left almost unreduced.

In many instances of algebraic attacks, one has to compute Gröbner bases for numerous polynomial systems that have the same shape, and whose coefficients are either random or depend on a relatively small number of parameters. In this context, one should use specifically-devised algorithms that take this information into account. A first idea would be to compute a parametric or *comprehensive Gröbner basis* [30]; its specializations yield the Gröbner bases of all the ideals in a parametric polynomial system. However, for the instances arising in cryptanalysis, the computational cost of a comprehensive Gröbner basis is prohibitive. Another method was proposed by Traverso in the context of modular computations of rational Gröbner bases [29]: by storing the *trace* of an initial execution of the Buchberger algorithm, one can greatly increase the speed of almost all subsequent computations. Surprisingly, it seems that this approach was never applied to cryptanalysis until now.

We present in this paper how a similar method allows to avoid all reductions to zero in the F4 algorithm after an initial precomputation. A list of relevant critical pairs is extracted from a first F4 execution, and is used for all following computations; the precomputation overhead is largely compensated by the efficiency of the F4 reduction step, yielding theoretically better performances than F5. This algorithm is by nature probabilistic: the precomputed list is in general not valid for all the subsequent systems. One of the main contribution of this article is to give a complete analysis of this F4 variant and to estimate its probability of failure, which is usually very small.

The paper is organized as follows. After recalling the basic structure of Buchberger-type algorithms, we explain in section 2 how to adapt it to the context of several systems of the same shape. The detailed pseudo-code of our variant of F4, which consists of the two routines `F4Precomp` and `F4Remake` for the first precomputation and the subsequent iterations respectively, is given in the appendix. In section 3, we recall the mathematical frame for the otherwise imprecise notion of "similar looking systems" and derive probability estimates for the correctness of our algorithm, depending on the type of the system and the size of the base field. We also compare the complexities of our variant and of F5, and explain when it is better to use our algorithm. The last section is devoted to applications: the first example comes from the index calculus method of [20] and is a typical case where our algorithm outperforms F4 and F5. We then show how it fits into the hybrid approach of [4] and consider the example of the cryptanalysis of the UOV signature scheme [22]. The next example is provided by the Kipnis-Shamir attack on the MinRank problem: we compare our results to those of [17]. Finally, we evaluate the performances of our F4 variant on the classical `Katsura` benchmarks. We would like to mention that the Gröbner trace method has already been applied to Faugère's algorithms for the decoding of binary cyclic codes [1]; however, the analysis therein was limited to this very specific case, and no implementation details nor probability estimates were given. This idea was then independently rediscovered in [20], where it was applied to the discrete log problem on elliptic curves.

2 The F4 Variant

2.1 Description of the Algorithm

We begin with the standard characterization of Gröbner bases. The notations LM and LT stand for the leading monomial and the leading term of a polynomial and $\vee$ denotes the least common multiple of two monomials.

Theorem 1 ([8]). *A family $G = \{g_1, \dots, g_s\}$ in $\mathbb{K}[X_1, \dots, X_n]$ is a Gröbner basis if and only if for all $1 \leq i < j \leq s$, the remainder of $S(g_i, g_j)$ on division by G is zero, where $S(g_i, g_j)$ is the S-polynomial of g_i and g_j:*

$$S(g_i, g_j) = \frac{LM(g_i) \vee LM(g_j)}{LT(g_i)} g_i - \frac{LM(g_i) \vee LM(g_j)}{LT(g_j)} g_j$$

It is straightforward to adapt this result into the Buchberger's algorithm [8], which outputs a Gröbner basis of an ideal $I = \langle f_1, \dots, f_r \rangle$: one computes iteratively the remainder by G of every possible S-polynomial and appends this remainder to G whenever it is different from zero. In the following, we will rather work with critical pairs instead of S-polynomials: the critical pair of two polynomials f_1 and f_2 is defined as the tuple $(lcm, u_1, f_1, u_2, f_2)$ where $lcm = LM(f_1) \vee LM(f_2)$ and $u_i = \frac{lcm}{LM(f_i)}$.

The reduction of critical pairs is by far the biggest time-consuming part of the Buchberger's algorithm. The main idea of Faugère's F4 algorithm is to use linear algebra to simultaneously reduce a large number of pairs. At each iteration step, a Macaulay-style matrix is constructed, whose columns correspond to monomials and rows to polynomials. This matrix contains the products $(u_i f_i)$ coming from the selected critical pairs (classically, all pairs with the lowest total degree lcm, but other selection strategies are possible) and also all polynomials involved in their reductions, which are determined during the preprocessing phase. By computing the reduced row echelon form of this matrix, we obtain the reduced S-polynomials of all pairs considered. This algorithm, combined with an efficient implementation of linear algebra, yields very good results.

As mentioned in the introduction, F4 has the drawback of computing many useless reductions to zero, even when the classical criteria of Buchberger [7] are taken into account. But when one has to compute several Gröbner bases of similar polynomial systems, it is possible to avoid, in most cases, all reductions to zero by means of a precomputation on the first system.

Here is the outline of our F4 variant:

1. For precomputation purposes, run a standard F4 algorithm on the first system, with the following modifications:
   – At each iteration, store the list of all polynomial multiples (u_i, f_i) coming from the critical pairs.
   – During the row echelon computing phase, reductions to zero correspond to linear dependency relations between the rows of the matrix; for each such relation, remove a multiple (u_i, f_i) from the stored list (some care must be taken in the choice of the multiple, see the appendix for details).
2. For each subsequent system, run a F4 computation with these modifications:
   – Do not maintain nor update a queue of untreated pairs.
   – At each iteration, instead of selecting pairs from the queue, pick directly from the previously stored list all the relevant multiples (u_i, f_i).

We give in the appendix the detailed pseudo-code of the `F4Precomp` algorithm which performs the precomputation, and of the `F4Remake` algorithm which is used for the subsequent systems.

2.2 Additional Features

For the sake of concision, the pseudo-code of `F4Remake` given in the appendix does not contain a test of the correctness of the computation, except for the basic verification of line 9. More checks could be easily included: for instance, it is possible to store during the precomputation the leading monomials of the generators created at each step, and check in `F4Remake` if the new generators have the correct LM. In case of a failed computation, proper error handling would be recommended, e.g. by resuming the computation with the standard F4. At the end of the execution, a last check would be to verify whether the result (which is always a basis of the ideal) is indeed a Gröbner basis. This can be quite expensive, but is usually unnecessary: indeed, the output is always correct if the sets of leading monomials of the bases returned by `F4Remake` and `F4Precomp` coincide, assuming that the precomputation behaved generically (see section 3). Anyway, when the ideal is zero-dimensional with a small degree (as is often the case in the context of algebraic attacks), a verification is almost immediate.

It is also possible to store during precomputation all the relevant polynomial multiples appearing in the matrices M, instead of only those arising from the critical pairs. This increases greatly the size of `F4Precomp`'s output, but allows to skip the preprocessing phase in `F4Remake`. However, the gain provided by this optimization is relatively minor, since the cost of the preprocessing is usually small compared to the computation of the reduced row echelon form. A different approach is outlined in [1]: instead of recording the information about relevant polynomials in a file, the precomputation directly outputs a program (in the C language) containing the instructions for the subsequent computations. Clearly, this code generating technique is much more complicated, but should be faster even when the compilation time of the output program is taken into account.

3 Analysis of the Algorithm and Complexity

3.1 Similar Systems

Our algorithm is designed to be applied on many systems of the "same shape". If $\{f_1, \ldots, f_r\}$ and $\{f'_1, \ldots, f'_r\}$ are two similarly-looking polynomial systems, we want to estimate the probability that our algorithm computes the Gröbner basis of the second system, the precomputation having been done with the first system. This requires some more precise definitions.

Definition 2. *A generic polynomial F of degree d in n variables $X_1, \ldots, X_n$ is a polynomial with coefficients in $\mathbb{K}[\{Y_{i_1,\ldots,i_n}\}_{i_1+\ldots+i_n \le d}]$ of the form*

$$F = \sum Y_{i_1,\ldots,i_n} X_1^{i_1} \ldots X_n^{i_n}.$$

A generic polynomial is thus a polynomial in which each coefficient is a distinct variable. Such polynomials are interesting to study because a system of random polynomials $f_1, \ldots, f_r$ (i.e. such that each coefficient is random) of total degree $d_1, \ldots, d_r$ respectively, is expected to behave like the corresponding system of generic polynomials.

Let $F_1, \ldots, F_r$ be a system of generic polynomials. If we consider F_i as an element of $\mathbb{K}(\underline{Y})[\underline{X}]$, we can compute the Gröbner basis of this system with the F4 algorithm, at least theoretically (in practice, the rational fraction coefficients will likely become extremely large). Now let $f_1, \ldots, f_r$ be a random system with $\deg(f_i) = \deg(F_i)$. We say that $f_1, \ldots, f_r$ behaves generically if we encounter the same number of iterations as with $F_1, \ldots, F_r$ during the computation of its Gröbner basis using F4, and if the same number of new polynomials with the same leading monomials are generated at each step of the algorithm.

We will now translate this condition algebraically. Assume that the system $f_1, \ldots, f_r$ behaves generically until the $(i-1)$-th step; this implies in particular that the critical pairs involved at step i for both systems are similar, in the following sense: $(lcm, u_1, p_1, u_2, p_2)$ is similar to $(lcm', u_1', p_1', u_2', p_2')$ if $LM(p_1) = LM(p_1')$ and $LM(p_2) = LM(p_2')$ (so that $u_i = u_i'$ and $lcm = lcm'$). Let M_g be the matrix of polynomial multiples constructed by F4 at step i for the generic system, and M be the one for $f_1, \ldots, f_r$. It is possible that after the preprocessing M is smaller than M_g, but for the purpose of our discussion, we may assume that the missing polynomial multiples are added to M; the corresponding rows will have no effect whatsoever later in the algorithm. Thus the k-th rows of M and M_g, seen as polynomials, have identical leading monomial; we note s the number of distinct leading monomials in M (or M_g). Remark that the matrices constructed by F4 are usually almost upper triangular, so that s is close to the number of rows. If we compute the reduced row echelon form of M_g, up to a well-chosen permutation of columns we obtain the following matrix $\tilde{M}_g$ where $r = \ell + s$ is the rank of M_g. Using the same transformations on M with adapted coefficients, we obtain a matrix $\tilde{M}$ where B is a matrix with ℓ columns.

$$\tilde{M}_g = \left(\begin{array}{c|c|c} I_s & 0 & A_{g,1} \\ \hline 0 & I_\ell & A_{g,2} \\ \hline 0 & 0 & 0 \end{array}\right) \qquad \tilde{M} = \left(\begin{array}{c|c|c} I_s & \multicolumn{2}{c}{B_1} \\ \hline 0 & B & B_2 \end{array}\right)$$

Then the system $f_1, \ldots, f_r$ behaves generically at step i if and only if this matrix B has full column rank. Finally, the condition for generic behavior is that at each step, the corresponding matrix B has full column rank. Heuristically, since the system is random, we will assume that these matrices B are random. This hypothesis will allow us to give estimates for the probability that a system behaves generically, using the following easy lemma:

Lemma 3. *Let $M = (m_{ij}) \in \mathcal{M}_{n,\ell}(\mathbb{F}_q)$, $n \geq \ell$, be a random matrix, i.e. such that the coefficients m_{ij} are chosen randomly, independently and uniformly in $\mathbb{F}_q$. Then M has full rank with probability $\prod_{i=n-\ell+1}^{n}(1 - q^{-i})$. This probability is greater than the limit*

$$c(q) = \prod_{i=1}^{\infty}(1 - q^{-i}) = 1 - 1/q + \underset{q \to \infty}{O}(1/q^2).$$

Since a system behaves generically if and only if all the matrices B have full rank, we obtain the probability that our F4 variant works successfully:

Theorem 4. *The algorithm* `F4Remake` *outputs a Gröbner basis of a random system $f_1, \ldots, f_r \in \mathbb{F}_q[\underline{X}]$ with a probability that is heuristically greater than $c(q)^{n_{step}}$, assuming that the precomputation has been done with* `F4Precomp` *in n_{step} steps, for a system $f_1^0, \ldots, f_r^0 \in \mathbb{F}_q[\underline{X}]$ that behaves generically.*

This estimate is relevant as soon as the distribution of the matrices B is sufficiently close to the uniform one and the correlation between the steps is small enough.

For a system of generic polynomials, it is known that the number of steps n_{step} during the execution of F4 (for a degree-graded monomial order) is at most equal to the degree of regularity d_{reg} of the homogenized system, which is smaller than the Macaulay bound $\sum_{i=1}^{r}(\deg F_i - 1) + 1$ [24]; this bound is sharp when the system is underdetermined. Since $c(q)$ converges to 1 when q goes to infinity, for a fixed degree of regularity the probability of success of our algorithm will be very close to 1 when the base field $\mathbb{F}_q$ is sufficiently large.

In practice, it is rather uncommon to deal with completely random polynomials. For many applications, the involved polynomial systems actually depend on some random parameters, hence a more general framework is the following:

Definition 5. *Let V be an algebraic variety in $\mathbb{K}^\ell$ and $F_1, \ldots, F_r$ be polynomials in $\mathbb{K}(V)[\underline{X}]$, where $\mathbb{K}(V)$ is the function field of V. We call the image of the map $V \to \mathbb{K}[\underline{X}]^r, \quad y \mapsto (F_1(y), \ldots, F_r(y))$ a parametric family (or family for short) of systems. We call the system $(F_1, \ldots, F_r)$ the generic parametric system of the family.*

A system of generic polynomials is of course a special case of a generic parametric system. As above, the `F4Remake` algorithm will give correct results for systems $f_1, \ldots, f_r$ in a family that behave like its associated generic parametric system. The probability for this is difficult to estimate since it obviously depends on the family considered, but is usually better than for systems of generic polynomials. An important class of examples is when the highest degree homogeneous part of the F_i has coefficients in $\mathbb{K}$ (instead of $\mathbb{K}(V)$). Then all systems of this parametric family behave generically until the first fall of degree occurs. As a consequence, the probability of success of our algorithm can be quite good even when the base field is relatively small, see section 4.2 for an example.

3.2 Change of Characteristic

Another application of our algorithm is the computation of Gröbner bases of "random" polynomial systems over a large field, using a precomputation done over a small finite field. Even for a single system $f_1, \ldots, f_r$ in $\mathbb{F}_p[\underline{X}]$, it is sometimes more advantageous to precompute the Gröbner basis of a system $f'_1, \ldots, f'_r$ with $\deg f_i = \deg f'_i$ in $\mathbb{F}_{p'}[\underline{X}]$ for a small prime p', and then use `F4Remake` on the initial system, than to directly compute the Gröbner basis with F4. The estimated probabilities derived in section 3.1 do not directly apply to this situation, but a similar analysis can be done.

We recall that for every prime number p, there exists a well-defined reduction map $\mathbb{Q}[\underline{X}] \to \mathbb{F}_p[\underline{X}]$, which sends a polynomial P to $\bar{P} = cP \mod p$, where $c \in \mathbb{Q}$ is such that cP belongs to $\mathbb{Z}[\underline{X}]$ and is primitive (i.e. the gcd of its coefficients is one). Let $I = \langle f_1, \ldots, f_r \rangle$ be an ideal of $\mathbb{Q}[\underline{X}]$, and let $\bar{I} = \langle \bar{f}_1, \ldots, \bar{f}_r \rangle$ be the corresponding ideal in $\mathbb{F}_p[\underline{X}]$; we note $\{g_1, \ldots, g_s\}$ the minimal reduced Gröbner basis of I. According to [12], we say that p is a "lucky" prime if $\{\bar{g}_1, \ldots, \bar{g}_s\}$ is the minimal reduced Gröbner basis of $\bar{I}$, and "unlucky" otherwise. There is a weaker, more useful notion (adapted from [27]) of "F4 (or weak) unlucky prime": a prime number p is called so if the computation of the Gröbner bases of I and $\bar{I}$ with F4 differs. By doing the same analysis as in section 3.1, we can show that p is weakly unlucky if and only if one of the above-defined matrices B is not of full rank. As before, these matrices can heuristically be considered as random and thus we obtain that the probability that a prime p is not weakly unlucky, is bounded from below by $c(p)^{n_{step}}$. So, if we want to compute the Gröbner basis of a system $f_1, \ldots, f_r \in \mathbb{F}_p[\underline{X}]$ where p is a large prime, we can lift this system to $\mathbb{Q}[\underline{X}]$ and then reduce it to $f_1', \ldots, f_r' \in \mathbb{F}_{p'}[\underline{X}]$ where p' is a small prime number. Then we execute `F4Precomp` on the latter system and use the precomputation on the initial system with `F4Remake`. This will produce the correct result if p and p' are not weakly unlucky, thus p', while small enough so that the precomputation takes the least time possible, must be large enough so that the probability $c(p')^{n_{step}}$ is sufficiently close to 1. In practice, this last approach should be used whenever possible. If one has to compute several Gröbner bases over a large field $\mathbb{F}_q$ of systems of the same parametric family, the precomputation should not be done over $\mathbb{F}_q$, but rather over a smaller field. We will adopt this strategy in almost all the applications presented in section 4.

3.3 Precomputation Correctness

The output of `F4Precomp` is correct if the first system behaves generically; we have seen that this occurs with a good probability $c(q)^{n_{step}}$. We will now consider what can happen when the precomputation is not correct, and how to detect it. We can, at least theoretically, run `F4Remake` on the generic system; following Traverso's analysis [29] two cases are then possible:

1. This would produce an error. Then `F4Remake` will fail for most subsequent systems, so this situation can be easily detected after very few executions (the probability of no detection is very low: rough estimates have been given in [29] for the different characteristic case). More precisely, as soon as an error occurs with `F4Remake`, one has to determine whether the precomputation was incorrect or the current system does not behave generically. This can be done by looking at the course of the algorithm: if at some step `F4Remake` computes more new generators than `F4Precomp`, or generators with higher leading monomials, then clearly it is the precomputation which is incorrect.
2. The computation would succeed but the resulting output is not a Gröbner basis. This situation, while unlikely, is more difficult to detect: one has to check that the outputs of `F4Remake` on the first executions are indeed Gröbner bases. If there is a system for which this is not true, then the precomputation is incorrect.

Alternatively, one can run `F4Precomp` on several systems and check that the outputs coincide. If it is not the case, one should obviously select the most common output; the probability that a majority of precomputations is similarly incorrect is extremely low. Of course, if $c(q)^{n_{step}}$ is sufficiently close to 1, then the probability of an incorrect precomputation is low enough not to have to worry about these considerations.

3.4 Complexity

Generally, it is difficult to obtain good estimates for the complexity of Gröbner basis computation algorithms, especially of those based on Buchberger's approach. However, we can give a broad upper bound of the complexity of `F4Remake`, by observing that it can be reduced to the computation of the row echelon form of a D-Macaulay matrix of the homogenized system, whose useless rows would have been removed. In the case of generic systems, D is equal to the degree of regularity d_{reg} of the homogenized system. Thus we have an upper-bound for the complexity of our algorithm:

Proposition 6. *The number of field operations performed by* `F4Remake` *on a system of random polynomials over* $\mathbb{K}[X_1,\ldots,X_n]$ *is bounded by*

$$O\left(\binom{d_{reg}+n}{n}^{\omega}\right)$$

where d_{reg} *is the degree of regularity of the homogenized system and* ω *is the constant of matrix multiplication.*

Since there is no reduction to zero as well with F5 (under the assumption that the system is semi-regular), the same reasoning applies and gives the same upper-bound, cf [3]. However, we emphasize that these estimates are not really sharp and do not reflect the difference in performances between the two algorithms. Indeed, `F4Remake` has two main advantages over F5: the polynomials it generates are fully reduced, and it avoids the incremental structure of F5. More precisely, the F5 criterion relies on the use of a signature or label for each polynomial, and we have already mentioned in the introduction that signature compatibility conditions prohibit some reductions; therefore, the polynomials generated by F5 are not completely reduced, or are even redundant [13]. This incurs either more costly reductions later in the algorithm or a larger number of critical pairs. Secondly, the incremental nature of F5 implies that the information provided by the last system polynomials cannot be used to speed up the first stages of the computation.

Thus, our F4 variant should be used preferentially as soon as several Gröbner bases have to be computed and the base field is large enough for this family of systems. Nevertheless, the F5 algorithm remains irreplaceable when the Gröbner basis of only one system has to be computed, when the base field is too small (in particular over $\mathbb{F}_2$) or when the systems are so large that a precomputation would not be realisable.

4 Applications

In all applications, the variant `F4Remake` is compared with an implementation of F4 which uses the same primitives and structures (in language C), and also with the proprietary software Magma (V2.15-15) whose implementation is probably the best publicly available for the considered finite fields. Unless otherwise specified, all tests are performed on a 2.6 GHz Intel Core 2 Duo processor and times are given in seconds.

4.1 Index Calculus

An index calculus method has been recently proposed in [11,18] for the resolution of discrete logarithm on $E(\mathbb{F}_{q^n})$ where E is an elliptic curve defined over a small degree extension field. In order to find "relations", they make use of Semaev's idea [28] which allows to convert the relation search into the resolution of a multivariate polynomial system. A variation of this approach is given in [20], where relations with a slightly different form are considered: it has the advantage of leading to overdetermined systems and is thus faster in practical cases. We focus on the resolution of the polynomial systems arising from this last attack in the special case of $E(\mathbb{F}_{p^5})$ where p is a prime number. The polynomial systems in this example fit into the framework of parametric families: the coefficients polynomially depend on the x-coordinate of a random point $R \in E(\mathbb{F}_{p^5})$ (and also of the equation of the curve E). Our algorithm is particularly relevant for this example because of the large number of relations to collect, leading to an average of $4!p^2$ systems to solve. Moreover, p is large in all applications so the probability of success of our F4 variant is extremely good.

The systems to solve are composed of 5 equations defined over $\mathbb{F}_p$ of total degree 8 in 4 variables. Degrevlex Gröbner bases of the corresponding ideals over several prime fields of size 8, 16, 25 and 32 bits are computed. The probabilities of failure are estimated under the assumption that the systems are random, and knowing that the computation takes 29 steps.

size of p	est. failure probability	F4Precomp	F4Remake	F4	F4/F4Remake	F4 Magma
8 bits	0.11	8.963	2.844	5.903	2.1	9.660
16 bits	4.4×10^{-4}	(19.07)	3.990	9.758	2.4	9.870
25 bits	2.4×10^{-6}	(32.98)	4.942	16.77	3.4	118.8
32 bits	5.8×10^{-9}	(44.33)	8.444	24.56	2.9	1046

Step	degree	F4Remake matrix size	F4 matrix size	size ratio
14	17	1062×3072	1597×3207	1.6
15	16	1048×2798	1853×2999	1.9
16	15	992×2462	2001×2711	2.2
17	14	903×2093	2019×2369	2.5
18	13	794×1720	1930×2000	2.8

Fig. 1. Experimental results on $E(\mathbb{F}_{p^5})$

As explained in section 3.2, it is sufficient to execute the precomputation on the smaller field to get a list of polynomial multiples that works for the other cases; the timings of `F4Precomp` over the fields of size $16, 25$ and 32 bits are thus just indicative. The above figures show that the precomputation overhead is largely compensated as soon as there are more than two subsequent computations. Note that it would have been hazardous to execute `F4Precomp` on a smaller field as the probability of failure increases rapidly. It is mentioned in [20] that the systems have also been solved with a personal implementation of F5, and that the size of the Gröbner basis it computes at the last step before minimization is surprisingly large (17249 labeled polynomials against no more than 2789 polynomials for both versions of F4). As a consequence, the timings of F5 obtained for these systems are much worse than those of F4 or its variants. This shows clearly that on this example, it is much more efficient to apply our algorithm rather than F5.

4.2 Hybrid Approach

The hybrid approach proposed in [4] relies on a trade-off between exhaustive search and Gröbner basis computation. The basic idea is that when one wants to find a solution of a given system $f_1, \ldots, f_r \in \mathbb{K}[X_1, \ldots, X_n]$, it is sometimes faster to try to guess a small number of variables $X_1, \ldots, X_k$. For each possible k-tuple $(x_1, \ldots, x_k)$, one computes the Gröbner basis of the corresponding specialized system $f_1(x_1, \ldots, x_k), \ldots, f_r(x_1, \ldots, x_k) \in \mathbb{K}[X_{k+1}, \ldots, X_n]$ until a solution is found; the advantage is that the specialized systems are much simpler to solve than the initial one.

The hybrid approach is thus a typical case when many systems of the same shape have to be solved and fits perfectly into the framework of parametric families we have described in section 3.1. However, this method is most useful when the search space is reasonably small, which implies in particular that the size of the base field cannot be too large, so one should be wary of the probability of success before applying our F4 variant to this context. Note that, when the right guess is made for the k-tuple $(x_1, \ldots, x_k)$, the corresponding specialized system does not have a generic behaviour. As soon as this is detected by `F4Remake` (see section 2.2), the computation can be continued with e.g. standard F4.

As an example, we consider the cryptanalysis of the Unbalanced Oil and Vinegar system (UOV, [22]), described in [4]. Briefly, the attack can be reduced to the resolution of a system of n quadratic equations in n variables over a finite field $\mathbb{K}$; for the recommended set of parameters, $n = 16$ and $\mathbb{K} = \mathbb{F}_{16}$. Although the base field is quite small, our F4 variant has rather good results in this cryptanalysis: this is due to the fact that the quadratic part of the evaluated polynomials $f_i(x_1, \ldots, x_k) \in \mathbb{K}[X_{k+1}, \ldots, X_n]$ does not depend on the values of the specialized variables $X_1, \ldots, X_k$, and hence all the systems behave generically until the first fall of degree.

For instance, for $k = 3$ the computation with F4 takes 6 steps, and no fall of degree occurs before the penultimate step, so a heuristic estimation of the probability of success is $c(16)^2 \simeq 0.87$. To check this estimate we have performed an exhaustive exploration of the search space $\mathbb{F}_{16}^3$ using `F4Remake`. The measured probability of success depends of the actual system and varies around 90%, which shows that our estimate is satisfying.

The timings obtained during this experiment confirm that our variant provides a non-negligible speed-up. In particular, our timings are better (after a precomputation of 32.3 sec) than the 9.41 sec of F5 given in [4]; of course, this comparison is not really meaningful since the implementation of finite field arithmetic and linear algebra cannot be compared and since different (but similar) computers have been used.

	F4Remake	F4	F4 Magma	F4/F4Remake
Timing (sec)	5.04	16.77	120.6	3.3
Largest matrix	5913×7005	10022×8329	10245×8552	2.0

Fig. 2. Experimental results on UOV with 3 specialized variables

4.3 MinRank

We briefly recall the MinRank problem: given $m+1$ matrices $M_0, M_1, \ldots, M_m \in \mathcal{M}_n(\mathbb{K})$ and a positive integer r, is there a m-tuple $(\alpha_1, \ldots, \alpha_m) \in \mathbb{K}^m$ such that $\text{Rank}\left(\sum_{i=1}^{m} \alpha_i M_i - M_0\right) \leq r$.

We focus on the challenge A proposed in [9]: $\mathbb{K} = \mathbb{F}_{65521}; m = 10; n = 6; r = 3$. The Kipnis-Shamir's attack converts instances of the MinRank problem into quadratic multivariate polynomial systems [23]. For the set of parameters from challenge A, we thus have to solve systems of 18 quadratic equations in 20 variables, and since they are underdetermined, we can specialize two variables without loss of generality. These systems can be solve either directly or with the hybrid approach [17]; in the first case, our F4 variant will be relevant only if one wants to break several different instances of the MinRank problem.

Experiments with F4 and our variant show that, either for the full systems or the systems with one specialized variable, the matrices involved at different steps are quite large (up to 39138×22968) and relatively sparse (less than 5% non-zero entries). With both types of systems, a lot of reductions to zero occurs; for example, we have observed that for the full system at the 8th step, 17442 critical pairs among 17739 reduce to zero. This makes it clear that the classic F4 algorithm is not well suited for these specific systems.

It is difficult to compare our timings with those given in [17] using F5: besides the fact that the experiments were executed on different computers, the linear algebra used in Faugère's FGb implementation of F5 (whose source code is not public) seems to be highly optimized, even more so than in Magma's implementation of F4. On this point, our own implementation is clearly not competitive: for example, at the 7th step for the full system, Magma's F4 reduces a 26723×20223 matrix in 28.95 sec, whereas at the same step our implementation reduces a slightly smaller matrix of size 25918×19392 in 81.52 sec. Despite these limitations, we have obtained timings comparable with those of [17], listed in the table below. This means that with a more elaborate implementation of linear algebra, our F4 variant would probably be the most efficient for these systems.

	F5	F4Remake	F4	F4 Magma
full system	30.0	27.87	320.2	116.6
1 specialized variable	1.85	2.605	9.654	3.560

Fig. 3. Experimental results on MinRank

Computations were executed on a Xeon bi-processor 3.2 GHz for F5. The results of `F4Remake` have been obtained after a precomputation over $\mathbb{F}_{257}$ of 4682 sec for the full system and 113 sec for the system with one variable specialized.

4.4 Katsura Benchmarks

To illustrate the approach presented in section 3.2, we have applied our algorithm to the computation of the Gröbner bases of the Katsura11 and Katsura12 systems [21], over two prime fields of size 16 and 32 bits. As already explained, the idea is to run a precomputation on a small prime field before executing `F4Remake` over a large field (actually, for Katsura12 the first prime $p = 251$ we chose was weakly unlucky). The timings show that for both systems, the speed gain on 32 bits compensates the precomputation overhead, contrarily to the 16 bits case.

	8 bits	16 bits			32 bits		
	Precomputation	F4Remake	F4	F4 Magma	F4Remake	F4	F4 Magma
Katsura11	27.83	9.050	31.83	19.00	15.50	60.93	84.1
Katsura12	202.5	52.66	215.4	143.3	111.4	578.8	$> 5\,h$

Fig. 4. Experimental results on Katsura11 and Katsura12

As a side note, we observed that surprisingly, the matrices created by F4 are quite smaller in our version than in Magma (e.g. 15393×19368 versus 20162×24137 at step 12 of Katsura12); of course, both version still find the same new polynomials at each step. This phenomenon was already present in the previous systems, but not in such a proportion. This seems to indicate that our implementation of the `Simplify` subroutine is much more efficient.

Step	degree	F4Remake matrix size	F4 matrix size	size ratio
9	10	14846×18928	18913×20124	1.4
10	11	15141×19235	17469×19923	1.2
11	12	8249×12344	16044×19556	3.1
12	13	2225×6320	15393×19368	21.2
13	14	–	15229×19313	–

(At the 13^{th} step, F4 finds no new generator so this step is skipped by `F4Remake`)

Fig. 5. Sizes of the matrices involved in the last steps of Katsura12

5 Conclusion

We have presented in this article a variant of the F4 algorithm that provides a very efficient probabilistic method for computing Gröbner bases; it is especially designed for the case where many similar polynomial systems have to be solved. We have given a precise analysis of this context, estimated the probability of success, and evaluated both theoretically and experimentally the performances of our algorithm, showing that it is well adapted for algebraic attacks on cryptosystems.

Since Faugère's F5 algorithm is considered as the most efficient tool for computing Gröbner bases, we have tried as much as possible to compare its performances with our F4 variant. Clearly, F5 remains irreplaceable when the Gröbner basis of only one system has to be computed or when the base field is too small, in particular over $\mathbb{F}_2$. However, our method should be used preferentially as soon as several Gröbner bases have to be computed and the base field is large enough for the considered family of systems. The obtained timings support in part this claim, indicating that with a more elaborate implementation of linear algebra our algorithm would outperform F5 in most cases.

References

1. Augot, D., Bardet, M., Faugère, J.-C.: On the decoding of binary cyclic codes with the Newton identities. J. Symbolic Comput. 44(12), 1608–1625 (2009)
2. Bard, G.: Algebraic Cryptanalysis, 1st edn. Springer, New York (2009)
3. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. Presented at MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry (2005)
4. Bettale, L., Faugère, J.-C., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology, 177–197 (2009)
5. Bosma, W., Cannon, J.J., Playoust, C.: The Magma algebra system I: The user language. J. Symb. Comput. 24(3/4), 235–265 (1997)

6. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD thesis, University of Innsbruck, Austria (1965)
7. Buchberger, B.: A criterion for detecting unnecessary reductions in the construction of Gröbner bases. In: Ng, K.W. (ed.) EUROSAM 1979 and ISSAC 1979. LNCS, vol. 72, pp. 3–21. Springer, Heidelberg (1979)
8. Buchberger, B.: Gröbner bases: An algorithmic method in polynomial ideal theory. In: Bose, N. (ed.) Multidimensional systems theory, Progress, directions and open problems, Math. Appl., vol. 16, pp. 184–232. D. Reidel Publ. Co., Dordrecht (1985)
9. Courtois, N.: Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 402–421. Springer, Heidelberg (2001)
10. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
11. Diem, C.: On the discrete logarithm problem in elliptic curves. Preprint (2009), http://www.math.uni-leipzig.de/~diem/preprints/dlp-ell-curves.pdf
12. Ebert, G.L.: Some comments on the modular approach to Gröbner-bases. SIGSAM Bull. 17(2), 28–32 (1983)
13. Eder, C., Perry, J.: F5C: a variant of Faugère's F5 algorithm with reduced Gröbner bases arXiv/0906.2967 (2009)
14. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139(1-3), 61–88 (1999)
15. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of ISSAC 2002. ACM, New York (2002)
16. Faugère, J.-C., Joux, A.: Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
17. Faugère, J.-C., Levy-dit-Vehel, F., Perret, L.: Cryptanalysis of MinRank. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 280–296. Springer, Heidelberg (2008)
18. Gaudry, P.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. J. Symbolic Computation (2008), doi:10.1016/j.jsc.2008.08.005
19. Gebauer, R., Möller, H.M.: On an installation of Buchberger's algorithm. J. Symbolic Comput. 6(2-3), 275–286 (1988)
20. Joux, A., Vitse, V.: Elliptic curve discrete logarithm problem over small degree extension fields. Application to the static Diffie–Hellman problem on $E(\mathbb{F}_{q^5})$. Cryptology ePrint Archive, Report 2010/157 (2010)
21. Katsura, S., Fukuda, W., Inawashiro, S., Fujiki, N.M., Gebauer, R.: Distribution of effective field in the Ising spin glass of the $\pm J$ model at $T = 0$. Cell Biochem. Biophys. 11(1), 309–319 (1987)
22. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
23. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
24. Lazard, D.: Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In: van Hulzen, J.A. (ed.) ISSAC 1983 and EUROCAL 1983. LNCS, vol. 162, pp. 146–156. Springer, Heidelberg (1983)

25. Macaulay, F.: Some formulae in elimination. In: Proceedings of London Mathematical Society, pp. 3–38 (1902)
26. Mohamed, M.S.E., Mohamed, W.S.A.E., Ding, J., Buchmann, J.: *MXL2*: Solving polynomial equations over GF(2) using an improved mutant strategy. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 203–215. Springer, Heidelberg (2008)
27. Sasaki, T., Takeshima, T.: A modular method for Gröbner-basis construction over $\mathbb{Q}$ and solving system of algebraic equations. J. Inf. Process. 12(4), 371–379 (1989)
28. Semaev, I.: Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031 (2004)
29. Traverso, C.: Gröbner trace algorithms. In: Gianni, P. (ed.) ISSAC 1988. LNCS, vol. 358, pp. 125–138. Springer, Heidelberg (1989)
30. Weispfenning, V.: Comprehensive Gröbner bases. J. Symbolic Comput. 14(1), 1–29 (1992)

A Pseudo-code

A.1 The Precomputation

Given a family of polynomials $\{f_1, \ldots, f_r\}$, the `F4Precomp` algorithm computes for each iteration step of the classical F4 algorithm, the list of polynomial multiples that will be used by `F4Remake` on subsequent computations. This algorithm follows very closely [14], with these additional features:

– A list L of lists of couples is introduced; at the end of the i-th main iteration, $L[i]$ contains the desired list of polynomial multiples for that step. Each polynomial multiple is represented by a couple (m, n), where m is a monomial and n is the index of the polynomial in a global list G (this list G will be progressively reconstructed by `F4Remake`). In the same way, a list L_{tmp} is used to temporary store these couples.
– Instead of just computing the reduced row echelon form M' of the matrix M, we also compute an auxiliary matrix A such that $AM = M'$. If reductions to zero occur, then the bottom part of M' is null and the corresponding bottom part of A gives the linear dependencies between the rows of M. This information is exploited in lines 21 to 26, in order to remove from the temporary list L_{tmp} the useless multiples before copy in $L[step]$. Actually, only the bottom-left part A' of A is of interest: it contains the linear dependencies between the rows of M coming from the critical pairs, modulo those coming from the preprocessing. It is clear that with each dependency relation, one polynomial multiple can be removed, but some care must be taken in this choice. To do so, the row echelon form $\tilde{A}$ of A' is then computed and the polynomial multiples corresponding to the pivots of $\tilde{A}$ are removed. Among the remaining polynomial multiples, those whose leading monomial is now unique can also be removed.

Apart from these modifications, the pseudo-code is basically the F4 algorithm with Gebauer and Möller installation of the Buchberger's criteria (`Update` subroutine) [19]. The only notable change concerns the implementation of the `Simplify` procedure: instead of searching through all the former matrices and their row echelon forms for the adequate simplification as in [14], we introduce an array $TabSimplify$ which contains for each polynomial f in the basis a list of couple of the form $(m, g) \in T \times \mathbb{K}[\underline{X}]$, meaning that the product mf can be simplified into the more reduced polynomial g. This array is updated after the reduced row echelon form is computed (lines 12 to 16 of `Postprocessing`).

Alg. 1. F4Precomp

```
INPUT: $f_1, \ldots, f_r \in \mathbb{K}[\underline{X}]$    OUTPUT: a list of lists of couples $(m, n) \in T \times \mathbb{N}$
 1. $G \leftarrow [\,],\ G_{min} \leftarrow \emptyset,\ P \leftarrow \emptyset, TabSimplify \leftarrow [\,], L \leftarrow [\,]$
 2. for $i = 1$ to $r$ do
 3.   $G[i] \leftarrow f_i,\quad TabSimplify[i] \leftarrow [(1, f_i)],\quad Update(f_i)$
 4. $step = 1$
 5. while $P \neq \emptyset$ do
 6.   $P_{sel} \leftarrow Sel(P)$
 7.   $F \leftarrow [\,],\ LM(F) \leftarrow \emptyset,\ T(F) \leftarrow \emptyset,\ L[step] \leftarrow [\,],\ L_{tmp} \leftarrow [\,]$
 8.   for all $pair = (lcm, t_1, g_1, t_2, g_2) \in P_{sel}$ do
 9.     for $k = 1$ to $2$ do
10.       $ind \leftarrow index(g_k, G)$
11.       if $(t_k, ind) \notin L_{tmp}$ then
12.         $Append(L_{tmp}, (t_k, ind))$
13.         $f \leftarrow Simplify(t_k, ind)$
14.         $Append(F, f)$
15.         $LM(F) \leftarrow LM(F) \cup \{LM(f)\}$
16.         $T(F) \leftarrow T(F) \cup \{m \in T : m$ monomial of $f\}$
17.   $Preprocessing(F, T(F), LM(F))$
18.   $M \leftarrow$ matrix whose rows are the polynomials in $F$
19.   $(M'|A) \leftarrow ReducedRowEchelonForm(M|I_{\#F})$ $(\Rightarrow AM = M')$
20.   $rank \leftarrow Postprocessing(M', LM(F))$
21.   if $rank < \#F$ then
22.     $A' \leftarrow A[rank + 1..\#F][1..\#L_{tmp}]$
23.     $\tilde{A} \leftarrow ReducedRowEchelonForm(A')$
24.     $C \leftarrow \{c \in \{1, \ldots, \#L_{tmp}\} : c$ is not a column number of a pivot in $\tilde{A}\}$
25.     for $j \in C$ do
26.       if $\exists k \in C,\ k \neq j$ and $LM(F[k]) = LM(F[j])$ then $Append(L[step], L_{tmp}[j])$
27.   else $L[step] \leftarrow L_{tmp}$
28.   $step \leftarrow step + 1$
29. return $L$
```

In the pseudo-code, some variables are supposed to be global: G, a list of polynomials that forms a basis of $\langle f_1, \ldots, f_r \rangle$; G_{min}, a set of polynomials which is the minimized version of G; $TabSimplify$, an array of lists of couples used for the simplification of polynomials multiples; P, a queue of yet untreated critical pairs. The function Sel on line 6 is a selection function, whose expression depends on the chosen strategy; usually, selecting all pairs of lowest total degree lcm (normal strategy) yields the best performances. The notation $index(g, G)$ stands for the integer i such that $G[i] = g$, and the function $pair(f_1, f_2)$ outputs the critical pair $(lcm, u_1, f_1, u_2, f_2)$. Finally, *ReducedRowEchelonForm* computes as expected the reduced row echelon form of its input matrix. We stress that great care should be taken in the implementation of this last function since almost all the execution time of the algorithm is spent in it. Note that the test on line 10 in `Update` is only necessary during the initialisation phase of `F4Precomp` (line 3).

Alg. 2. Update

```
INPUT:  $f \in \mathbb{K}[\underline{X}]$
 1. for all $pair = (lcm, t_1, g_1, t_2, g_2) \in P$ do
 2.   if $(LM(f) \vee LM(g_1)$ divides strictly $lcm)$ AND $(LM(f) \vee LM(g_2)$ divides strictly $lcm)$ then $P \leftarrow P \setminus \{pair\}$
 3. $P_0 \leftarrow \emptyset, P_1 \leftarrow \emptyset, P_2 \leftarrow \emptyset$
 4. for all $g \in G_{min}$ do
 5.   if $LM(f) \wedge LM(g) = 1$ then $P_0 \leftarrow P_0 \cup pair(f,g)$ else $P_1 \leftarrow P_1 \cup pair(f,g)$
 6. for all $pair = (lcm, t_1, g_1, t_2, g_2) \in P_1$ do
 7.   $P_1 \leftarrow P_1 \setminus \{pair\}$
 8.   if $\nexists pair' = (lcm', t'_1, g'_1, t'_2, g'_2) \in P_0 \cup P_1 \cup P_2$ s.t.$lcm' | lcm$ then $P_2 \leftarrow P_2 \cup \{pair\}$
 9. $P \leftarrow P \cup P_2$
10. if $\nexists g \in G_{min}$ such that $LM(g) | LM(f)$ then
11.   for all $g \in G_{min}$ do
12.     if $LM(f) | LM(g)$ then $G_{min} \leftarrow G_{min} \setminus \{g\}$
13.   $G_{min} \leftarrow G_{min} \cup \{f\}$
```

Alg. 3. Preprocessing

```
INPUT:  $F, T(F), LM(F)$
 1. $Done \leftarrow LM(F)$
 2. while $T(F) \neq Done$ do
 3.   $m \leftarrow \max(T(F) \setminus Done)$
 4.   $Done \leftarrow Done \cup \{m\}$
 5.   for all $g \in G_{min}$ do
 6.     if $LM(g) | m$ then
 7.       $g' \leftarrow Simplify\left(\frac{m}{LM(g)}, index(g, G)\right)$
 8.       $Append(F, g')$
 9.       $LM(F) \leftarrow LM(F) \cup \{m\}$
10.       $T(F) \leftarrow T(F) \cup \{m' \in T : m'$ monomial of $g'\}$
11.       break
```

Alg. 4. Simplify

```
INPUT: $t \in T, ind \in \mathbb{N}$     OUTPUT: $p \in \mathbb{K}[\underline{X}]$
 1. for $(m, f) \in TabSimplify[ind]$ (from last to first) do
 2.   if $m = t$ then return $f$
 3.   else if $m|t$ then
 4.     $Append\left(TabSimplify[ind], \left(m, \frac{t}{m}f\right)\right)$
 5.     return $\frac{t}{m}f$
```

Alg. 5. Postprocessing

```
INPUT: a matrix $M$ in reduced row echelon form with $\#F$ lines and an ordered set of
    monomials $LM(F)$
OUTPUT: the rank of the matrix $M$
 1. for $i = 1$ to $\#F$ do
 2.   $f \leftarrow M[i]$
 3.   if $f = 0$ then break
 4.   if $LM(f) \notin LM(F)$ then
 5.     $Append(G, f)$
 6.     $Update(f)$
 7.     $TabSimplify[\#G] \leftarrow [(1, f)]$
 8.   else
 9.     for $g \in G_{min}$ do
10.       $ind \leftarrow index(g, G)$
11.       if $LM(g)|LM(f)$ then
12.         for $j = 1$ to $\#TabSimplify[ind]$ do
13.           if $TabSimplify[ind][j] = \left(\frac{LM(f)}{LM(g)}, .\right)$ then
14.             $TabSimplify[ind][j] = \left(\frac{LM(f)}{LM(g)}, f\right)$
15.             break
16.         if $j > \#TabSimplify[ind]$ then $Append\left(TabSimplify[ind], \left(\frac{LM(f)}{LM(g)}, f\right)\right)$
17. return $i - 1$
```

A.2 F4Remake

The `F4Remake` algorithm uses the same routines `Simplify`, `Preprocessing` and `Postprocessing`. Since it no longer uses critical pairs, the subroutine `Update` can be greatly simplified and is replaced by `Update2`.

Alg. 6. F4Remake

```
INPUT: $f_1, \ldots, f_r \in \mathbb{K}[\underline{X}]$, a list $L$ of lists of couples $(m,n) \in T \times \mathbb{N}$
OUTPUT: $G_{min}$, the reduced minimal Gröbner basis of $f_1, \ldots, f_r$
 1. $G \leftarrow [\,]$, $G_{min} \leftarrow \emptyset, TabSimplify \leftarrow [\,]$
 2. for $i = 1$ to $r$ do
 3.   $G[i] \leftarrow f_i$
 4.   $TabSimplify[i] \leftarrow [(1, f_i)]$
 5.   $Update2(f_i)$
 6. for $step = 1$ to $\#L$ do
 7.   $F \leftarrow [\,]$, $LM(F) \leftarrow \emptyset$, $T(F) \leftarrow \emptyset$
 8.   for all $(m,n) \in L[step]$ do
 9.     if $n > \#G$ then computation fails ! exit
10.     $f \leftarrow Simplify(m,n)$, $Append(F,f)$
11.     $LM(F) \leftarrow LM(F) \cup \{LM(f)\}$
12.     $T(F) \leftarrow T(F) \cup \{m \in T : m$ monomial of $f\}$
13.   $Preprocessing(F, T(F), LM(F))$
14.   $M \leftarrow$ matrix whose rows are the polynomials in $F$
15.   $M' \leftarrow ReducedRowEchelonForm(M)$
16.   $Postprocessing(M', LM(F))$
17. return $InterReduce(G_{min})$
```

Alg. 7. Update2

```
INPUT: $f \in \mathbb{K}[\underline{X}]$
 1. if $\nexists g \in G_{min}$ such that $LM(g) | LM(f)$ then
 2.   for all $g \in G_{min}$ do
 3.     if $LM(f) | LM(g)$ then $G_{min} \leftarrow G_{min} \setminus \{g\}$
 4.   $G_{min} \leftarrow G_{min} \cup \{f\}$
```

Attribute-Based Signatures*

Hemanta K. Maji[1], Manoj Prabhakaran[1], and Mike Rosulek[2]

[1] Department of Computer Science, University of Illinois, Urbana-Champaign
{hmaji2,mmp}@uiuc.edu
[2] Department of Computer Science, University of Montana
mikero@cs.umt.edu

Abstract. We introduce *Attribute-Based Signatures (ABS)*, a versatile primitive that allows a party to sign a message with fine-grained control over identifying information. In ABS, a signer, who possesses a set of attributes from the authority, can sign a message with a predicate that is satisfied by his attributes. The signature reveals no more than the fact that a single user with some set of attributes satisfying the predicate has attested to the message. In particular, the signature hides the attributes used to satisfy the predicate and any identifying information about the signer (that could link multiple signatures as being from the same signer). Furthermore, users cannot collude to pool their attributes together.

We give a general framework for constructing ABS schemes, and then show several practical instantiations based on groups with bilinear pairing operations, under standard assumptions. Further, we give a construction which is secure even against a malicious attribute authority, but the security for this scheme is proven in the generic group model. We describe several practical problems that motivated this work, and how ABS can be used to solve them. Also, we show how our techniques allow us to extend Groth-Sahai NIZK proofs to be simulation-extractable and identity-based with low overhead.

1 Introduction

Alice, a finance manager in a big corporation, while going through her company's financial records, has learned about a major international scandal. She decides to send these records to a major newspaper, retaining her anonymity, but with a proof that she indeed has access to the records in question. It turns out that several people, due to a combination of reasons, may have access to these records: those in the New York, London or Tokyo office who are either finance managers associated with project Skam, or internal auditors. Alice considers using a *ring signature* [26] to endorse her message anonymously, but realizes that it is infeasible not only because of the large number of people involved, but also because she does not know who these people are. She realizes she cannot use a *group signature* [14] either, because the set of people Alice needs to refer to here is idiosyncratic to her purposes, and may not have been already collected into a group.[1] She is also aware of *mesh signatures* [9], but mesh signatures provide

* Partially supported by NSF grants CNS 07-16626 and CNS 07-47027.

[1] Even if a group exists, the group manager could identify Alice as the informant.

no way to convince the newspaper that the financial record was endorsed by a single person, not, say, a programmer in the New York office colluding with an internal auditor in the Smalltown office.

Alice's needs in this story reflect the challenges in a system where the roles of the users depend on the *combination* of attributes they possess. In such systems, users obtain multiple attributes from one or more *attribute authorities*, and a user's capabilities in the system (e.g., sending or receiving messages, access to a resource) depend on their attributes. While offering several advantages, attribute-based systems also present fundamental cryptographic challenges. For instance, suppose Alice wants to simply send a message to the above group of people using an "attribute-based messaging" system; then to provide *end-to-end* secure communication, it must be possible for her to encrypt a message using attribute-keys (rather than individual users' keys). Recently cryptographic tools have emerged to tackle some of these challenges for encryption [27,16,3,31]. In this work, we provide a solution for authentication, which among other things, will let Alice in the above example leak the financial records anonymously, but with the appropriate claim regarding her credentials.

Why Attribute-Based Signatures?

The kind of authentication required in an attribute-based system differs from that offered by digital signatures, in much the same way public-key encryption does not fit the bill for attribute-based encryption. An attribute-based solution requires a richer semantics, including anonymity requirements, similar to signature variants like group signatures [14], ring signatures [26], and mesh signatures [9]. The common theme in all these signature primitives is that they provide a guarantees of *unforgeability* and *signer anonymity*. A valid signature can only be generated in particular ways, but the signature does not reveal any further information about which of those ways was actually used to generate it.

More specifically, group and ring signatures reveal only the fact that a message was endorsed by one of a list of possible signers. In a ring signature, the list is public, chosen by the signer *ad hoc*, and given explicitly. In a group signature, the group must be prepared in advance by a group manager, who can revoke the anonymity of any signer. In mesh signatures, a valid signature describes an access structure and a list of pairs (m_i, vk_i), where each vk_i is the verification key of a standard signature scheme. A valid mesh signature can only be generated by someone in posession of enough standard signatures σ_i, each valid under vk_i, to satisfy the given access structure.

In this work we introduce *attribute-based signatures (ABS)*. Signatures in an ABS scheme describe a message and a predicate over the universe of attributes. A valid ABS signature attests to the fact that "a single user, whose attributes satisfy the predicate, endorsed the message." We emphasize the word "single" in this informal security guarantee; ABS signatures, as in most attribute-based systems, require that colluding parties not be able to pool their attributes together.[2] Furthermore, attribute signatures do

[2] Note that for attribute-based *encryption*, if collusion is allowed there are fairly easy solutions; but for ABS, even after allowing collusion (for instance by considering all users to have the same identity while generating keys), the residual primitive is essentially a mesh signature, which is already a non-trivial cryptographic problem.

not reveal more than the claim being made regarding the attributes, even in the presence of other signatures.

Ring and group signatures are then comparable to special cases of ABS, in which the only allowed predicates are *disjunctions* over the universe of attributes (identities). Only one attribute is required to satisfy a disjunctive predicate, so in these cases collusion is not a concern. As in ring signatures, ABS signatures use *ad hoc* predicates. Mesh signatures allow more fine-grained predicates, but do not provide hiding of signature data that would be needed in an ABS scheme. A straight-forward application of mesh signatures as an ABS scheme would either allow collusion (as in the previous example, a New York programmer colluding with a Smalltown auditor to satisfy the “New York auditor” predicate) or allow signatures to be associated with a pseudonym of the signer (thus linking several signatures as originating from the same signer).

Applications

Attribute-based signatures have natural applications in many systems where users' capabilities depend on possibly complex combinations of attributes. ABS is a natural choice for simple authentication in such systems. One of our motivations for developing such schemes comes from the authentication requirements in an Attribute-Based Messaging (ABM) system. In addition to the “leaking secrets” application described above, in Section 6 we also identify applications in trust negotiation systems.

Overview of Our Results

We introduce the concept of Attribute-Based Signatures (ABS) as a powerful primitive with several applications and several efficient instantiations. Our main technical contributions in this work are the following:

– A formal security definition for ABS, that includes the guarantees of unforgeability (even in the presence of collusion) and privacy for the signer.

– A general framework for constructing ABS schemes. Our framework consists of a “credential bundle” representing the attributes associated with a single user and a non-interactive proof of credential ownership that can be bound to a message. The credential bundle must have the property that multiple users should not be able to collude and combine their credentials. The proof system must have some zero-knowledge-like guarantee so that the signature does not leak information about the signer's identity or attributes.

We instantiate this framework using Boneh-Boyen [6] or Waters [30] signatures as the credential bundle, and Groth-Sahai NIZK proofs [18] as the efficient non-interactive proof system. These instantiations provide practical ABS schemes secure under standard assumptions in bilinear groups.

– We present a practical ABS scheme suitable for high throughput systems. This construction deviates from our framework of credential bundles and proof of credential ownership. In this scheme we do employ a credential bundle scheme (same as the one in the last item above), but use a novel randomization technique to blind the actual attributes. This gives the best efficiency among our schemes. Further, this scheme remains secure even against a corrupt attribute-authority. However, the security of this scheme is proven in the heuristic generic-group model (augmented to handle groups with bilinear pairings).

– One of the most striking features of our construction is that it is very easily amenable to natural multi-authority settings. We describe practical considerations related to such a deployment.

– In the full version we show how our techniques of incorporating digital signatures and non-interactive proofs can be used to add *simulation-extractability* to the Groth-Sahai proof system, several orders of magnitude more efficiently than the only other comparable scheme, constucted by Groth in [17].

Which among the above schemes will suit an application will depend on the specific efficiency and security requirements in the system. In all these schemes, the privacy is unconditional, and it is only the unforgeability that depends on computational assumptions. Within a large enterprise setting (with pre-authenticated users) where the threat of forgery may be limited but the volume of signatures may be large, the final scheme may be the most suited. In more susceptible systems with a high security requirement, one of the schemes based on the Groth-Sahai proof systems maybe more suitable (at the expense of efficiency). The choice also depends on whether the application demands high-volume real-time performance (as in a messaging system) or involves only offline signing and verification (as in leaking a secret).

All of our instantiations depend on expressing the attribute predicate as a monotone-span program, which is the state of the art for attribute-based cryptography [16,3,31]. We remark that unlike in many constructions of attribute-based encryption schemes, we achieve "full security" in all our constructions. That is, we do not weaken the definition in the manner of "selective-ID" security. Nor do we need to limit our construction to a small universe of attributes. In all our instantiations, attributes can be arbitrary strings: given a collision-resistant hash function, an *a priori* unbounded attribute universe can be used.

Further Related Work

Groups with bilinear pairings have been used to construct identity-based (e.g., [8]) and attribute-based encryption schemes [27,16,3]. Non-interactive zero-knowledge proofs (including identity-based proofs) have previously been used in the context of efficient constructions of signature primitives [1,20,10,17].

Khader [22,21] proposes a notion called *attribute-based group signatures*. This primitive hides only the identity of the signer, but reveals which attributes the signer used to satisfy the predicate. It also allows a group manager to identify the signer of any signature (which is similar to the semantics of group signatures [14]); in contrast we require signer privacy to hold against everyone, including all authorities.

Subsequent to a preliminary (unpublished) version of this work, Li and Kim [24] gave an ABS scheme that supports predicates which are solely conjunctions of attributes (hence privacy is required only for the identity of the signer and not for the attributes used in satisfying the predicate), and is restricted to a "selective" unforgeability definition. Guo and Zeng [19] construct an attribute-based signature scheme, although their definition of security did not include any privacy for the signer. Shahandashti and Safavi-Naini [28] and Li et al. [23] construct efficient ABS schemes that support predicates consisting of a single threshold gate.

Binding a non-interactive proof to a message, as we do, is also a feature of *identity-based* proofs [20], in which every proof is bound to some identity, and proofs under one identity cannot be used to forge any proofs under a different identity. Indeed, such ID-based proofs have been used to construct signature-like primitives; however the construction from [20] does not have all the properties we need.

Anonymous credentials [13] is one primitive that has some parallels with ABS, but with goals that differ from ABS in several important ways. ABS could be considered as providing some of the functionality of AC as a very special case, but with a weaker anonymity guarantee. Conversely, some of the techniques used to construct efficient AC systems bear some resemblance to some of our efficient ABS constructions. In the full version we discuss these similarities and differences in more detail.

Another related primitive (but much simpler than ABS) is identity-based signatures (IBS) [29]. It is well-known that a simple scheme using traditional certificates realizes IBS, but dedicated schemes aimed at achieving better efficiency have been widely studied. We refer the reader to a comprehensive survey by Bellare et al. [2] for details.

Supporting multiple attribute-authorities is crucial to many attribute-based systems. Previously, there has been much interest on this aspect for attribute-based *encryption* schemes; see Chase et al. [11,12]. The constructions in this paper readily generalize to the multi-authority setting.

2 Preliminaries

2.1 Groups with Bilinear Pairings

Let $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ be cyclic (multiplicative) groups of order p, where p is a prime. Let g be a generator of $\mathbb{G}$, and h be a generator of $\mathbb{H}$. Then $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ is a *bilinear pairing* if $e(g, h)$ is a generator of $\mathbb{G}_T$, and $e(g^a, h^b) = e(g, h)^{ab}$ for all a, b. We review several standard cryptographic assumptions in such groups:

Definition 1 (q-SDH assumption [6]). *Let* $\mathbb{G}$, $\mathbb{H}$, *and* $\mathbb{G}_T$ *be as above. The* q-Strong Diffie-Hellman (q-SDH) assumption *holds in* $(\mathbb{G}, \mathbb{H})$ *if, given the elements* $(g, g^x, g^{x^2}, \ldots, g^{x^q}, h, h^x) \in \mathbb{G}^{q+1} \times \mathbb{H}^2$, *for random choice of* $x \leftarrow \mathbb{Z}_p$ *and random generators* $g \in \mathbb{G}, h \in \mathbb{H}$, *it is computationally infeasible to compute any pair of the form* $\left(c, g^{\frac{1}{x+c}}\right) \in \mathbb{Z}_p \times \mathbb{G}$.

Definition 2 (SXDH assumption [18]). *Let* $\mathbb{G}$, $\mathbb{H}$, *and* $\mathbb{G}_T$ *be as above. The* Symmetric External Diffie-Hellman (SXDH) assumption *holds in* $(\mathbb{G}, \mathbb{H})$ *if the standard Decisional Diffie-Hellman (DDH) assumption holds simultaneously in* $\mathbb{G}$ *and* $\mathbb{H}$.

Definition 3 (DLIN assumption [7]). *Let* $\mathbb{G}$, $\mathbb{H}$, *and* $\mathbb{G}_T$ *be as above, but with* $\mathbb{G} = \mathbb{H}$. *The* Decision-Linear (DLIN) assumption *holds in* $\mathbb{G}$ *if, given the elements* $(g^x, g^y, g^{rx}, g^{sy}, g^t) \in \mathbb{G}^5$, *for a random choice of* $x, y, r, s \leftarrow \mathbb{Z}_p$, *it is computationally infeasible to determine whether* $t = r + s$ *or* t *is random in* $\mathbb{Z}_p$.

2.2 Monotone Span Programs

Let $\Upsilon : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function. A *monotone span program for* Υ over a field $\mathbb{F}$ is an $\ell \times t$ matrix $\mathbf{M}$ with entries in $\mathbb{F}$, along with a labeling function $a : [\ell] \to [n]$ that associates each row of $\mathbf{M}$ with an input variable of Υ, that, for every $(x_1, \ldots, x_n) \in \{0,1\}^n$, satisfies the following:

$$\Upsilon(x_1, \ldots, x_n) = 1 \iff \exists\, \boldsymbol{v} \in \mathbb{F}^{1\times\ell} : \boldsymbol{v}\mathbf{M} = [1, 0, 0, \ldots, 0] \\ \text{and } (\forall i : x_{a(i)} = 0 \Rightarrow v_i = 0)$$

In other words, $\Upsilon(x_1, \ldots, x_n) = 1$ if and only if the rows of $\mathbf{M}$ indexed by $\{i \mid x_{a(i)} = 1\}$ span the vector $[1, 0, 0, \ldots, 0]$.

We call ℓ the *length* and t the *width* of the span program, and $\ell + t$ the *size* of the span program. Every monotone boolean function can be represented by some monotone span program, and a large class do have compact monotone span programs. In particular, given a circuit expressed using *threshold gates*, with the i-th gate being an $\binom{\ell_i}{t_i}$ threshold gate, it is easy to recursively construct a monotone span program with length $\sum_i(\ell_i - 1) + 1$ and width $\sum_i(t_i - 1) + 1$.

2.3 Non-interactive Proofs

We refer the reader to [18] for detailed definitions of non-interactive witness-indistinguishable (NIWI) proofs, but give a brief overview of the necessary definitions here. A NIWI scheme is comprised of the following main algorithms:

- NIWI.Setup: Outputs a reference string crs.
- NIWI.Prove: On input $(crs; \Phi; x)$, where Φ is a boolean formula and $\Phi(x) = 1$, outputs a proof π.
- NIWI.Verify: On input $(crs; \Phi; \pi)$, outputs a boolean.

The completeness requirement is that $\mathsf{NIWI.Verify}(crs; \Phi; \mathsf{NIWI.Prove}(crs; \Phi; x)) = 1$, if $\Phi(x) = 1$ (i.e., x is a *witness* for Φ). The (perfect) witness indistinguishability requirement is that the distributions $\mathsf{NIWI.Prove}(crs; \Phi; x_1)$ and $\mathsf{NIWI.Prove}(crs; \Phi; x_2)$ are identical when x_1 and x_2 are witnesses for Φ. For the soundness/proof of knowledge requirement, we require the following additional algorithms:

- NIWI.SimSetup: Outputs a simulated reference string crs and trapdoor ψ.
- NIWI.Extract: On input $(crs, \psi; \Phi; \pi)$, outputs a witness x.

We require that the crs output by NIWI.SimSetup is indistinguishable to that of NIWI.Setup. Further, we require that for every $(crs, \psi) \leftarrow \mathsf{NIWI.SimSetup}$, if $\mathsf{NIWI.Verify}(crs; \Phi; \pi) = 1$ then $\mathsf{NIWI.Extract}(crs, \psi; \Phi; \pi)$ outputs a valid witness for Φ, with overwhelming probability.

3 Attribute-Based Signatures: Definitions and Security

Let $\mathbb{A}$ be the universe of possible attributes. A *claim-predicate* over $\mathbb{A}$ is a monotone boolean function, whose inputs are associated with attributes of $\mathbb{A}$. We say that an attribute set $\mathcal{A} \subseteq \mathbb{A}$ *satisfies* a claim-predicate Υ if $\Upsilon(\mathcal{A}) = 1$ (where an input is set to be true if its corresponding attribute is present in $\mathcal{A}$).

Definition 4 (ABS). *An* Attribute-Based Signature (ABS) scheme *is parameterized by a universe of possible attributes $\mathbb{A}$ and message space $\mathbb{M}$, and consists of the following four algorithms.*

- $\mathsf{ABS.TSetup}$ *(to be run by a* signature trustee*: Generates public reference information TPK.*
- $\mathsf{ABS.ASetup}$ *(to be run by an* attribute-issuing authority*): generates a key pair $APK, ASK \leftarrow \mathsf{ABS.ASetup}$.*
- $\mathsf{ABS.AttrGen}$*: On input $(ASK, \mathcal{A} \subseteq \mathbb{A})$, outputs a signing key $SK_{\mathcal{A}}$.*[3]
- $\mathsf{ABS.Sign}$*: On input $(PK = (TPK, APK), SK_{\mathcal{A}}, m \in \mathbb{M}, \Upsilon)$, where $\Upsilon(\mathcal{A}) = 1$, outputs a signature σ.*
- $\mathsf{ABS.Ver}$*: On input $(PK = (TPK, APK), m, \Upsilon, \sigma)$, outputs a boolean value.*

Definition 5 (Correctness). *We call an ABS scheme* correct *if for all $TPK \leftarrow \mathsf{ABS.TSetup}$, all purported APK, all messages m, all attribute sets $\mathcal{A}$, all signing keys $SK_{\mathcal{A}} \leftarrow \mathsf{ABS.AttrGen}(ASK, \mathcal{A})$, all claim-predicates Υ such that $\Upsilon(\mathcal{A}) = 1$, and all signatures $\sigma \leftarrow \mathsf{ABS.Sign}\big(PK = (TPK, APK), SK_{\mathcal{A}}, m, \Upsilon\big)$, we have $\mathsf{ABS.Ver}(PK = (TPK, APK), m, \Upsilon, \sigma) = 1$.*

We present two formal definitions that together capture our desired notions of security. Slightly weaker security requirements may also be useful for most applications, but we use the stronger ones because our constructions satisfy them and because they are much easier to work with.

For simplicity, we only present definitions for the simpler case of a single attribute-issuing authority. The definitions for multiple authorities are analogous, and we discuss this case in Section 5.

Definition 6 (Perfect Privacy). *An ABS scheme is* perfectly private *if, for all honestly generated $TPK \leftarrow \mathsf{ABS.TSetup}$, all purported APK, all attribute sets $\mathcal{A}_1, \mathcal{A}_2$, all $SK_1 \leftarrow \mathsf{ABS.AttrGen}(ASK, \mathcal{A}_1)$, $SK_2 \leftarrow \mathsf{ABS.AttrGen}(ASK, \mathcal{A}_2)$, all messages m, and all claim-predicates Υ such that $\Upsilon(\mathcal{A}_1) = \Upsilon(\mathcal{A}_2) = 1$, the distributions $\mathsf{ABS.Sign}(PK, SK_1, m, \Upsilon)$ and $\mathsf{ABS.Sign}(PK, SK_2, m, \Upsilon)$ are equal.*

In other words, the signer's privacy relies only on the signature trustee, and not the attribute-issuing authority. Even a malicious and computationally unbounded attribute-issuing authority cannot link a signature to a set of attributes or the signing key used to generate it.

We slightly overload notation and write $\mathsf{ABS.Sign}(ASK, m, \Upsilon)$ (i.e., with the attribute authority's private key ASK instead of PK and $SK_{\mathcal{A}}$) to denote the following procedure: first, run $SK_{\mathcal{A}} \leftarrow \mathsf{ABS.AttrGen}(ASK, \mathcal{A})$ for any arbitrary $\mathcal{A}$ satisfying Υ; then output the result of $\mathsf{ABS.Sign}(PK, SK_{\mathcal{A}}, m, \Upsilon)$. For convenience in the experiment below we use $\mathsf{ABS.Sign}(ASK, \cdot, \cdot)$ to generate signatures requested by the adversary. This is reasonable when the scheme satisfies perfect privacy, since any other way of letting the adversary obtain signatures will result in the same distribution.

[3] For simplicity, we treat the signing key as a monolithic quantity. However, in our construction the signing key consists of separate components for each attribute in $\mathcal{A}$, and the $\mathsf{ABS.Sign}$ algorithm needs only as much of $SK_{\mathcal{A}}$ as is relevant to the claim-predicate.

Definition 7 (Unforgeability). *An ABS scheme is* unforgeable *if the success probability of any polynomial-time adversary in the following experiment is negligible:*

1. *Run* $TPK \leftarrow$ ABS.TSetup *and* $(APK, ASK) \leftarrow$ ABS.ASetup. *Give* $PK = (TPK, APK)$ *to the adversary.*
2. *The adversary is given access to two oracles:* ABS.AttrGen$(ASK, \cdot)$ *and* ABS.Sign$(ASK, \cdot, \cdot)$.
3. *At the end the adversary outputs* $(m^*, \Upsilon^*, \sigma^*)$.

We say the adversary succeeds if (m^*, Υ^*) *was never queried to the* ABS.Sign *oracle, and* ABS.Ver$(PK, m^*, \Upsilon^*, \sigma^*) = 1$, *and* $\Upsilon^*(\mathcal{A}) = 0$ *for all* $\mathcal{A}$ *queried to the* ABS.AttrGen *oracle.*

Thus any signature which could not have been legitimately made by a *single* one of the adversary's signing keys is considered a forgery. Note that we do not consider it a forgery if the adversary can produce a *different* signature on (m, Υ) than the one he received from the signing oracle.

4 Constructing ABS Schemes

4.1 Credential Bundles

We introduce a new generic primitive called *credential bundles*, which we use in our ABS constructions. Credential bundles model the intuitive requirements of publicly verifiable attributes that resist collusion.

Definition 8 (Credential bundle scheme). *A* credential bundle scheme *is parameterized by a message space* $\mathbb{M}$, *and consists of the following three algorithms.*

- CB.Setup: *Outputs a verification key* vk *and a secret key* sk.
- CB.Gen: *On input* $(sk, \{m_1, \ldots, m_n\} \subseteq \mathbb{M})$, *outputs a tag* τ *and values* $\sigma_1, \ldots, \sigma_n$.
- CB.Ver: *On input* $(vk, m, (\tau, \sigma))$, *outputs a boolean value.*

The scheme is correct *if, for all* $(\tau, \sigma_1, \ldots, \sigma_n) \leftarrow$ CB.Gen$(sk, m_1, \ldots, m_n)$, *we have* CB.Ver$(vk, m_i, (\tau, \sigma_i)) = 1$ *for all* i.

Clearly by excluding some of the σ_i's from an existing bundle, one can generate a new bundle on a subset of attributes. Our main security definition requires that taking a subset of a *single* bundle is the only way to obtain a new bundle from existing bundles; in particular, attributes from several bundles cannot be combined.

Definition 9. *A credential bundle scheme is* secure *if the success probability of any polynomial-time adversary in the following experiment is negligible:*

1. *Run* $(vk, sk) \leftarrow$ CB.Setup, *and give* vk *to the adversary.*
2. *The adversary is given access to an oracle* CB.Gen$(sk, \cdot)$.
3. *At the end the adversary outputs* $(\tau^*, (m_1^*, \sigma_1^*), \ldots, (m_n^*, \sigma_n^*))$.

We say the adversary succeeds *if* CB.Ver$(vk, m_i^*, (\tau^*, \sigma_i^*)) = 1$ *for all* $i \leq n$, *and if no superset of* $\{m_1^*, \ldots, m_n^*\}$ *was ever queried (in a single query) to the* CB.Gen *oracle.*

From any plain digital signature scheme we can easily construct a credential bundle scheme in which the bundle is a collection of signatures of messages "$\tau \| m_i$", where each m_i is the name of an attribute and τ is an identifier that is unique to each user (e.g., an email address). Conversely, when a credential bundle scheme is restricted to singleton sets of messages, its unforgeability definition is equivalent to normal digital signature unforgeability. Despite this equivalence under black-box reductions, the syntax of credential bundles more closely models our desired semantics for ABS.

4.2 A Framework for ABS

Our generic ABS construction for the case of a *single attribute authority* is given in Figure 1. The construction generalizes easily to the multiple attribute authority case (Section 5). At a high level, to sign a message m with claim-predicate Υ, the signer proves that she possesses either a credential bundle containing either sufficient attributes to satisfy Υ, or a "pseudo-attribute" identified with the pair (m, Υ). Only the signature trustee is capable of generating bundles involving pseudo-attributes (these are verified against the trustee's verification key tvk), but it never does so. Thus the proof is convincing that the signer satisfied Υ. However, in the security reduction, the pseudo-attribute provides a mechanism to bind the NIWI proof to a message and give simulated signatures. In the full version we prove the following:

Let $\mathbb{A}$ be the desired universe of ABS attributes. Let $\mathbb{A}'$ denote a space of *pseudo-attributes*, where $\mathbb{A} \cap \mathbb{A}' = \emptyset$. For every message m and claim-predicate Υ we associate a psuedo-attribute $a_{m,\Upsilon} \in \mathbb{A}'$. Let CB be a secure credential bundle scheme, with message space $\mathbb{A} \cup \mathbb{A}'$, and let NIWI be a perfect NIWI proof of knowledge scheme. Our ABS construction is as follows:

$\mathsf{ABS.TSetup}$: The signature trustee runs $crs \leftarrow \mathsf{NIWI.Setup}$ as well as $(tvk, tsk) \leftarrow \mathsf{CB.Setup}$ and publishes $TPK = (crs, tvk)$.

$\mathsf{ABS.ASetup}$: The attribute-issuing authority runs $(avk, ask) \leftarrow \mathsf{CB.Setup}$ and publishes $APK = avk$ and sets $ASK = ask$.

$\mathsf{ABS.AttrGen}(ASK, \mathcal{A})$: Ensure that $\mathcal{A}$ contains no pseudo-attributes. Then output the result of $\mathsf{CB.Gen}(ask, \mathcal{A})$.

$\mathsf{ABS.Sign}(PK, SK_{\mathcal{A}}, m, \Upsilon)$: Assume that $\Upsilon(\mathcal{A}) = 1$. Parse $SK_{\mathcal{A}}$ as $(\tau, \{\sigma_a \mid a \in \mathcal{A}\})$. Υ is a formula over formal variables $\mathbb{A}$. Define $\widetilde{\Upsilon} := \Upsilon \vee a_{m,\Upsilon}$, where $a_{m,\Upsilon} \in \mathbb{A}'$ is the pseudo-attribute associated with (m, Υ). Thus, we still have $\widetilde{\Upsilon}(\mathcal{A}) = 1$. Let $\{a_1, \ldots, a_n\}$ denote the attributes appearing in $\widetilde{\Upsilon}$. Let vk_i be avk if attribute a_i is a pseudo-attribute, and tvk otherwise. Finally, let $\Phi[vk, m, \Upsilon]$ denote the following boolean expression:

$$\exists\, \tau, \sigma_1, \ldots, \sigma_n : \widetilde{\Upsilon}\Big(\big\{a_i \mid \mathsf{CB.Ver}(vk_i, a_i, (\tau, \sigma_i)) = 1\big\}\Big) = 1 \qquad (1)$$

For each i, set $\hat{\sigma}_i = \sigma_{a_i}$ from $SK_{\mathcal{A}}$ if it is present, and to any arbitrary value otherwise (since then its value does not matter). Compute $\pi \leftarrow \mathsf{NIWI.Prove}\big(crs; \Phi[vk, m, \Upsilon]; (\tau, \hat{\sigma}_1, \ldots, \hat{\sigma}_n)\big)$. Output π as the ABS signature.

$\mathsf{ABS.Ver}(PK, m, \Upsilon, \pi)$: Output the result of $\mathsf{NIWI.Verify}(crs; \Phi[vk, m, \Upsilon]; \pi)$.

Fig. 1. General framework for an ABS scheme

Theorem 1. *Given a NIWI argument of knowledge scheme and any secure credential bundle scheme (equivalently, any digital signature scheme), the construction in Figure 1 is a secure ABS scheme. Further, if the NIWI argument is perfectly hiding, the ABS scheme is perfectly private.*

4.3 Practical Instantiation 1

Our first practical instantiation uses Groth-Sahai proofs [18] as the NIWI component and Boneh-Boyen signatures [5] as the credential bundle component. One notable feature of this choice is that attributes in the scheme are simply Boneh-Boyen signatures on messages of the form "userid||attr".

This instantiation requires cyclic groups of prime order equipped with bilinear pairings (Section 2.1). The Groth-Sahai system can prove satisfiability of *pairing-product equations* in such groups, and the main challenge in this instantiation is expressing the logic of the claim-predicate and the Boneh-Boyen signature verification in this limited vocabulary. We identify $\mathbb{Z}_p^*$ with the universe of attributes, where p is the size of the cyclic group used in the scheme.[4]

Boneh-Boyen signatures. We briefly review the Boneh-Boyen digital signature scheme [6]. As before, we suppose there is a bilinear pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{H}$ have prime order p, and where g is a generator of $\mathbb{G}$, and h is a generator of $\mathbb{H}$. The scheme, described below, is strongly unforgeable under the q-SDH assumpion (Definition 1).

DS.KeyGen: Choose random $b, c, d \leftarrow \mathbb{Z}_p$ and compute $B = g^b$, $C = g^c$, $D = g^d$. The verification key is $(B, C, D) \in \mathbb{G}^3$, and the signing key is $(b, c, d) \in (\mathbb{Z}_p)^3$.

DS.Sign$(sk, m \in \mathbb{Z}_p)$**:** Choose random $r \leftarrow \mathbb{Z}_p$; output $\sigma = \left(h^{1/(b+cm+dr)}, t\right) \in \mathbb{H} \times \mathbb{Z}_p$.

DS.Ver$(vk, m, \sigma = (S, r))$**:** Output 1 if $e(BC^m D^r, S) = e(g, h)$, and 0 otherwise.

Expressing the Non-Interactive Proof using Pairing Equations. We use the notation introduced in Figure 1. We must show how the statement $\Phi[vk, m, \Upsilon]$ (equation 1) can be efficiently encoded in the Groth-Sahai system when the credential bundles use Boneh-Boyen signatures.

Groth-Sahai proofs work by first giving a commitment to the values of the witness, and then proving that the commited values satisfy given pairing equations. Suppose we commit to a group element Z (where the group $\mathbb{G}$ or $\mathbb{H}$ will be clear from context), then we will let $\langle Z \rangle$ denote the formal variable corresponding to that commitment. Thus, we express the statements to be proven as pairing equations whose formal variables we will write in the $\langle Z \rangle$ notation.

[4] More precisely $\mathbb{A} \cup \mathbb{A}' \subseteq \mathbb{Z}_p^*$ where $\mathbb{A}'$ is the universe of pseudo-attributes. As is standard, the universe of (pseudo-)attributes can be extended to $\{0,1\}^*$ by applying a collision-resistant hash with range $\mathbb{Z}_p^*$.

Suppose the modified predicate $\widetilde{\Upsilon}$ has a canonical monotone span program $\mathbf{M}$ of size $\ell \times t$, where the ith row corresponds to the $a(i)$-th attribute mentioned in $\widetilde{\Upsilon}$. To establish $\Phi[vk, m, \Upsilon]$, we prove the following equation, which implies it:

$$\exists\, \tau, \sigma_1, \ldots, \sigma_n, v_1, \ldots, v_n : \boldsymbol{v}\mathbf{M} = [1, 0, \ldots, 0] \\ \wedge \bigwedge_{i=1}^{\ell} \Big[v_i \neq 0 \Rightarrow \mathsf{CB.Ver}(vk, a_{a(i)}, (\tau, \sigma_{a(i)})) = 1 \Big]$$

Then, in addition to τ, $\{\sigma_i\}$, we will have the signer commit to the vector $\boldsymbol{v}$ which can be canonically computed from his satisfying assignment of $\widetilde{\Upsilon}$.

This new boolean expression is a conjunction of two kinds of clauses: The first has the form $\exists \boldsymbol{v} : \boldsymbol{v}\mathbf{M} = [1, \ldots, 0]$. To prove it, we commit to the values g^{v_i} and prove the following pairing equations (for each $j \in [t]$):

$$\prod_{i=1}^{\ell} e(\langle g^{v_i} \rangle, h^{\mathbf{M}_{i,j}}) = \begin{cases} e(g, h) & \text{if } j = 1 \\ e(g^0, h) & \text{otherwise} \end{cases}$$

The other clauses have the form $\exists\, \tau, \sigma, v : \big[v \neq 0 \Rightarrow \mathsf{CB.Ver}(vk, m, (\tau, \sigma)) = 1 \big]$. When we use Boneh-Boyen signatures as the instantiation of credential bundles, these clauses can be simplified to

$$\exists\, \tau, \sigma, v : \big[v \neq 0 \Rightarrow \mathsf{DS.Ver}(vk, \tau \| m, \sigma) = 1 \big]$$

where $\mathsf{DS.Ver}$ is the Boneh-Boyen signature verification.

It is crucial that the proof is a proof *of knowledge*, so the simulator can extract the credential bundles. Thus we commit to τ and r *bitwise*, since they are elements of $\mathbb{Z}_p$and could not otherwise be efficiently extracted in the Groth-Sahai scheme. In this way, the extractor can extract the bits and reconstruct the entire witness τ and r.[5] Let $(\tau, \sigma = (S, r), v)$ be a witness to the above expression. Express τ bitwise as $\tau = \sum_i \tau_i 2^i$. Then $\tau \| m$ may be identified with a number $m2^{|\tau|} + \sum_i \tau_i 2^i$. Similarly, interperet r bitwise as $r = \sum_i r_i 2^i$.

Using the same notation as before, we can prove satisfiability of the clause as follows. We commit to each r_i and τ_i in both groups, as $g^{r_i}, h^{r_i}, g^{\tau_i}, h^{\tau_i}$, and then prove that each is indeed a single bit, using the following pairing equations for all i:

$$\begin{aligned} e(\langle g^{r_i} \rangle, h) &= e(g, \langle h^{r_i} \rangle); & e(\langle g^{\tau_i} \rangle, h) &= e(g, \langle h^{\tau_i} \rangle); \\ e(\langle g^{r_i} \rangle, \langle h^{r_i} \rangle) &= e(\langle g^{r_i} \rangle, h); & e(\langle g^{\tau_i} \rangle, \langle h^{\tau_i} \rangle) &= e(\langle g^{\tau_i} \rangle, h). \end{aligned}$$

Next, observe that the pairing equation $e(BC^{\tau \| m} D^r, S^v) = e(g^v, h)$ is logically equivalent to the expression $v \neq 0 \Rightarrow \mathsf{DS.Ver}(vk, \tau \| m, (S, r)) = 1$, which we need to prove. However, the prover cannot directly compute $BC^{\tau \| m} D^r$ or S^v given

[5] We remark that the proof need not be a proof of knowledge with respect to $\boldsymbol{v}$, so it was safe to use these values directly in $\mathbb{Z}_p$.

the committed values. Thus the prover commits to some additional intermediate values $S^v \in \mathbb{H}$ and $C^\tau, D^r \in \mathbb{G}$, and proves the following equations:

$$\begin{aligned}
&e(\langle D^r \rangle, h) = \textstyle\prod_i e(D^{2^i}, \langle h^{r_i} \rangle); \qquad e(\langle g^v \rangle, \langle S \rangle) = e(g, \langle S^v \rangle);\\
&e(\langle C^\tau \rangle, h) = \textstyle\prod_i e(C^{2^i}, \langle h^{\tau_i} \rangle);\\
&e(\langle g^v \rangle, h) = e(BC^{2^{|\tau|}m}, \langle S^v \rangle)\, e(\langle C^\tau \rangle, \langle S^v \rangle)\, e(\langle D^r \rangle, \langle S^v \rangle).
\end{aligned}$$

Note that since m and $|\tau|$ are public, all the coefficients in these equations can be publicly computed. This completes the description of how we encode the required logic into the Groth-Sahai proof system.

There are two instantiations of the Groth-Sahai proof system over prime order groups, based on the DLIN and SXDH assumptions, both of which are suitable for our purposes. Using these we obtain the following (a more detailed analysis of the efficiency is given in the full version).

Theorem 2. *Under the q-SDH and either DLIN or SXDH assumptions, there is an ABS scheme supporting claim-predicates represented as monotone span programs, with signatures consisting of $O(ks)$ group elements, where s is the size of the monotone span program.*

4.4 Practical Instantiation 2

We can also instantiate our framework using the same approach as above, but with the signature scheme of Waters [30]. Signatures in Waters' scheme do not include any elements of $\mathbb{Z}_p$. This fact allows us to avoid the inefficiency of committing to many components of the Boneh-Boyen signatures in a bitwise fashion. Furthermore, Waters signatures are secure under the much weaker BDH assumption, which is implied by the assumptions required for Groth-Sahai proofs. Thus this instantiation does not require the additional q-SDH assumption. However, as a tradeoff, the Waters instantiation requires larger public parameters: a linear (in the security parameter) number of group elements, not the constant number of group elements needed by the Boneh-Boyen instantiation.

The details of this instantiation follow a similar approach as the previous one, incorporating the verification equation of the Waters signature. We refer the reader to the full version for the complete details.

Theorem 3. *Under either the DLIN or SXDH assumptions, there is an ABS scheme supporting claim-predicates represented as monotone span programs, with signatures consisting of $O(k + s)$ group elements, where s is the size of the monotone span program.*

4.5 Practical Instantiation 3

We now present an ABS scheme which is our most practical. Signatures in the scheme consist of *exactly* $s + 2$ group elements, where s is the size of the claim-predicate's monotone span program. This scheme does not use the Groth-Sahai proof system; we use our own randomization techniques to blind the attributes that are used in signing. One additional advantage of avoiding a NIZK proof system is that the privacy of the signers is provided even against a malicious signature trustee; in contrast the above NIZK-based constructions rely on the signature trustee to set up a common reference string honestly.

Our approach is motivated by the construction of mesh signatures [9], but incorporates the efficient credential bundles of the previous construction, as well as the concept of "pseudo-attributes" to bind a message to the signature. In the full version we give a high-level motivation of the details of this scheme. Below we give a description of the construction:

This construction supports all claim-predicates whose monotone span programs have width at most t_{max}, where t_{max} is an arbitrary parameter. We treat $\mathbb{A} = \mathbb{Z}_p^*$ as the universe of attributes, where p is the size of the cyclic group used in the scheme.[6]

ABS.TSetup: Choose suitable cyclic groups G and H of prime order p, equipped with a bilinear pairing $e : G \times H \to G_T$. Choose a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p^*$. Choose random generators: $g \leftarrow G$; $h_0, \dots h_{t_{\mathsf{max}}} \leftarrow H$. The trustee public key is $TPK = (G, H, \mathcal{H}, g, h_0, \dots, h_{t_{\mathsf{max}}})$.

ABS.ASetup: Choose random $a_0, a, b, c \leftarrow \mathbb{Z}_p^*$ and set:

$$C = g^c; \qquad A_0 = h_0^{a_0}; \qquad A_j = h_j^a \text{ and } B_j = h_j^b \quad (\forall j \in [t_{\mathsf{max}}]).$$

The master key is $ASK = (a_0, a, b)$. The public key APK is $(A_0, \dots, A_{t_{\mathsf{max}}}, B_1, \dots, B_{t_{\mathsf{max}}}, C)$

ABS.AttrGen: On input ASK as above and attribute set $\mathcal{A} \subseteq \mathbb{A}$, Choose random generator $K_{\mathsf{base}} \leftarrow G$. Set:

$$K_0 = K_{\mathsf{base}}^{1/a_0}; \qquad K_u = K_{\mathsf{base}}^{1/(a+bu)} \quad (\forall u \in \mathcal{A})$$

The signing key is then $SK_{\mathcal{A}} = (K_{\mathsf{base}}, K_0, \{K_u \mid u \in \mathcal{A}\})$.

ABS.Sign: On input $(PK, SK_{\mathcal{A}}, m, \Upsilon)$ such that $\Upsilon(\mathcal{A}) = 1$, first convert Υ to its corresponding monotone span program $\mathbf{M} \in (\mathbb{Z}_p)^{\ell \times t}$, with row labeling $u : [\ell] \to \mathbb{A}$. Also compute the vector $\boldsymbol{v}$ that corresponds to the satisfying assignment $\mathcal{A}$. Compute $\mu = \mathcal{H}(m \| \Upsilon)$.

Pick random $r_0 \leftarrow \mathbb{Z}_p^*$ and $r_1, \dots r_\ell \leftarrow \mathbb{Z}_p$ and compute:

$$Y = K_{\mathsf{base}}^{r_0}; \qquad S_i = (K_{u(i)}^{v_i})^{r_0} \cdot (Cg^{\mu})^{r_i} \quad (\forall i \in [\ell]);$$

$$W = K_0^{r_0}; \qquad P_j = \prod_{i=1}^{\ell} (A_j B_j^{u(i)})^{\mathbf{M}_{ij} \cdot r_i} \quad (\forall j \in [t]).$$

We note that the signer may not have $K_{u(i)}$ for every attribute $u(i)$ mentioned in the claim-predicate. But when this is the case, $v_i = 0$, and so the value is not needed. The signature is $\sigma = (Y, W, S_1, \dots, S_\ell, P_1, \dots, P_t)$.

[6] As always, the universe of attributes can be further extended to $\{0,1\}^*$ by applying a collision-resistant hash having range $\mathbb{Z}_p^*$. For simplicity of presentation, we do not include this modification.

ABS.Ver: On input $(PK, \sigma = (Y, W, S_1, \ldots, S_\ell, P_1, \ldots, P_t), m, \Upsilon)$, first convert Υ to its corresponding monotone span program $\mathbf{M} \in (\mathbb{Z}_p)^{\ell \times t}$, with row labeling $u : [\ell] \rightarrow \mathbb{A}$. Compute $\mu = \mathcal{H}(m \| \Upsilon)$. If $Y = 1$, then output reject. Otherwise check the following constraints:

$$e(W, A_0) \stackrel{?}{=} e(Y, h_0)$$

$$\prod_{i=1}^{\ell} e\Big(S_i, (A_j B_j^{u(i)})^{\mathbf{M}_{ij}}\Big) \stackrel{?}{=} \begin{cases} e(Y, h_1)\, e(Cg^\mu, P_1), & j = 1 \\ e(Cg^\mu, P_j), & j > 1, \end{cases}$$

for $j \in [t]$. Return accept if all the above checks succeed, and reject otherwise.

We defer the detailed proof of security (carried out in the generic group model) to the full version.

Theorem 4. *In the generic group model, there is an ABS scheme supporting claim-predicates represented as monotone span programs, with signatures consisting of $s + 2$ group elements, where s is the size of the monotone span program.*

5 Multiple Attribute-Authorities

Our first two intantiations of ABS (indeed, our general framework) can be easily extended for use in an environment with multiple attribute-issuing authorities. Except in a centralized enterprise setting, a single user would acquire her attributes from different authorities (e.g., different government agencies, different commercial services she has subscribed to, different social networks she is registered with and so on). These different attribute authorities may not trust each other, nor even be aware of each other. Indeed, some attribute authorities may be untrustworthy, and this should not affect the trustworthiness of attributes acquired from other authorities, or of ABS signatures involving trustworthy attributes.

Apart from these mutually distrusting attribute authorities, we still require a (possibly separate) *signature trustee* to set up the various public parameters of the ABS signature scheme itself. A signature trustee does not have to trust any attribute authority. The attribute authorities use only the public keys from the signature trustee. As long as the signature trustee is trusted, then the ABS signatures are secure and leak no information about the identity or attributes of the signer. The only requirement for compatibility among attribute authorities is that they all have a mechanism for agreeing on a user's userid (say, an email address) so that a user's bundle of credentials may contain compatible attributes from several authorities.

Finally, the claim-predicate in the ABS signature must carry the identity of the attribute-authorities who *own* the various attributes (possibly as meta-data attached to the attribute description). Given this information, the statement proven in the non-interactive proof can be modified to refer to the appropriate digital signature verification keys corresponding to each attribute, including the pseudo-attribute. If one attribute authority's signatures are compromised, then an ABS verifier should not give much importance to attributes from that authority. However, the ABS signatures themselves are still valid (in that they indeed attest to the given claim-predicate being satisfied) as long as the trustee is uncorrupted.

6 Applications

We identify several natural applications of ABS schemes:

Attribute-based messaging. Attribute-Based Messaging, or ABM, (e.g., [4]) provides an example of a quintessential attribute-based system. In an ABM system, messages are addressed not by the identities of the recipients, but by a predicate on users' attributes which the recipients must satisfy. The users need not be aware of each other's identities or attributes. To provide *end-to-end* message privacy (against users whose attributes do not satisfy the sender's policy), one can use *ciphertext-policy attribute-based encryption*, as proposed by Bethencourt, Sahai and Waters [3]. However, there was no satisfactory way to achieve *authentication* (i.e., for the receiver to verify that the sender also satisfied a particular policy) in an ABM system until now. Existing cryptographic technology, including certificates and mesh signatures, would not provide an adequate level of anonymity for the senders while simultaneously preventing collusions.

In a typical ABM system, a certain degree of authorization is required to send messages to certain groups of users. That is, an attribute-based access control mechanism must decide whether to allow a messaging attempt from a sender, depending on both the attributes of the sender and the attribute-based address attached to the message. ABS can be used to authenticate a sender to the ABM system itself (as opposed to the scenario above, where the sender was authenticating to the message recipient). As the messaging system can publicly verify the ABS signature, this solution eliminates the need for the messaging system to query the attribute database to determine the sender's authorization. Indeed, the messaging system need not know the sender's identity at all.

Finally, because our construction is so readily suited for multi-authority settings, ABS is a natural choice for inter-domain ABM systems. However, there are many engineering and cryptographic challenges involved in other aspects of a truly inter-domain ABM system. For example, Chase's proposal [11] for multi-authority attribute-based encryption (originally for the schemes in [27,16], but can be extended to the one in [3]) requires all the attribute-authorities to share secret keys with a central authority, thereby requiring the central authority to trust all the attribute authorities. In contrast, our ABS system requires no such trust between the signature trustee and attribute authorities. As such, ABS is much better suited to practical inter-domain attribute-based systems than its encryption counterparts.

Attribute-based authentication and trust-negotiation. ABS can also be used as a more general fine-grained authentication mechanism. For instance, a server can publish its access policy for a particular resource along with its encryption public key. When a client wishes to access the resource, the server issues a random challenge string. The client can then generate a session key for (private-key) communication, generate an ABS signature of $(challenge, sessionkey)$ under the server's policy, and send these to the server encrypted under the server's public key. Thereafter, the client and server can communicate using the shared session key. This simple protocol is robust even against a man in the middle.

This technique can be extended to multiple rounds as a simple *trust negotiation* protocol, in which two parties progressively reveal more about their attributes over several rounds of interaction. Several recent works also consider cryptographic approaches to trust negotiation that give more privacy to users than is achieved when they simply take turns revealing their attributes [25,15]. Instead of these techniques, ABS can provide a sophisticated way to reveal partial information about one's attributes that is natural for this setting. Being able to bind a message to such a proof about one's attributes, as ABS permits, also allows one to protect the trust negotiation from outside attack, using an approach as above. At each step of the negotiation, the active party can choose an "ephemeral key" for secure (private-key) communication and sign it using ABS. This approach prevents a man-in-the-middle attacks by an adversary who has enough attributes to intercept the first few steps of the negotiation.

Leaking secrets. The classical application for which the notion of ring-signatures was developed by Rivest, Shamir and Tauman [26] is "leaking secrets," that we used as the motivating example in the opening of this paper. Ring signatures support only claim-predicates which are disjunctions. Mesh signatures are an extension of this concept which allow more sophisticated claim-predicates, but permit multiple parties to pool their attributes (atomic signatures). This is not necessarily the intended semantics in natural secret-leaking environment. ABS, on the other hand, provides the semantics that a *single* user (not a coalition) whose attributes satisfy the stated predicate attests to the secret.

References

1. 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)
2. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. Journal of Cryptology 22(1), 1–61 (2009); Preliminary version appeared in Eurocrypt 2004
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
4. Bobba, R., Fatemieh, O., Khan, F., Gunter, C.A., Khurana, H.: Using attribute-based access control to enable attribute-based messaging. In: ACSAC, pp. 403–413. IEEE Computer Society, Los Alamitos (2006)
5. 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)
6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
8. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)
9. Boyen, X.: Mesh signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
10. 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)
11. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)

12. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM Conference on Computer and Communications Security, pp. 121–130. ACM, New York (2009)
13. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. ACM Commun. 28(10), 1030–1044 (1985)
14. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
15. Frikken, K.B., Li, J., Atallah, M.J.: Trust negotiation with hidden credentials, hidden policies, and policy cycles. In: NDSS. The Internet Society, San Diego (2006)
16. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 89–98. ACM, New York (2006)
17. 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)
18. 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)
19. Guo, S., Zeng, Y.: Attribute-based signature scheme. In: International Conference on Information Security and Assurance, pp. 509–511. IEEE, Los Alamitos (2008)
20. Katz, J., Ostrovsky, R., Rabin, M.O.: Identity-based zero-knowledge. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 180–192. Springer, Heidelberg (2005)
21. Khader, D.: Attribute based group signature with revocation. Cryptology ePrint Archive, Report 2007/241 (2007), http://eprint.iacr.org/2007/241
22. Khader, D.: Attribute based group signatures. Cryptology ePrint Archive, Report 2007/159 (2007), http://eprint.iacr.org/2007/159
23. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: Feng, D., Basin, D.A., Liu, P. (eds.) ASIACCS, pp. 60–69. ACM, New York (2010)
24. Li, J., Kim, K.: Attribute-based ring signatures. Cryptology ePrint Archive, Report 2008/394 (2008), http://eprint.iacr.org/2008/394
25. Li, N., Du, W., Boneh, D.: Oblivious signature-based envelope. Distributed Computing 17(4), 293–302 (2005)
26. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
27. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
28. Shahandashti, S.F., Safavi-Naini, R.: Threshold attribute-based signatures and their application to anonymous credential systems. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 198–216. Springer, Heidelberg (2009)
29. 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)
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. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290 (2008), http://eprint.iacr.org/2008/290

Sub-linear Size Traceable Ring Signatures without Random Oracles

Eiichiro Fujisaki

NTT Information Sharing Platform Laboratories, NTT Corporation,
1-1 Hikari-no-oka, Yokosuka, Kanagawa, 239-0847 Japan

Abstract. Traceable ring signatures, proposed at PKC'07, are a variant of ring signatures, which allow a signer to anonymously sign a message with a tag behind a ring, i.e., a group of users chosen by the signer, unless he signs two messages with the *same tag*. However, if a signer signs twice on the same tag, the two signatures will be linked and the identity of the signer will be revealed when the two signed messages are different. Traceable ring signatures can be applied to anonymous write-in voting without any special voting authority and electronic coupon services. The previous traceable ring signature scheme relies on random oracles at its security and the signature size is linear in the number of ring members. This paper proposes the first secure traceable ring signature schemes without random oracles in the common reference string model. In addition, the proposed schemes have a signature size of $O(\sqrt{N})$, where N is the number of users in the ring.

1 Introduction

1.1 Traceable Ring Signatures

Traceable ring signatures [16], initially proposed at PKC'07, are tag-based ring signatures with the restriction that a signer may sign messages only once per tag, where a tag consists of a ring, i.e., a group of users chosen by the signer, and a string, called the issue, that refers to, for instance, a social problem or an election. Similar to ordinary ring signatures, traceable ring signatures require no group manager and no special setup protocol. Traceable ring signatures satisfy the following security requirements.

– **Anonymity -** As long as a signer observes the rule that they may sign only once per tag, his anonymity is guaranteed — it is infeasible for an adversary to determine which (incorrupted) signer in the ring created the signature, and whether or not two signatures on different tags are generated by the same signer.
– **Tag-Linkability -** If a signer breaks the rule and signs twice on the same tag, the signatures will be publicly verified as linked; in other words, the total number of (unlinked) signatures on the same tag is guaranteed to be at most the total number of ring members.

– **Public Traceability -** If a signer breaks the rule and signs two different messages with the same tag, his identity will be publicly traced – the trace can be done without using any secret information.
– **Exculpability -** As long as a signer observes the rule, he cannot be accused of being dishonest by breaking the rule, even if every ring member but him is corrupted.

Traceable ring signatures were inspired by blind signatures and off-line e-cash systems in terms of anonymity restriction. Tag-linkability is the ring signature version of one-more unforgeability in the context of blind signatures, while public traceability is similar to double-spending traceability in off-line e-cash systems.

Traceable ring signatures can be applied to anonymous write-in voting without any voting authority [16]. Another application of traceable ring signatures is an anonymous off-line coupon service, where a service provider issues a tag consisting of customers' public keys and the name of a coupon service. Customers use the tag to make purchases at merchants tied with the service provider, by producing a traceable ring signature on a challenge message from a merchant, including the shop id and a nonce selected by the merchant. A merchant checks if the signature is valid on the tag and the message. Later, the merchant sends the message and signature pairs to the service provider, which searches all signatures to find linked pairs. If there are linked signatures, they can trace dishonest customers because every message, containing the shop id and a nonce chosen by the shop, should be different.

Fujisaki and Suzuki [16] presented the first traceable ring signature in the random oracle model, where the signature size of the scheme is linear in the number of ring members.

1.2 Related Work

Traceable ring signatures are a variant of ring signatures. Ring signatures were introduced by Rivest et al. [23], which allow a signer to sign a message anonymously on behalf of a "ring" of the signer's choice. A verifier can check the validity of the signature, but cannot know who generated it among all possible ring members. In addition, two signatures generated by the same signer are unlinkable. Rivest et al. suggested that this concept can be useful in anonymous leaking of secrets, such as a high-ranking government official leaking important information to the media, by preserving anonymity and verifying the source of information at the same time.

Since [23], this topic has been studied extensively. Chow et al. [13] suggested the first construction for ring signatures without random oracles. Bender et al. [3] refined and strengthened security definitions of ring signatures and showed a construction based on general assumptions, using ZAPs for NP, in the setting of the refined security definitions. Later, Shacham and Waters [24] and Boyen [6] gave more efficient constructions without random oracles under the refined definitions in [3]. Both schemes are based on pairing assumptions and use common reference strings. Compared to the constructions in [3], the schemes in [24,6] are more efficient, but the size of these signatures are still linear in the number of ring members. Although Dodis et al. [14] suggested a scheme with constant signature size independent of ring members, it relies on the random oracle model. Chandran et al. [10] proposed a sub-linear size ring signature scheme in the common reference string model under pairing-based assumptions, which, to our knowledge, is the only known *sub-linear size ring signature scheme without random oracles*.

In ring signatures, the signer is anonymous behind the ring. However, it would be situationally more desirable that anonymity can be flexibly controlled. Linkable ring signatures [21,27,22,26,1,12] are a kind of ring signature with an additional property that two signatures are linked if they are generated by the same signer *with respect to the same ring*, but it is unnecessary that linked signatures are capable of tracing a dishonest signer. Early proposed linkable ring signatures [21,27,22,26] did not take into account insider-attacks[1]. The insider attack is a practical threat in practical applications, as shown in Appendix D.

Recently, Au et. al [1] revisited the security definitions, in which insider attacks were considered. Tsang and Wei [26] proposed a short linkable ring signature scheme whose signature size is independent of the number of ring members, but their scheme is based on [14] and hence relies on random oracles. In addition, it requires a *very strong new assumption* such that, given two distinct RSA moduli $n_1 = p_1q_1$ and $n_2 = p_2q_2$, an adversary cannot distinguish $g^{p_1+q_1}$ from $g^{p_2+q_2}$, where g is in $QR(N)$ for RSA modulus N. Chow et al. [12] suggested a short identity-based linkable ring signature from bilinear pairings. All proposals of linkable ring signature schemes in the literature [21,27,22,26,1,12] rely on random oracles.

Traceable ring signatures imply linkable ring signatures by incorporating a tag system into linkable ring signatures. The definition set of traceable ring signatures can be applied to that of linkable ring signatures by modifying the tracing algorithm, so that it may output the symbol "linked" when two linked signatures on different messages are given (and the definition of exculpability should be modified accordingly, but this is straightforward). The definitions derived from traceable ring signatures are similar to the refined definitions of linkable ring signatures [1]. As also stated in [1], the unforgeability defined in the linkable ring signature is unnecessary because the linkable ring signature version of the exculpability implies unforgeability, as it does in the traceable ring signature. Our proposed schemes can be considered as the first linkable ring signature schemes without random oracles (when seeing our scheme as a linkable ring signature, it can be further simplified by removing Σ, z, and the associated proofs).

[1] In an insider attack, an adversary is given a signing key in the target ring and, after seeing a ring signature on a message created by a honest signer, he forges a ring signature on a different message so that two signatures are linked. In [21,27,22,26], unforgeability is only considered, namely the inability of an adversary to forge a signature with respect to a ring where he is not given any signing key of the ring.

Au et. al [2] suggested a revocable-iff-linked ID-based linkable ring signature scheme, which is somewhat comparable to a traceable ring signature scheme, but their proposal is cryptoanalyzed by [19].

The reader should not confuse traceable *ring* signature schemes with the traceable signature schemes proposed by Kiayias et. al [20]. Traceable signature is a kind of *group* signature with *bi-directional* traceability, where there is a group manager who can reveal all signatures of the target users and all signers of the target signatures by using his master secret key[2].

2 Our Results

The traceable ring signature scheme proposed in [16] is proven secure in the random oracle model. However, as shown in [9], cryptographic protocols proven secure in the random oracle model are not always secure in the real world. In addition, the signature size of the proposal in [16] is linear in the number of ring members. We propose the first traceable ring signature schemes that are secure without random oracles. These schemes have a signature size of $O(k\sqrt{N})$, which are comparable to the most efficient ring signature schemes without random oracles [10], where k is a security parameter and N is the number of ring members. Security of our first proposal is based on familiar assumptions in the pairing-based cryptography except for one, the pseudo-random (PR)-DDHI assumption, which is a slightly stronger analogue of the decisional "bilinear" DHI assumption [15], and can be justified in the generic group model [25]. Our second proposal is a tweak of our first proposal. We can replace the PR-DDHI assumption with a little bit weaker assumption, called the decisional (D)DHI assumption, but the second proposal instead requires a bulletin board.

3 Definitions: Traceable Ring Signatures

We provide the definitions of traceable ring signatures based on the original definitions [16]. The main difference from the definitions in [16] is that in our definition, we incorporate the common reference string generator with a traceable ring signature scheme to treat the common reference string model. We also refine the original definitions to improve readability.

We begin by presenting the functional definition of a traceable ring signature scheme. Let $tag = (tag[1], tag[2])$ be a pair of strings. We refer to an ordered list $R = (pk_1, \ldots, pk_N)$ of public keys as a ring. We assume that $tag[2]$ uniquely defines a ring $R = (pk_1, \ldots, pk_N)$. We write $[N]$ for positive integer N to express set $\{1, \ldots, N\}$.

A traceable ring signature scheme is a tuple of five probabilistic polynomial-time algorithms, $\mathsf{TRS} := (\mathsf{CRSGen}, \mathsf{TRKGen}, \mathsf{TRSig}, \mathsf{TRVrfy}, \mathsf{Trace})$, such that, for $k \in \mathbb{N}$,

[2] In a traceable signature scheme a user can also reveal his identity on his arbitrary signatures without revealing his identity on the other signatures. A traceable ring signature scheme is also implicitly equipped with this property (from the definition of exculapbility).

– $\mathsf{CRSGen}(1^k)$, where k is a security parameter, outputs a common reference string crs.
– $\mathsf{TRKGen}(crs)$ outputs a public-key/secret-key pair (pk, sk).
– $\mathsf{TRSig}(i, sk, crs, tag, m)$ outputs a signature sig on a message $m \in \{0,1\}^*$ with respect to a tag tag under a common reference string crs using the secret-key sk, which corresponds to pk_i in $tag[2]$.
– $\mathsf{TRVrfy}(crs, tag, m, sig)$ verifies purported signature sig on m with respect to tag under crs. It outputs accept if accepting the purported signature; otherwise, outputs reject.
– $\mathsf{Trace}(crs, tag, m, sig, m', sig')$ outputs accept, reject, linked or a public key in the ring $R = tag[2]$. We assume that it always outputs reject if $\mathsf{TRVrfy}(crs, tag, m, sig) = \mathsf{reject}$ or $\mathsf{TRVrfy}(crs, tag, m', sig') = \mathsf{reject}$. We also assume that it always outputs accept or a public key in R if $m \neq m'$, $\mathsf{TRVrfy}(crs, tag, m, sig) = \mathsf{accept}$ and $\mathsf{TRVrfy}(crs, tag, m', sig') = \mathsf{accept}$. We say that (tag, m, sig) is independent of (tag, m', sig') if $\mathsf{Trace}(crs, tag, m, sig, m', sig') = \mathsf{accept}$.

We require the following two correctness conditions.

Completeness: For any $k \in \mathbb{N}$, $N \in \mathbb{N}$, $I \in [N]$, any $tag[1] \in \{0,1\}^*$, and $m \in \{0,1\}^*$, if $crs \leftarrow \mathsf{CRSGen}(1^k)$, $\{(pk_i, sk_i)\}_{i\in[N]} \leftarrow \mathsf{TRKGen}(crs)$, and $sig \leftarrow \mathsf{TRSig}(I, sk_I, crs, tag, m)$, where $tag[2] := (pk_1, \ldots, pk_N)$, TRVrfy always accepts (tag, m, sig), where $\{(pk_i, sk_i)\}_{i\in[N]} \leftarrow \mathsf{TRKGen}(crs)$ denotes the sequence of the experiments of $(pk_i, sk_i) \leftarrow \mathsf{TRKGen}(1^k)$ for $i = 1, \ldots, N$.

Public Traceability: For any $k \in \mathbb{N}$, $l \in \mathbb{N}$, $I, J \in [N]$, $tag[1] \in \{0,1\}^*$, and $m \in \{0,1\}^*$, if $crs \leftarrow \mathsf{CRSGen}(1^k)$, $\{(pk_i, sk_i)\}_{i\in[N]} \leftarrow \mathsf{TRKGen}(crs)$, $sig \leftarrow \mathsf{TRSig}(I, sk_I, crs, tag, m)$, and $sig' \leftarrow \mathsf{TRSig}(J, sk_J, crs, tag, m')$, where $tag[2] := (pk_1, \ldots, pk_N)$,

$$\mathsf{Trace}(crs, tag, m, sig, m', sig') = \begin{cases} \mathsf{accept} & \text{if } I \neq J, \\ \mathsf{linked} & \text{else if } m = m', \\ pk_I & \text{otherwise } (m \neq m'), \end{cases}$$

with an overwhelming probability in k.

For simplicity, we generally omit the input "crs" to TRSig, TRVrfy, and Trace. Similarly, we often omit the input "i" to TRSig, to improve readability.

A secure traceable ring signature scheme satisfies the following security requirements, *tag-linkability*, *anonymity*, and *exculpability*. We give the formal definitions in Appendix A.

The outputs of Trace can be interpreted as follows: Trace outputs accept if it has accepted that (tag, m, sig) and (tag, m', sig') were created by different users. If it outputs pk_I, it means that the algorithm has judged that user I in ring R created two signatures on the same tag. If it outputs linked, it means that it has determined that two signatures are linked. Since public traceability is merely a correctness condition, the decisions given by Trace are not guaranteed in general. However, if a traceable ring signature scheme satisfies tag-linkability, anonymity, and exculpability, the following is guaranteed: If tag-linkability holds true in a traceable ring signature scheme, it is guaranteed that a pair of two signatures accepted by Trace are in fact generated by different signers. If tag-linkability and exculpability both hold true, it is guaranteed that the user traced by Trace is really a cheater. However, it is not guaranteed that two "linked" signatures are created by the same signer. If the tracing algorithm outputs linked, there are two cases: two signatures are created by the same signer on the same tag and message, or one of the signatures is intentionally created by a malicious user so that it is linked with the other signature. One could have a stronger variant of the definitions, such that Trace outputs linked if and only if two signatures are in fact created by the same signer, except for where two signatures are identical. However, the weaker definition is not so problematic because if two signatures are linked on the same message, we can just regard one of them as a copy of the other. On the contrary, if two signatures are created on two different messages with the same tag, it is important to be able to distinguish whether two signatures are created by the same signer or two different signers.

3.1 (Sender-Anonymous) Bulletin Board Model

Our second proposal is secure under a weaker mathematical assumption than our first proposal, but it instead requires a bulletin board. A bulletin board is what one expects, on which every party can see the same contents and append a new content to them, but no one can modify nor erase the contents. The bulletin board gives only one user "write" permission at the same time, and the contents on the board can be parsed into an unique ordered list $(a_1, \ldots, a_n)$ according the order of entries, where each content a_i is sent by one party at a time and a new entry a' is appended to the list as $(a_1, \ldots, a_n, a')$. We use the board to convert a tag into a unique tag id τ of a logarithmic length — a signer registers a tag on the board and in return obtains τ when producing a signature. In the first scheme, τ is merely a hash value of a tag, i.e., $\tau = H(tag)$, and hence a bulletin board is unnecessary.

We note that the bulletin board model raises another subtle issue. We expect that ring signature schemes allow a signer to anonymously produce a signature behind a ring of *his choice*. However, if an entry of a tag in the bulletin board is not "sender-anonymous", anonymity of a signer is not preserved on ad-hoc tags. Therefore, we should further assume that each party can make sender-anonymous entries of tags in the bulletin board. However, we take a note that sender-anonymous entries of tags are not required in anonymous e-voting or e-coupon service, and hence the "non-sender anonymous" bulletin board model also makes sense, in which a signer can anonymously produce ring signatures only on the registered tags. Technically in the non-seder-anonymous bulletin board model, the anonymity game can be naturally modified such that an adversary should make an entry of a tag in the bulletin board before he asks the signing oracles to return a signature on the tag.

4 Preliminaries

We use the Boneh, Goh and Nissim (BGN) cryptosystem [5], the Boneh-Boyen (BB) signature scheme [4] adapted to the composite order bilinear group setting, and non-interactive witness indistinguishable proofs, developed by Groth, Ostrovsky, and Sahai [17], Boyen and Waters [7] and Groth and Sahai [18]. We also use, as a building block, the sub-linear size ring signature scheme proposed by Chandran, Groth and Sahai [10].

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate bilinear map defined over two (multiplicative) cyclic groups, $\mathbb{G}$ and $\mathbb{G}_T$, of order $n \triangleq pq$, where p, q are odd primes. By bilinearity, we have $e(x^a, y^b) = e(x, y)^{ab}$ for all $x, y \in \mathbb{G}$ and $a, b \in \mathbb{Z}/n\mathbb{Z}$. By non-degeneration, $e(g, g)$ is a generator of $\mathbb{G}_T$ if g is a generator of $\mathbb{G}$. We require that group operations, group membership, random sampling from $\mathbb{G}$, and the bilinear map be efficiently computable. We write $\mathbb{G}_p$ and $\mathbb{G}_q$ for the unique subgroup of $\mathbb{G}$ that have order p and order q, respectively. Note that for every $x \in \mathbb{G}$, $x_p(\triangleq x^q) \in \mathbb{G}_p$. We denote by Gen_{BGN} the BGN generator that takes a security parameter 1^k and outputs $\Gamma = (\mathbb{G}, \mathbb{G}_T, n, e, g)$ and (p, q), where $n = pq$.

We recall the following assumptions.

Definition 1 (Subgroup Decision Assumption [5]). *We say that the subgroup decision assumption holds if there is a BGN generator* Gen_{BGN}*, such that for any non-uniform adversary* A*,* $|\Pr[\Gamma \leftarrow \mathsf{Gen}_{BGN}(1^k); r \leftarrow_R (\mathbb{Z}/n\mathbb{Z})^\times : C(\Gamma, g^r) = 1] - \Pr[\Gamma \leftarrow \mathsf{Gen}_{BGN}(1^k); r \leftarrow (\mathbb{Z}/p\mathbb{Z})^\times : A(\Gamma, g^{qr}) = 1]| = \mathsf{negl}(k)$*.*

Definition 2 (Strong Diffie-Hellman Assumption in $\mathbb{G}_p$ [4]). *Let* $g_p = g^q$ *and* $\mathbb{G}_p = \langle g_p \rangle$*. The Q-SDH assumption holds in* $\mathbb{G}_p$ *if given* $(e, p, g_p, g_p^x, g_p^{x^2}, \ldots, g_p^{x^Q})$*, no adversary can output* $(c, g_p^{\frac{1}{x+c}}) \in (\mathbb{Z}/p\mathbb{Z})^\times \times \mathbb{G}_p$ *with a non-negligible probability. The probability is taken over random* $x \in \mathbb{Z}/p\mathbb{Z}$ *and random coins of* $\mathsf{Gen}_{BGN}(1^k)$*.*

We regard the BB signature $\sigma_x(m) = g^{1/(x+m)}$ as a pseudo random function. Of course, due to the bilinear map, $\sigma_x(m)$ is distinguishable from a random element in $\mathbb{G}$ if g, g^x are both given and hence g^x is not given to the adversary.

Definition 3 (Pseudo-Random DDHI Assumption in $\mathbb{G}_p$). *Let* Γ*,* $g_p = g^q$*, and* $\mathbb{G}_p$ *be mentioned above. Let* A *be an adversary that takes* $(\Gamma, (p, q), g_p)$ *and may have access to the BB signature oracle* $\sigma_{x,g_p}(\cdot) = g_p^{\frac{1}{x+\cdot}}$ *with queries in* $[Q'] = \{1, \ldots, Q'\}$ *or access to a random function rand* $: [Q'] \to \mathbb{G}_p$*, and finally outputs a single bit to distinguish which oracle he has accessed. We define the advantage of* A *as* $\mathsf{Adv}_{\mathbb{G}_p}^{\mathrm{Q'-PRDDHI}}(A)(k) \triangleq |p_1 - p_0|$*, where* $p_1 = \Pr[\Gamma \leftarrow \mathsf{Gen}_{BGN}(1^k); x \leftarrow_R \mathbb{Z}/n\mathbb{Z} : A(e, p, g_p)^{\sigma_{x,g_p}(\cdot)} = 1]$*, and* $p_0 = \Pr[\Gamma \leftarrow \mathsf{Gen}_{BGN}(1^k) : A(e, p, g_p)^{rand(\cdot)} = 1]$*. We say that the Q'-PR-DDHI assumption holds in* Γ *if for any non-uniform adversary* A*,* $\mathsf{Adv}_{\Gamma}^{\mathrm{Q'-PR-DDHI}}(A)(k)$ *is negligible in* k*.*

We also assume a weaker version of the PR-DDHI assumption.

Definition 4 (Decisional Diffie-Hellman Inversion Assumption in $\mathbb{G}_p$). *Let A be an adversary. Let Γ, g_p and $\mathbb{G}_p$ be mentioned above. Let $\tau \in [Q'] = \{1, \ldots, Q'\}$. We define the advantage of A on τ as $\mathsf{Adv}^{\mathrm{Q'-DDHI}}_{\mathbb{G}_p}(A, \tau)(k) \triangleq |p_1 - p_0|$, where $p_1 = \Pr[\Gamma \leftarrow \mathsf{Gen}_{BGN}(1^k); x \leftarrow_R \mathbb{Z}/p\mathbb{Z} : A(e, p, g_p, g_p^{\frac{1}{x+1}}, \ldots, g_p^{\frac{1}{x+Q'}}) = 1]$, and $p_0 = \Pr[\Gamma \leftarrow \mathsf{Gen}_{BGN}(1^k); x \leftarrow_R \mathbb{Z}/p\mathbb{Z}; \alpha \leftarrow \mathbb{G}_p : A(e, p, g_p, g_p^{\frac{1}{x+1}}, \ldots, g_p^{\frac{1}{x+\tau-1}}, \alpha, g_p^{\frac{1}{x+\tau+1}}, \ldots, g_p^{\frac{1}{x+Q'}}) = 1]$. We say that the Q'-DDHI assumption holds in $\mathbb{G}_p$ if for any non-uniform probabilistic polynomial-time (in k) algorithm A and any $\tau \in [Q']$, $\mathsf{Adv}^{\mathrm{Q'-DDHI}}_{\Gamma}(A, \tau)(k)$ is negligible in k.*

This assumption is a tweak of two existing assumptions [15,8]. Dodis and Yampolsky [15] proposed an indistinguishability assumption based on bilinear groups, called Q'-decisional bilinear DHI assumption, defined as the inability of an adversary to distinguish $e(g, g)^{1/x}$ from a random element in $\mathbb{G}_T$, given $g, g^x, \ldots, g^{x^{Q'}} \in \mathbb{G}$. Camenisch, Hohenberger, and Lysyankaya [8] proposed a similar assumption, where g^x is given to an adversary but $\mathbb{G}$ is not equipped with a bilinear map.

Although it is obvious that the Q'-PR-DDHI assumption implies the Q'-DDHI assumption for same Q', the following is also true:

Theorem 1. *The Q'-PR-DDHI assumption holds if the Q'-DDHI assumption holds, when $Q' = O(poly(k))$.*

The proof can be provided by a standard hybrid argument and hence omitted.

In the full version, we examine these assumptions in the generic group model [25]. The $O(2^k)$-PR-DDHI assumption is stronger than the $O(poly(k))$-DDHI assumption, but they are comparable when analyzed in the generic group model.

We note that if the DDHI assumption or the PR-DDHI assumption holds true in $\mathbb{G}_p$ and $\mathbb{G}_q$, it also holds true in $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q$.

4.1 NIWI Proofs

We use the BGN commitment scheme: We say that $\Gamma^* = (\mathbb{G}, \mathbb{G}_T, n, e, g, h)$ is the BGN commitment public-key, where h ($\neq g$) is a generator of $\mathbb{G}$. To commit to message $m \in \mathbb{G}$, we compute $c = mh^r$ with randomness $r \in \mathbb{Z}/n\mathbb{Z}$. The commitment c perfectly hides m. On the other hand, c uniquely determines the projection of m on $\mathbb{G}_p$ if h is a generator of $\mathbb{G}_q$. We also commit to message $m \in \mathbb{Z}/n\mathbb{Z}$ by computing $c = g^m h^r$, which perfectly hides m, whereas c uniquely determines $m \mod p$ if h is a generator of $\mathbb{G}_q$. We use non-interactive witness indistinguishable proofs [7,18,10] based on the BGN commitments described above. The public-key $\Gamma^* = (n, \mathbb{G}, \mathbb{G}_T, e, g, h)$ for the perfectly-hiding commitment scheme is used as the common reference string for these NIWI proofs.

We define the following languages: $\mathcal{L}_{BB} \triangleq \{(C, v, L) \in \mathbb{G} \times \mathbb{Z}/n\mathbb{Z} \times \mathbb{G} \mid \exists x \in \mathbb{Z}/n\mathbb{Z}$ s.t. $C_p = g_p^x \in \mathbb{G}_p \wedge L_p = g_p^{\frac{1}{x+v}} \in \mathbb{G}_p\}$ and $\mathcal{L}_N^1 \triangleq \{(C, R) \in \mathbb{G} \times \mathbb{G}^N \mid \exists i \in [N]$ s.t. $C_p = (Y_i)_p \in \mathbb{G}_p\}$, where $R = (Y_1, \ldots, Y_N) \in \mathbb{G}^N$. As defined above, we write x_p for $x \in \mathbb{G}$ to denote x^q. We use the following NIWI proofs for the above languages. All the proofs are perfectly witness indistinguishable when h has order n, whereas perfectly sound if h has order q:

NIWI proof that $(C, v, L) \in \mathcal{L}_{BB}$ *[18]:* Common input: $(C, v, L) \in \mathbb{G} \times \mathbb{Z}/n\mathbb{Z} \times \mathbb{G}$. Witness to the prover: $(y, \sigma, r, s) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$, such that $C = yh^r$, $L = \sigma h^s$ and $e(g^v y, \sigma) = e(g, g)$. NIWI proof: $\pi = (g^v y^s)L^r$. The verifier accepts if and only if $e(g^v y, L) = e(g, g) \cdot e(h, \pi)$. This protocol is perfectly witness indistinguishable when h has order n, whereas the protocol is perfectly sound when h has order q.

NIWI proof that $(C, R) \in \mathcal{L}_2^1$*:* Common input: (C, R) where $R = (Y_1, Y_2) \in \mathbb{G}^2$. Witness to the prover: $(i, r) \in \{1, 2\} \times \mathbb{Z}/n\mathbb{Z}$ such that $C = Y_i h^r$. NIWI proof: $(\alpha_1, \pi_\alpha, \pi_C)$, which constructed as follows: The prover picks up $a \leftarrow_R \mathbb{Z}/n\mathbb{Z}$; sets $\alpha_i = gh^a$ and $\alpha_{1-i} = h^{-a}$; computes $\pi_\alpha = (g^{2i-1}h^a)^a$; and $\pi_C = g^{-r} Y_i^a Y_{1-i}^{-a}$. The verifier retrieves $\alpha_2 = g\alpha_1^{-1}$ and accepts if $e(\alpha_1, \alpha_2) = e(h, \pi_\alpha)$ and $e(Y_1, \alpha_1) \cdot e(Y_2, \alpha_2) = e(g, C) \cdot e(h, \pi_C)$. This proof is a special case of the NIWI proof on $\mathcal{L}_N^1$, where $N = 2$. This protocol is perfectly witness indistinguishable when h has order n, whereas the protocol is perfectly sound when h has order q.

NIWI proof that $(C, R) \in \mathcal{L}_N^1$ *[10]:* Common input: (C, R) where $R = (Y_1, \ldots, Y_N) \in \mathbb{G}^N$. Witness to the prover: $(i, Y_i, r) \in [N] \times \mathbb{Z}/n\mathbb{Z}$ such that $C = Y_i h^r$. The $O(\sqrt{N})$ size NIWI proof was proposed by Chandran, Groth and Sahai [10]. Due to the space limitation, we give this protocol in Appendix B. This protocol is perfectly witness indistinguishable when h has order n, whereas the protocol is perfectly sound when h has order q.

5 The First Proposal

In this section, we present a sub-linear size traceable ring signature scheme without random oracles based on the BGN cryptosystem. The basic idea for our construction is to incorporate the intersection technique suggested in [16] into the Chandran-Groth-Sahai (CGS) ring signature [10]. We start by describing the CGS ring signature [10] and then give a naive construction of a traceable ring signature to help the reader's understanding, along with some security problems. Finally, we show how to fix the problems and provide a full description of our scheme. Let $\Gamma^* = (\mathbb{G}, \mathbb{G}_T, n, e, g, h)$ be the BGN commitment public-key, where h $(\neq g)$ is a generator of $\mathbb{G}$, as described in Sec. 4. We write $S[i]$ to denote the i-th entry in the ordered list $S = (S[1], \ldots, S[i], \ldots)$. We let $[N]$ denote $\{1, \ldots, N\}$ for integer N.

The CGS Ring Signature. Let the BGN commitment public-key Γ^* be the common reference string. Let $R = (Y_1, \ldots, Y_N)$ be a ring of public key $Y_i = y_i h^d$, where $y_i = g^{x_i}$ and $i \in [N]$ (In the original version [10], $Y_i = g^{x_i}$, whereas we use commitment $Y = yh^d$ to $y(= g^x)$ as a public-key because we later open the BB signature $\sigma_x(\cdot) = g^{\frac{1}{x+\cdot}}$ and regard it as pseudo random). The CGS ring signature on message m with secret key (x_I, d) of public key Y_I in ring R is produced as follows: The signer first generates a pair of verification and signing keys (vk, sk) of a one-time signature. He then computes the BB signature on message vk, as $\sigma_{x_I}(vk)$. He then commits to y_I in commitment $C = y_I \cdot h^r$ and similarly to $\sigma_{x_I}(vk)$ in $L_{vk} = \sigma_{x_I}(vk) \cdot h^s$. He then creates NIWI proof $\pi_{C,vk,L_{vk}}$ such that $(C, vk, L_{vk}) \in \mathcal{L}_{BB} = \{(C, v, L) \mid \exists x \in \mathbb{Z}/n\mathbb{Z} \text{ s.t. } C_p = g_p^x \in \mathbb{G}_p \wedge L_p = g_p^{\frac{1}{x+v}} \in \mathbb{G}_p\}$. He also produces a NIWI proof, $\Pi_{C,R}$, such that $(C, R) \in \mathcal{L}_N^1 = \{(C, R) \in \mathbb{G} \times \mathbb{G}^N \mid \exists i \in [N] \text{ s.t. } C_p = (Y_i)_p \in \mathbb{G}_p\}$, where $R = (Y_1, \ldots, Y_N) \in \mathbb{G}^N$. The signer finally makes a one-time signature on everything above, $otsig = \mathsf{otSign}_{sk}\ (m, vk, R, C, L_{vk},\ \pi_{C,vk,L}, \Pi_{C,R})$, and outputs $(vk, otsig, C, L, \pi_{C,vk,L}, \Pi_{C,R})$. The size of this proof is $O(\sqrt{N})$, so is that of the whole proof. This ring signature scheme is perfectly anonymous when the common reference string is honestly generated (namely, when the order of g is n). It is also existentially unforgeable, under the subgroup decision assumption, the strong Diffie-Hellman assumption, and the assumption that the one-time signature is unforgeable.

Toward Our Construction. As an intermediate, we consider a naive construction, which is insecure but makes it easy to understand the concept of our construction.

Let Γ^* be the common reference string as above. Let $R = (Y_1, \ldots, Y_N)$ be a ring of public keys as above. Let signer I compute a pseudo random function $F_{x_I}(tag)$, where tag is a tag. We view the BB signature as a pseudo random function, i.e., $F_{x_I}(tag) = \sigma_{x_I}(tag) = g^{\frac{1}{x_I + tag}} \in \mathbb{G}$. Remember that since $y_I = g^{x_I}$ is committed to in Y_I, the bilinear map does not help to distinguish the output of F from a random element. Let $\sigma_I = F_{x_I}(tag)$. Let signer I compute $Q = (\sigma_I \cdot g^{-H(tag,m)})^{I^{-1}}$. We let $\Sigma(tag, m) \triangleq \{\sigma_i | \sigma_i = g^{H(tag,m)} Q^i \text{ for } i \in [N]\}$. Note that $\Sigma(tag, m)$ lies on the line uniquely determined by $H(tag, m)$ and σ_I. What we want to do next is to let the signer I prove that he has *honestly* produced σ_I in a non-interactive zero-knowledge manner. Apparently, he could do this in the BGN setting. In fact, he can produce a non-interactive zero-knowledge proof Π' such that there exists $i \in [N]$, such that $(Y_i, tag, \sigma_i) \in \mathcal{L}_{BB}$, where $Y_i = R[i]$ and $\sigma_i = \Sigma(tag, m)[i]$. The proof Π' can be constructed as follows: Commit to y_I and σ_I in commitments, $C = y_I h^r$ and $L = \sigma_I h^{s'}$, respectively; produce NIWI proof $\pi_{C,tag,L}$ such that $(C, tag, L) \in \mathcal{L}_{BB}$, as described in Sec. 4.1; produce NIZK proofs, $\Pi_{C,R}$ and $\Pi_{L,\Sigma}$, such that $(C, R) \in \mathcal{L}_N^1$ and $(L, \Sigma(tag, m)) \in \mathcal{L}_N^1$, respectively (To convert an NIWI proof in $\mathcal{L}_N^1$ to an NIZK one, we use the standard technique, where we append Y_0 (resp. σ_0) to the common reference string, make an NIZK proof in $\mathcal{L}_N^1$ based on an NIWI proof in $\mathcal{L}_{N+1}^1$, and use the secret of Y_0 (resp. σ_0) when simulating the protocol); finally, make an

NIZK proof such that the same I is used in both $\Pi_{C,R}$ and $\Pi_{L,\Sigma}$, which can be constructed based on the NIWI protocol in Appendix C (End of Π'). The signer then produces the CGS ring signature above, except that everything produced using the intersection technique is added to the message to be signed by the one-time signature. Namely, $otsig = \mathsf{otSign}_{sk}\ (m, vk, R, C, L_{vk},\ \pi_{C,vk,L}, \Pi_{C,R}, tag, L, Q, \Pi')$.

This construction has a signature size of $O(\sqrt{N})$ because $\Sigma(tag, m)$ is retrieved only from $H(tag, m)$ and Q, and the length of Π' is $O(\sqrt{N})$.

The public tracing algorithm outputs I if there is I for $\Sigma(tag, m)$ and $\Sigma(tag, m')$, with $m \neq m'$, such that $\sigma = \sigma'$ where $\sigma = \Sigma(tag, m)[I]$ and $\sigma' = \Sigma(tag, m')[I]$.

This naive construction satisfies anonymity and public traceability: Let $\Sigma(I, tag,\ m) = \Sigma(tag, m)$ such that $Q = g^{(\frac{1}{x_I + tag} - H(tag,m))\cdot I^{-1}}$. We assume the BB signature to be pseudo random. Then, $\Sigma(I, tag, m)$ is computationally indistinguishable from $\Sigma(J, tag', m')$, *roughly* if $I \neq J$ (i.e., if generated by different signers) or $tag \neq tag'$ (i.e., if signed on different tags). On one hand, if signer I *honestly* signs two different messages, m and m', with the same tag, his identity will be revealed because the lines, $\Sigma(tag, m)$ and $\Sigma(tag, m')$, will intersect at a unique point, (I, σ_I), except for the unlikely case of $H(tag, m) = H(tag, m')$. We stress that proof Π' should be *zero-knowledge* instead of witness indistinguishable, because $\Sigma(I, tag, m)$ is only computationally indistinguishable from $\Sigma(J, tag, m)$. In general, two witness indistinguishable proofs are not known indistinguishable when the inputs to the proofs are just computational indistinguishable.

The naive construction does not satisfy tag-linkability and exculpability, however. This mainly comes from the following two reasons:

– Proof Π' is not sufficient to make a malicious signer *honestly* generate σ_I. Π' only proves that, when the order of h is q, the projection of σ_I on $\mathbb{G}_p$ is the BB signature w.r.t. Y_I on $\mathbb{G}_p$. Therefore a malicious signer I can make σ_I such that $\sigma_I = g^{\frac{1}{x_I+tag}} h^r$ for random r and show that the projection of σ_I on $\mathbb{G}_p$ satisfies the statements (on $\mathbb{G}_p$). Therefore, the malicious signer can make two different tokens, $\sigma_I = g^{\frac{1}{x_I+tag}} h^r$ and $\sigma'_I = g^{\frac{1}{x_I+tag}} h^{r'}$, in two signatures with respect to the same tag. We cannot trace him with σ_I and σ'_I, because $\sigma_I \neq \sigma'_I$.
– The other problem is how to prevent two malicious signers, say j and k, with $j \neq k$, from producing the signatures when they collaborate such that $\sigma_I^{(j)} = \sigma_I^{(k)}$, where $\Sigma(j, tag, m) = (\sigma_1^{(j)}, \ldots, \sigma_N^{(j)})$ and $\Sigma(k, tag, m') = (\sigma_1^{(k)}, \ldots, \sigma_N^{(k)})$. If they can do so, they succeed in trapping I (to break exculpability). The original intersection technique in [16] used the unpredictability of the output of random oracles at this crucial point, but we cannot use the random oracles.

As for the first issue, it seems difficult to prove that a signer *honestly* creates σ_I. We instead make him generate it *in such a way that he can never output two different σ_I's on the same tag* by letting him produce π associated with σ

such that $e(\sigma, \hat{h}) = e(g, \pi)$, where $\hat{h}$ is a random element in $\mathbb{G}$, to be appended to the common reference string[3]. If a malicious signer can create both (σ, π) and (σ', π'), with $\sigma \neq \sigma'$ but $\sigma^q = (\sigma')^q$, he can distinguish whether $\hat{h}$ has order n or p because $\pi \neq \pi'$ if $\#\langle h \rangle = n$; otherwise, $\pi = \pi'$, which contradicts the subgroup-decision assumption. More specifically, we let a signer generate π_Q such that $e(Q, \hat{h}) = e(g, \pi_Q)$, instead of π such that $e(\sigma_I, \hat{h}) = e(g, \pi)$, so as not to reveal the identity of I.

To fix the second problem, we introduce two more tokens, z and σ_{τ_2}. Let τ_0 be a random element in $\mathbb{Z}/n\mathbb{Z}$ and $z = \sigma_{x_I}(\tau_0)$. Let π_z be the proof mentioned above such that $e(z, \hat{h}) = e(g, \pi_z)$. We append τ_0 to the common reference string. We replace a public-key Y with $pk = (Y, z, \pi_z, \pi^{\mathsf{TRKGen}})$, where π^{TRKGen} is an NIZK proof for $(Y, \tau_0, z) \in \mathcal{L}_{BB}$. Let τ_1 and τ_2 be strings such that they are uniquely determined by tag, such that $\tau_1(tag) \neq \tau_2(tag')$ for every tag and tag'. We set $\sigma_I \triangleq \sigma_{x_I}(\tau_1)$, instead of $\sigma_{x_I}(tag)$. We let signer I produce $\sigma_{\tau_2} \triangleq \sigma_{x_I}(\tau_2)$ along with the associated proofs (to show that σ_{τ_2} is constructed appropriately), to append them to the signature. We modify the signature verification as the verifier rejects a signature w.r.t. ring R if there is $z^{(j)} = z^{(k)}$, $j \neq k$, with $z^{(j)} \in pk^{(j)} = R[j]$ and $z^{(k)} \in pk^{(k)} = R[k]$. We also modify the public tracing algorithm so that it may regard two signatures as independent if $\sigma_{\tau_2}^{(j)} \neq \sigma_{\tau_2}^{(k)}$, regardless of the fact that there is $I \in [N]$ such that $\sigma_I^{(j)} = \sigma_I^{(k)}$. Now assume that the malicious signers above, j and k, can create $\sigma_I^{(j)} = \sigma_I^{(k)}$ on the condition that

– $z^{(j)} = \sigma_{x_j}(\tau_0)$, $\sigma_j^{(j)} = \sigma_{x_j}(\tau_1)$, and $\sigma_{\tau_2}^{(j)} = \sigma_{x_j}(\tau_2)$.
– $z^{(k)} = \sigma_{x_k}(\tau_0)$, $\sigma_k^{(k)} = \sigma_{x_k}(\tau_1)$, and $\sigma_{\tau_2}^{(k)} = \sigma_{x_k}(\tau_2)$.

To break exculpability, they are now additionally required to produce $z^{(j)} \neq z^{(k)}$ and $\sigma_{\tau_2}^{(j)} = \sigma_{\tau_2}^{(k)}$. We stress that $\sigma_{\tau_2}^{(j)} = \sigma_{\tau_2}^{(k)}$ occurs only if $x_j = x_k$ (under the subgroup decision assumption). If so, it leads to $z^{(j)} = z^{(k)}$ and the signatures do not pass the signature verification test.

5.1 The Full-Fledged Scheme

We now provide the full-fledged scheme.

Common Reference String. $\mathsf{CRSGen}(1^k)$ does the following: Given 1^k, run the BGN generator Gen_{BGN} with 1^k to get $(\mathbb{G}, \mathbb{G}_T, n, p, q, e, g)$, where $\mathbb{G} = \langle g \rangle$ and $n = pq$. Pick up randomly hash $H : \{0,1\}^* \to \mathbb{Z}/n\mathbb{Z}$ from collision resistant hash family $\mathcal{H}_k$. Similarly, pick up randomly hash $H' : \{0,1\}^* \to \{0,1\}^{k-1}$ from collision resistant hash family $\mathcal{H}_{k-1}$. Pick up randomly $\gamma, \hat{\gamma} \leftarrow_R (\mathbb{Z}/n\mathbb{Z})^\times$ to set $h = g^\gamma$ and $\hat{h} = g^{\hat{\gamma}}$, which are also generators of $\mathbb{G}$. Pick up at random $\tau_0 \leftarrow_R \mathbb{Z}/n\mathbb{Z}$ and $Y_0, \sigma_0 \leftarrow_R \mathbb{G}$. Output as the common reference string, $crs = (\mathit{\Gamma}^*, Y_0, \sigma_0, \hat{h}, \tau_0, H, H')$, where $\mathit{\Gamma}^* = (\mathbb{G}, \mathbb{G}_T, n, e, g, h)$.

[3] We note that (σ, π) construct a simulatable verifiable random function [11] with *computational* verifiability on public parameter $(\mathit{\Gamma}^*, \hat{h}, Y)$.

Key Generation. $\mathsf{TRKGen}(crs)$ does the following: Given $crs = (\Gamma, Y_0, \sigma_0, \tau_0, \mathcal{H})$, pick up at random $x, d \leftarrow_R \mathbb{Z}/n\mathbb{Z}$ to compute $y = g^x$, $Y = yh^d$, $z = g^{\frac{1}{x+\tau_0}}$, and $\pi_z = \hat{h}^{\frac{1}{x+\tau_0}}$. Produce (adaptive unbounded) NIZK proof π^{TRKGen} in the common string crs that proves $(Y, \tau_0, z) \in \mathcal{L}_{BB}$. Output (pk, sk), where $pk = (Y, (z, \pi_z, \pi^{\mathsf{TRKGen}}))$ and $sk = (x, d)$. NIZK proof π^{TRKGen} is constructed as follows: The prover picks up at random $r, s \leftarrow_R \mathbb{Z}/n\mathbb{Z}$ and computes $C = Yh^r$ and $L = zh^s$. Then he produces NIWI proofs such that $(C, \tau_0, L) \in \mathcal{L}_{BB}$, $(C, (Y_0, Y)) \in \mathcal{L}_2^1$, and $(L, (\sigma_0, z)) \in \mathcal{L}_2^1$, respectively. He outputs (C, L) and these NIWI proofs.

Public Keys and Rings. We let $R = (pk_1, \ldots, pk_N)$ be an ordered set of public keys. We write R_Y to denote $(Y_1, \ldots, Y_N)$, where $pk_i = (Y_i, (z_i, \pi_{z_i}, \pi_i^{\mathsf{TRKGen}}))$.

Signature Generation. $\mathsf{TRSig}((x_I, d_I), crs, tag, m)$ does the following, where $tag = (issue, R)$ and $Y_I = g^{x_I} h^{d_I} \in R_Y$:

1. Hash $tag = (issue, R)$ to $\tau = H'(tag)$ of $(k-1)$-bit length string. Set $\tau_1 = 0||\tau$ and $\tau_2 = 1||\tau$ $(=2^{k-1} + \tau)$, so that $\tau_1 \neq \tau_2$ for every possible τ_1, τ_2.
2. Generate $(vk, sk) \leftarrow \mathsf{otGen}(1^k)$, which is a pair of verification and signing keys for a one-time signature scheme.
3. Compute the BB signatures: $\sigma_{vk} = g^{\frac{1}{x_I+vk}}$, $\sigma_I = g^{\frac{1}{x_I+\tau_1}}$, and $\sigma_{\tau_2} = g^{\frac{1}{x_I+\tau_2}}$.
4. Generate $\pi_{\tau_2} = \hat{h}^{\frac{1}{x_I+\tau_2}}$, which is the proof that the signer cannot make another σ'_{τ_2}, such that $\sigma'_{\tau_2} \neq \sigma_{\tau_2} \in \mathbb{G}$ and $(\sigma'_{\tau_2})^q = (\sigma_{\tau_2})^q \in \mathbb{G}_p$.
5. Commit to σ_{vk} by computing $L_{vk} = \sigma_{vk} h^{s_{vk}}$.
6. Compute $Q = (\sigma_I g^{-H})^{\frac{1}{I}} \in \mathbb{G}$, where $H = H(tag, m)$. For every $i \in [N]$, with $i \neq I$, compute $\sigma_i := g^H Q^i \in \mathbb{G}$ to set an ordered list $\Sigma = (\sigma_1, \ldots, \sigma_N)$. Note that it holds $\sigma_i = g^H Q^i$ for every $i \in [N]$.
7. Compute $\pi_Q = \hat{h}^{\chi}$ where $\chi = ((x_I + \tau_1)^{-1} - H) \cdot I^{-1} \bmod n$, where π_Q is the proof that the signer cannot make another Q' such that $Q \neq Q' \in \mathbb{G}$ and $Q^q = (Q')^q \in \mathbb{G}_p$.
8. Produce a NIZK proof π_{TRS} to prove $(R_Y, vk, \tau_1, \tau_2, L_{vk}, \Sigma, \sigma_{\tau_2}) \in \mathcal{L}_{TRS} \triangleq$

$$\{(R_Y, (vk, \tau_1, \tau_2), L_{vk}, \Sigma, \sigma_{\tau_2}) | \exists\, i \text{ s.t. } (Y_i, vk, L_{vk}), (Y_i, \tau_1, \sigma_i), (Y_i, \tau_2, \sigma_{\tau_2}) \in \mathcal{L}_{BB}\},$$

   where the soundness holds when the order of h is q. The proof is constructed as follows:

   – Commit to y_I, σ_I, and σ_{τ_2} in C, L, and L_{τ_2}, respectively.
   – Prove (C, τ_1, L), (C, vk, L_{vk}), (C, τ_2, L_{τ_2}) in $\mathcal{L}_{BB}$ in a NIWI manner, as described in Sec. 4.1.
   – Set $\hat{R_Y}$ as $(Y_0, Y_1, \ldots, Y_N)$ and $\hat{\Sigma}(I, tag, m)$ as $(\sigma_0, \sigma_1, \ldots, \sigma_N)$.
   – Then prove that $(C, \hat{R_Y})$ and $(L, \hat{\Sigma}(I, tag, m))$ in $\mathcal{L}_{N+1}^1$ in a NIWI manner (as described in Sec. 4.1 and Appendix B), by using witness $Y_{I'}$ and σ_I, respectively. In fact, $I' = I$ if the signer is honest. Finally, prove that $I' = I$ using the NIWI protocol as described in Appendix C.

9. Produce one-time signature $otsig = \mathsf{otSign}_{sk}(tag, m, vk, L_{vk}, Q, \pi_Q, \sigma_{\tau_2}, \pi_{\tau_2}, \pi_{TRS})$.
10. Output signature $sig = (vk, L_{vk}, Q, \pi_Q, \sigma_{\tau_2}, \pi_{\tau_2}, \pi_{TRS}, otsig)$ on message m with respect to $tag = (issue, R)$.

Signature Verification. $\mathsf{TRVrfy}(crs, tag, m, sig)$ does the following:

1. Retrieve $R = (pk_1, \ldots, pk_N)$ from tag, where $pk_i = (Y_i, (z_i, \pi_{z_i}, \pi_i^{\mathsf{TRKGen}}))$. Reject if there is $i \in [N]$ such that $e(z_i, \hat{h}) \neq e(g, \pi_{z_i})$, or π_i^{TRKGen} is invalid, or there is $i, j \in [N]$, with $i \neq j$, such that $z_i = z_j$.
2. Reject if $e(Q, \hat{h}) \neq e(g, \pi_Q)$ or $e(\sigma_{\tau_2}, \hat{h}) \neq e(g, \pi_{\tau_2})$.
3. Retrieve $\Sigma = (\sigma_1, \ldots, \sigma_N)$ from Q and $H = H(tag, m)$, by computing $\sigma_i = g^H Q^i$ for every $i \in [N]$.
4. Reject if $\mathsf{otVrfy}_{vk}((tag, m, vk, L_{vk}, Q, \pi_Q, \sigma_{\tau_2}, \pi_{\tau_2}, \pi_{TRS}), otsig) \neq \mathsf{accept}$.
5. Accept if the proof π_{TRS} is valid, otherwise reject.

Public Tracing. $\mathsf{Trace}(crs, tag, m, sig, m', sig')$ does the following:

1. Reject if $\mathsf{TRVrfy}(tag, m, sig) \neq \mathsf{accept}$ or $\mathsf{TRVrfy}(tag, m', sig') \neq \mathsf{accept}$, otherwise
2. Retrieve $vk, \Sigma = (\sigma_1, \ldots, \sigma_N)$ and σ_{τ_2} from the first signature, and vk', $\Sigma' = (\sigma'_1, \ldots, \sigma'_N)$ and σ'_{τ_2} from the second signature.
3. Accept if $\sigma_{\tau_2} \neq \sigma'_{\tau_2}$, or there is no $i \in [N]$ such that $\sigma_i = \sigma'_i$, otherwise,
4. Output linked if $m = m'$, otherwise
5. Output pk_i by choosing a random i from a set $\{i\} \subset [N]$ such that $\sigma_i = \sigma'_i$.

The proposed scheme satisfies the following theorems, whose proofs are given in the full version.

Theorem 2. *The proposed scheme is strongly tag-linkable under the subgroup decision assumption.*

Theorem 3. *The proposed scheme is anonymous if the 2^k-PR-DDHI assumption holds in $\mathbb{G}_p$ and $\mathbb{G}_q$.*

Theorem 4. *The proposed scheme is exculpable under the subgroup decision assumption, the strong DH assumption (in $\mathbb{G}_p$), and the unforgeability of one-time signature.*

6 The Second Proposal

Our second proposal is a tweak of our first proposal defined in a slightly different security model. The second scheme requires a unique bulletin board in which every party can make entries of tags. A tag id τ is an integer determined automatically in order of entries of tags into the bulletin board. As mentioned in Sec. 3.1, it is optional whether one can anonymously make an entry of a tag or not. The second proposal is the same as the first proposal except that a signer takes as input not a tag itself but the corresponding tag id, and set $\tau_1 := 2\tau$ and $\tau_2 := 2\tau + 1$, when producing a traceable ring signature. Since the total number of registered tags is at most polynomial in k, we can replace the 2^k-PR-DDHI assumption with the Q'-DDHI assumption, where Q' is the maximum number of queries that an (polynomially-bounded) adversary submits to signing oracles in total.

7 Conclusions

We presented the first two secure traceable ring signature schemes without random oracles. The proposed schemes have a signature size of $O(\sqrt{N})$, which is comparable to the most efficient ring signature schemes without random oracles, where N is the number of the ring members. The assumptions used for each scheme are well known in the paring-based cryptography, except for one. The first proposal requires the 2^k-PR-DDHI assumption, while the second one instead assumes the Q'-DDHI assumption for some polynomial Q', where the later assumption is an analogue of the Q'-DBDHI assumption [15]. Although the $O(2^k)$-PR-DDHI assumption is stronger than the $O(poly(k))$-DDHI assumption in general, the complexity of breaking the both assumptions are comparable in generic group attacks [25], which we provide in the full version. In addition, the second proposal requires a bulletin board.

Acknowledgments

We thank Sherman Chow who provided carefully considered feedback and a comprehensive reference about linkable ring signature schemes. We thank the anonymous reviewers of CT-RSA 2011 for their valuable comments. We thank Aggelos Kiayias, the program chair of CT-RSA 2011, for his kind help in revising the submission version of this paper.

References

1. Au, M.H., Chow, S.S.M., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 101–115. Springer, Heidelberg (2006)
2. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Constant-size ID-based linkable and revocable-iff-linked ring signature. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 364–378. Springer, Heidelberg (2006)
3. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006)
4. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

5. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
6. Boyen, X.: Mesh signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
7. 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)
8. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
9. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: STOC 1998, pp. 209–218 (1998)
10. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007)
11. Chase, M., Lysyanskaya, A.: Simulatable vRFs with applications to multi-theorem NIZK. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 303–322. Springer, Heidelberg (2007)
12. Chow, S.S.M., Susilo, W., Yuen, T.H.: Escrowed linkability of ring signatures and its applications. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 175–192. Springer, Heidelberg (2006)
13. Chow, S.S.M., Wei, V.K., Liu, J.K., Yuen, T.H.: Ring signatures without random oracles. In: ASIACCS 2006, pp. 297–302. ACM Press, New York (2006)
14. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in *ad hoc* groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
15. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
16. Fujisaki, E., Suzuki, K.: Traceable ring signature. IEICE Trans. of Fund. E91-A(1), 83–93 (2008); Presented in PKC 2007, LNCS 4450
17. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
18. 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)
19. Jeong, I.R., Kwon, J.O., Lee, D.H.: Analysis of revocable-iff-linked ring signature scheme. IEICE Trans. of Fund. E92-A(1), 322–325 (2009)
20. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
21. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004)
22. Liu, J.K., Wong, D.S.: Linkable ring signatures: Security models and new schemes. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005)

23. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
24. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 166–180. Springer, Heidelberg (2007)
25. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
26. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for E-voting, E-cash and attestation. In: Deng, R.H., Bao, F., Pang, H., Zhou, J. (eds.) ISPEC 2005. LNCS, vol. 3439, pp. 48–60. Springer, Heidelberg (2005)
27. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 384–398. Springer, Heidelberg (2004)

A Security Requirements of Traceable Ring Signature

Here we give the formal security definitions of tag-linkability, anonymity, and exculpability.

Global Setup. In our model, we allow for a global setup by an authority to produce a common reference string.

Common Setting. We assume there is a global public-key list maintained properly: A public-key should be referred to only one user. An adversary may append new public-keys to the global public-key list with dishonest users' names in any timing during the games of tag-linkability, anonymity, and exculpability. In the games, the adversary is basically allowed to choose arbitrary subsets in the global public-key list and make arbitrary ordered lists as the sub-rings when having access to signing oracles, called *the adversarially-chosen-key-and-sub-ring attack* due to [3], unless the choices of the subring cause the trivial attacks that the traceable ring signatures can never avoid.

(Strong) Tag-Linkability. The tag-linkability is to hold the robustness of a traceable ring signature scheme. Informally, it is measured as the inability of an adversary to create signatures, each of which are mutually independent, such that the number of them is more than the number of the signing keys in the ring that he owns. More precisely, an adversary is given a set of public keys $\hat{R}$ and allowed to append a polynomial number of new public keys to the global public-key list in any timing and may have access to signing oracle $\mathsf{TRSig}(\cdot, crs, \cdot, \cdot)$ with query of form (pk, tag', m'), where $pk \in tag[2] \cap \hat{R}$. Finally, he outputs a tag with ring R of N public keys, in which $N-k$ public-keys, $0 \leq k \leq N$, belongs to $\hat{R}$, and also outputs $k+1$ message/signature pairs, $(m^{(1)}, sig^{(1)})$, ..., $(m^{(k+1)}, sig^{(k+1)})$, such that $\mathsf{TRVrfy}(crs, tag, m^{(i)}, sig^{(i)}) = \mathsf{accept}$ for every $i \in [k+1]$. We say that the adversary wins if every signature outputted by him are mutually independent and they are also independent of every signature asked to the signing

oracle. If a traceable ring signature scheme stands the attacks, we say that the traceable ring signature holds (k, N)-tag-linkability. We say that a traceable ring signature scheme is strongly tag-linkable if it holds (N, N)-tag-linkability. If a traceable ring signature scheme holds exculpability, then (k, N)-tag-linkability implies $(k-1, N)$-tag-linkability. We will define a standard unforgeability for a traceable ring signature later. From the definitions, $(0, N)$-tag-linkability implies the unforgeability if exculpability holds. Namely, if a traceable ring signature holds tag-linkability and exculpability, it also holds the unforgeability, which we will prove later.

We formally restate the strong tag-linkability. Note that it is unnecessary for an adversary to take a set of public keys as input and to have access to the signing oracle in the strong tag-linkability. Let F be an adversary. It takes a common reference string crs and tries to output $tag = (issue, R)$ and $(N+1)$ message/signature pairs, $\{(m^{(i)}, sig^{(i)})\}_{i \in [N+1]}$, where $R = (pk_1, \ldots, pk_N)$. We define the advantage of F against TRS to be

$$\mathsf{Adv}^{\mathrm{sforge}}_{\mathsf{TRS}}(F)(k) \triangleq \Pr[\mathsf{Expt}_F(k) = 1]$$

where $\mathsf{Expt}_F(k)$ is: $crs \leftarrow \mathsf{CRSGen}(1^k)$; $tag, \{(m^{(i)}, \sigma^{(i)})\}_{i \in [N+1]} \leftarrow F(crs)$; return 1 iff $\mathsf{Trace}(tag, m^{(i)}, sig^{(i)}, m^{(j)}, sig^{(j)}) = \mathsf{accept}$ for all $i, j \in [N+1]$, where $i \neq j$.

Definition 5. *We say that* TRS *is strongly tag-linkable if for any non-uniform probabilistic polynomial-time (in* k*) algorithm* F*,* $\mathsf{Adv}^{\mathrm{sforge}}_{\mathsf{TRS}}(F)(k)$ *is negligible in* k*.*

Anonymity and exculpability are to protect honest user(s) from the rest of all players (plus the authority in the case of the untrusted common reference string model). In the games of anonymity and exculpability, an adversary is given the target public key(s) and allowed to append a polynomial number (in total) of new public keys to the global public-key list in any timing. Possibly, these public-keys can be related to the given target key(s). The adversary may have access to signing oracles in the manner of the adversarially-chosen-key-and-sub-ring attack [3], unless the choices of the subring cause the trivial attacks that the traceable ring signatures can never avoid.

Anonymity. Let D be an adversary. Let crs be a common reference string generated by CRSGen. Let pk_0, pk_1 be the two target public keys, where (pk_0, sk_0) and (pk_1, sk_1) are generated by $\mathsf{TRKGen}(crs)$. Let $b \in \{0, 1\}$ be a random hidden bit. D takes common reference string crs and the target public keys, pk_0 and pk_1, and outputs bit b' after doing the following a polynomial number of times in an arbitrary order: D may append new public keys to the global public-key list and have access to two types of signing oracles, $\mathsf{chTRSigO}$ and TRSigO, such that

– $\mathsf{chTRSigO}(crs, b, sk_0, sk_1, \cdot, \cdot)$ is the challenge signing oracle that takes query (tag, m), where $tag = (issue, R)$, and outputs sig generated by

$\mathsf{TRSig}(sk_b, crs, tag, m)$ if $pk_0, pk_1 \in tag[2] (= R)$, otherwise outputs $\perp$, where b is the hidden challenge bit.
– $\mathsf{TRSigO}(crs, sk_0, sk_1, \cdot, \cdot, \cdot)$ is the signing oracle that takes query (i, tag, m), where $i \in \{0, 1\}$, and outputs sig generated by $\mathsf{TRSig}(sk_i, crs, tag, m)$ if $pk_i \in tag[2] (= R)$, otherwise outputs $\perp$.

To avoid the trivial attacks, the following patterns of queries are prohibited:

– D submits two queries of different messages with the same tag, i.e., (tag, m) and (tag, m'), where $m \neq m'$, to $\mathsf{chTRSigO}$.
– D submits $(tag, \cdot)$ to $\mathsf{chTRSigO}$ and $(\cdot, tag, \cdot)$ to TRSigO.

We define the advantage of D against TRS to be $\mathsf{Adv}^{\mathrm{anon}}_{\mathsf{TRS}}(D)(k) \triangleq |p_1 - p_0|$, where $p_1 =$

$$\Pr\left[\begin{array}{l} crs \leftarrow \mathsf{CRSGen}(1^k); \\ (pk_0, sk_0) \leftarrow \mathsf{TRKGen}(crs); \\ (pk_1, sk_1) \leftarrow \mathsf{TRKGen}(crs) \end{array} : D^{\mathsf{chTRSigO}(crs, sk_1, \cdots), \mathsf{TRSigO}}(crs, pk_0, pk_1) = 1\right],$$

and $p_0 =$

$$\Pr\left[\begin{array}{l} crs \leftarrow \mathsf{CRSGen}(1^k); \\ (pk_0, sk_0) \leftarrow \mathsf{TRKGen}(crs); \\ (pk_1, sk_1) \leftarrow \mathsf{TRKGen}(crs) \end{array} : D^{\mathsf{chTRSigO}(crs, sk_0, \cdots), \mathsf{TRSigO}}(crs, pk_0, pk_1) = 1\right].$$

Definition 6. *We say that* TRS *is anonymous if, for every non-uniform probabilistic polynomial-time (in k) adversary D, the advantage* $\mathsf{Adv}^{\mathrm{anon}}_{\mathsf{TRS}}(D)(k)$ *is negligible in k.*

By definition, traceable ring signature schemes do not stand against attribution and full-key exposure attacks, defined in [3]. In the attribution (resp. full-key exposure) attacks, all but one secret key (resp. all secrets) in the ring are exposed to the adversary, which condition is incompatible with public traceability.

Exculpability. Let A be an adversary and let crs be a common reference string generated by CRSGen. Let pk be the target public key, where (pk, sk) is generated by $\mathsf{TRKGen}(crs)$. A takes crs and the target pk and does the following a polynomial number of times in an arbitrary order. A may append new public keys to the global public-key list and have access to signing oracle $\mathsf{TRSig}(sk, crs, \cdot, \cdot)$ with query $(\hat{tag}, \hat{m})$, where $pk \in \hat{tag}$. Finally, A outputs two message/signature pairs with the same tag, (tag, m, sig) and (tag, m', sig'), such that $m \neq m'$ and $pk \in tag$. We say that A wins if $pk = \mathsf{Trace}(tag, m, sig, m', sig')$ and at least one of the two messages, i.e., (tag, m) or (tag, m'), was not asked to TRSig. The advantage of A against TRS is defined as $\mathsf{Adv}^{\mathrm{frame}}_{\mathsf{TRS}}(A)(k) \triangleq$

$$\Pr\left[\begin{array}{l} crs \leftarrow \mathsf{CRSGen}(1^k); \\ (pk, sk) \leftarrow \mathsf{TRKGen}(crs); \\ (tag, m, sig, m', sig') \leftarrow A^{\mathsf{TRSig}_{sk}}(crs, pk) \end{array} : pk = \mathsf{Trace}(tag, m, sig, m', sig')\right]$$

Definition 7. *We say that* TRS *is exculpable if, for any non-uniform probabilistic polynomial-time adversary* A*,* $\mathsf{Adv}^{\mathrm{frame}}_{\mathsf{TRS}}(A)(k)$ *is negligible in* k*.*

We do not provide any additional security requirement on a traceable ring signature. One might think that some unforgeability requirement might be necessary. However, if tag-linkability and exculpability both hold, a reasonable definition of unforgeability also holds. We define unforgeability on a traceable ring signature as the inability of an adversary to forge a signature on a fresh message with respect to a tag such that he is not given any signing key of the tag, after he was given a set of public keys $\hat{R}$, he appended new public keys to the global public-key list, and he had access to the signing oracle with query (i, tag, m) to get $\mathsf{TRSig}(sk_i, \mathit{tag}, m)$, where $pk_i \in \hat{R}$, polynomial times, where a message being fresh means that the it has not been asked to the signing oracle.

Lemma 1. *If a traceable ring signature scheme is strongly tag-linkable and exculpable, then it is also unforgeable.*

Proof. Suppose for contradiction that there is an adversary A' against unforgeability. Let $(\mathit{tag}, m, \mathit{sig})$ be the output of A', where $\mathit{tag} = (\mathit{issue}, R)$, with $\#R = N$. Note that if $(\mathit{tag}, m, \mathit{sig})$ is linkable to some $(\mathit{tag}, m', \mathit{sig}')$ in the query/answer list between A' and the signing oracle, Trace outputs a public key in tag, because $m \neq m'$. It clearly implies breaking exculpability. Therefore, $(\mathit{tag}, m, \mathit{sig})$ should be independent of any $(\mathit{tag}, m', \mathit{sig}')$ in the query/answer list. However, it implies breaking strong tag-linkability or exculpability as follows.

Let F be a breaker of strong tag-linkability. F picks up an appropriate number of pairs of public and secret keys, more than N, and feeds all the public keys to A'. If A' submits (i, tag, m) to the signing oracle, F generates the signature using sk_i and replies it to A'. When A' outputs $(\mathit{tag}, m, \mathit{sig})$, F creates N mutually-independent signatures, $\{(\mathit{tag}, m^{(1)}, \mathit{sig}^{(1)}), \ldots, (\mathit{tag}, m^{(N)}, \mathit{sig}^{(N)})\}$, such that $m^{(i)} \neq m$ for all $i \in [N]$. Finally, F outputs $\{(\mathit{tag}, m^{(1)}, \mathit{sig}^{(1)}), \ldots, (\mathit{tag}, m^{(N)}, \mathit{sig}^{(N)}), (\mathit{tag}, m, \mathit{sig})\}$ if they are all mutually independent, otherwise halts.

Let A be a breaker of exculpability. A is given the target public key pk and allowed to have access to the signing oracle w.r.t. pk. A picks up an appropriate number of pairs of public and secret keys, and feeds all the public keys plus the target public key to A'. If A' submits (i, tag, m) to the signing oracle, A generates the signature by himself if i does not indicate the target public key, otherwise asks the signing oracle. In any case, A' can reply to A'. When A' outputs $(\mathit{tag}, m, \mathit{sig})$, A creates N mutually-independent signatures, $\{(\mathit{tag}, m^{(1)}, \mathit{sig}^{(1)}), \ldots, (\mathit{tag}, m^{(N)}, \mathit{sig}^{(N)})\}$, such that $m^{(i)} \neq m$ for all $i \in [N]$, by using the corresponding secret key or by asking the signing oracle. If $\mathsf{Trace}(\mathit{tag}, m^{(i)}, \mathit{sig}^{(i)}, m, \mathit{sig}) = pk$ for some $i \in [N]$, A outputs $(\mathit{tag}, m^{(i)}, \mathit{sig}^{(i)}, m, \mathit{sig})$, otherwise halts.

Since $m \neq m^{(i)}$, $\mathsf{Trace}(\mathit{tag}, m^{(i)}, \mathit{sig}^{(i)}, m, \mathit{sig})$ outputs accept, or a public key in the ring for any $i \in [N]$. Let p be the probability that F breaks strong tag-linkability of the traceable ring signature. Since the view of A' in the simulation of F is identical with that of A, it turns out that the probability that A can get some public key in the ring is $1-p$. Therefore, A can break exculpability at least $(1-p)/N$. Since either p or $(1-p)/N$ is non-negligible, the existence of A' contradicts strong tag-linkability or exculpability. □

Corollary 1. *If a traceable ring signature scheme is (k,N)-tag-linkable and exculpable, then it is $(k-1,N)$-tag-linkable, for any $1 \le k \le N$.*

The proof is omitted because it is substantially equivalent to the proof of Lemma [1].

B Sub-linear Size NIWI proof for $\mathcal{L}_L^1$

The following sub-linear size NIWI proof is an essential protocol in the CGS ring signature scheme [10]. Let $\mathcal{L}_L^1 = \{(C,R) \in \mathbb{G} \times \mathbb{G}^L \mid \exists i \in [L] \text{ s.t. } C_p = (Y_i)_p \in \mathbb{G}_p\}$, where $R = (Y_1,\ldots,Y_L) \in \mathbb{G}^L$. In the original CGS ring signature scheme, the form of a public-key is $y = g^x$, whereas we use $Y = g^x h^d$ as a public-key. The proof is, however, available for both types of public-keys.
Common Reference String: $\Gamma^* = (\mathbb{G}, \mathbb{G}_T, n, e, g, h)$.
Common Input: (C,R), where $R = (Y_1,\ldots,Y_L)$.
Witness to $\mathcal{P}$: $(I,r) \in [L] \times \mathbb{Z}/n\mathbb{Z}$ such that $C = Y_I h^r$.
What to Prove: $(C,R) \in \mathcal{L}_L^1$.

Prover $\mathcal{P}$

1. If integer L is not a square number, convert it to the least square number L' more than L and simply copy Y_1 many times to make a new ring R' of L' public keys. Since $L' < L + 2\sqrt{L} + 1$, it does not matter in the complexity sense. We set L' as L and R' as R. Let $L = m^2$.
2. Map $i \in [L]$ to $(i,j) \in [m] \times [m]$ (Regard R as $m \times m$ matrix). Let (u,v) be the correspondence to I.
3. Pick up at random $t_1,\ldots,t_{m-1} \leftarrow_R \mathbb{Z}/n\mathbb{Z}$ and set $t_m = -\sum_{i=1}^{m-1} t_i \pmod{n}$. Set $\alpha_u = gh^{t_u}$ and $\alpha_i = h^{t_i}$ for every $i \neq u$. Note that $\prod_{i=1}^{m} \alpha_i = g$.
4. Produce $\pi_{\alpha_u} = (gh^{t_u})^{t_u}$ and $\pi_{\alpha_i} = (g^{-1}h^{t_i})^{t_i}$ for every $i \neq u$. These are NIWI proofs to show that every α_i, $i \in [m]$ commits to either g_p or 1 [17,7]. By $\prod_{i=1}^{m} \alpha_i = g$, the verifier can be convinced that exactly one $\alpha \in \boldsymbol{\alpha}$ that contains g_p.
5. Pick up at random $w_1,\ldots,w_{m-1} \leftarrow_R \mathbb{Z}/n\mathbb{Z}$ and set $w_m = -\sum_{j=1}^{m-1} w_j \pmod{n}$. Set $\beta_v = gh^{w_v}$ and $\alpha_j = h^{w_j}$ for every $j \neq v$. Note that $\prod_{j=1}^{m} \beta_j = g$.
6. Produce $\pi_{\beta_v} = (gh^{w_v})^{w_v}$ and $\pi_{\beta_j} = (g^{-1}h^{w_j})^{w_j}$ for every $i \neq v$. These are also NIWI proofs to show that every β_j, $j \in [m]$, commits to either g_p or 1 [17,7]. By $\prod_{j=1}^{m} \beta_j = g$, the verifier can be convinced that exactly one $\beta \in \boldsymbol{\beta}$ that contains g_p.
7. For every $j \in [m]$, compute $A_j = \Pi_{i=1}^{m} e(\alpha_i, Y_{i,j})$ (One can see it as $\boldsymbol{\alpha} \cdot R$, in which the element-wise operation between α and Y is defined as $e(\alpha, Y)$). $A_j = e(g, Y_{u,j}) e(h, \prod_{i=1}^{m} Y_{i,j}^{t_i})$, which means that A_j commits to $Y_{u,j}$.

8. Pick up at random $\lambda_1, \ldots, \lambda_m \leftarrow_R \mathbb{Z}/n\mathbb{Z}$ to set $B_j = Y_{u,j}h^{\lambda_j}$. Let $\pi_{B_j} = g^{-\lambda_j}\prod_{i=1}^{m} Y_{i,j}^{t_{i,j}}$. Observe that $A_j = e(g, B_j)e(h, \pi_{B_j})$, which means that the prover can convince the verifier, by showing B_j and π_{B_j}, that $(B_j)_p = (Y_{u,j})_p \in \mathbb{G}_p$ for some unknown u.
9. Compute $C' = \prod_{j=1}^{m} e(B_j, \beta_j)$. Note that $C' = e(g, Y_{u,v})e(h, g^{\lambda_v}\prod_{j=1}^{m} B_j^{w_j})$ $= e(g, C)e(h, \pi_C)$, where $C = Y_{u,v}h^r$ and $\pi_C = g^{\lambda_v - r}\prod_{j=1}^{m} B_j^{w_j}$. Note that the prover can convince the verifier, by showing π_C, that C' commits to $(Y_{u,v})_p$ for unknown (u, v).
10. Output $(\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $(\pi_{\alpha_i})_{i\in[m]}$, $(\pi_{\beta_j})_{j\in[m]}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $(\pi_{B_j})_{j\in[m]}$, $\pi_C)$.

Verifier $\mathcal{V}$

1. For every $i \in [m]$, verify that $e(\alpha_i, g^{-1}\alpha_i) = e(h, \pi_{\alpha_i})$.
2. For every $j \in [m]$, verify that $e(\beta_j, g^{-1}\beta_j) = e(h, \pi_{\beta_j})$.
3. Verify $\prod_{i=1}^{m} \alpha_i = \prod_{j=1}^{m} \beta_j = g$.
4. For every $j \in [m]$, compute $A_j = \Pi_{i=1}^{m} e(\alpha_i, Y_{i,j})$ and verify A_j $= e(g, B_j)e(h, \pi_{B_j})$.
5. Compute $C' = \prod_{j=1}^{m} e(B_j, \beta_j)$ and verify $C' = e(g, C)e(h, \pi_C)$.
6. Accept if all the above steps are verified correctly; otherwise, reject.

C NIWI Protocol Such That Same $\boldsymbol{I}$ Is Used in Two NIWI Proofs for $\boldsymbol{\mathcal{L}_{(1,l)}}$

Let $R = (Y_1, \ldots, Y_l) \in \mathbb{G}^l$ and $\Sigma = (\sigma_1, \ldots, \sigma_l) \in \mathbb{G}^l$, where $l = m^2$. Let $\Pi_{C,R}$ be an NIWI proof for $(C, R) \in \mathcal{L}_l^1$ and let $\Pi_{L,\Sigma}$ be an NIWI proof for $(L, \Sigma) \in \mathcal{L}_l^1$, as described in Appendix B. What we further want to show is that $C = Y_I h^r$ and $L = \sigma_I h^s$. More precisely, we want to show $C_p = (Y_I)_p \in \mathbb{G}_p$ and $L_p = (\sigma_I)_p \in \mathbb{G}_p$.
Common Reference String: $\Gamma^* = (\mathbb{G}, \mathbb{G}_T, n, e, g, h)$.
Common Input: $(C, R, \Pi_{C,R}, L, \Sigma, \Pi_{L,\Sigma})$.
Witness to $\mathcal{P}$: $\boldsymbol{t} = (t_1, \ldots, t_m)$, $\boldsymbol{w} = (w_1, \ldots, w_m)$, $\boldsymbol{t'} = (t'_1, \ldots, t'_m)$, and $\boldsymbol{w'} = (w'_1, \ldots, w'_m)$, such that

– $\alpha_u = gh^{t_u}$, $\alpha_i = h^{t_i}$ for every $i \neq u$, $\beta_v = gh^{w_v}$, and $\beta_j = h^{w_j}$ for every $j \neq v$, where $i, j \in [m]$.
– $\alpha'_u = gh^{t'_u}$, $\alpha'_i = h^{t'_i}$ for every $i \neq u$, $\beta'_v = gh^{w'_v}$, and $\beta'_j = h^{w'_j}$ for every $j \neq v$, where $i, j \in [m]$,

where $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \Pi_{C,R}$ and $\boldsymbol{\alpha'}$, $\boldsymbol{\beta'} \in \Pi_{L,\Sigma}$.

Prover $\mathcal{P}$: Set $\gamma_i = \frac{\alpha_i}{\alpha'_i}$ and $\delta_i = \frac{\beta_i}{\beta'_i}$ for every $i \in [m]$. Compute $\pi_{\gamma_i} = g^{t_i - t'_i}$ and $\pi_{\delta_i} = g^{w_i - w'_i}$ for every $i \in [m]$, which proves that all γ_i and δ_i commit to 1. Output $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$.

Verifier $\mathcal{V}$: Verify $e(\gamma_i, g) = e(h, \pi_{\gamma_i})$ and $e(\delta_i, g) = e(h, \pi_{\delta_i})$ for every $i \in [m]$. Accept if all the above equations are verified correctly; otherwise, reject.

D Note: Insider Attacks on Linkable Ring Signature

In an insider attack, an adversary is given signing keys (except for that of a honest signer) of the target ring and, after seeing a ring signature on message m created by a honest signer, forges a ring signature on different message m', with $m' \neq m$, so that two signatures are linked. In a traceable ring signature scheme, it is obvious that this attack is a threat to a honest signer, because of traceability property. This attack is also problematic for a linkable ring signature scheme. Let us consider an election using a linkable ring signature scheme, in which a voter can vote only once for a candidate and the candidates who have collected votes more than a "borderline" are all elected. The following two ways of vote pigging are considered.

- A dishonest voter votes for two different candidates, A and B, to increase both candidates' votes (Of course, due to linkability, these two votes are linked).
- A dishonest voter who does not want candidate A to be elected, forges a signature on a vote for B such that it is intentionally linked with a signature on a vote for A created by a honest voter. His aim is to void a valid vote for A submitted by the honest voter.

Let us assume that one cannot distinguish the first case from the second one. We can achieve such a scheme by slightly modifying an existing scheme [21], and the modified scheme is still "proven secure" in the sense of the definitions in [21]. This election system is vulnerable. You must remove or accept both votes, but the intentions of the dishonest voters in the two cases above are opposite and you cannot see which case has occurred.

Author Index