

LNCS 6052

Radu Sion (Ed.)

Financial Cryptography and Data Security

14th International Conference, FC 2010
Tenerife, Canary Islands, Spain, January 2010
Revised Selected Papers



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Radu Sion (Ed.)

Financial Cryptography and Data Security

14th International Conference, FC 2010
Tenerife, Canary Islands, Spain
January 25-28, 2010
Revised Selected Papers



Springer

Volume Editor

Radu Sion
Stony Brook University
Computer Science Department
Stony Brook, NY 11794, USA
E-mail: sion@cs.stonybrook.edu

Library of Congress Control Number: 2010930773

CR Subject Classification (1998): E.3, D.4.6, K.6.5, K.4.4, C.2, J.1, F.2.1-2

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743

ISBN-10 3-642-14576-0 Springer Berlin Heidelberg New York

ISBN-13 978-3-642-14576-6 Springer Berlin Heidelberg New York

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

springer.com

© IFCA/Springer-Verlag Berlin Heidelberg 2010

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

This volume contains the main proceedings of the 14th Financial Cryptography and Data Security International Conference 2010, held in Tenerife, Canary Islands, Spain, January 25–28, 2010.

Financial Cryptography and Data Security is a major international forum for research, advanced development, education, exploration, and debate regarding information assurance, with a specific focus on commercial contexts. The conference covers all aspects of securing transactions and systems and especially encourages original works focusing on both fundamental and applied real-world deployments on all aspects surrounding commerce security.

Despite the dire economic climate as well as strong competition from other top-tier related security conferences, the Program Committee received 130 high-quality submissions and accepted 19 full-length papers (14.6% acceptance rate), 15 short papers (26.1% acceptance rate), 7 posters and 1 panel.

Three workshops were co-located with FC 2010: the Workshop on Real-Life Cryptographic Protocols and Standardization (RLCPS), the Workshop on Ethics in Computer Security Research (WECSR), and the Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC).

Intimate and colorful by tradition, the high-quality program was not the only attraction of FC. In the past, FC conferences have been held in highly research-synergistic locations such as Tobago, Anguilla, Dominica, Key West, Guadelupe, Bermuda, the Grand Cayman, and Cozumel Mexico. 2010 was the first year that the conference was held on European soil, on the Spanish Canary Islands, in Atlantic waters, a few miles across from Morocco. Over 100 researchers from more than 20 countries were in attendance.

Organizing a conference with such high standards was a true team effort. We would like to thank all those who made this possible: the International Financial Cryptography Association, the Program Committee and Proceedings Chair for their work, the Workshop Chairs, the keynote speakers and panel members, the local Arrangements Committee, and the authors and participants that made this such a exhilarating intellectually rich experience. Last but not least, we are thankful to our sponsors for their valuable support.

Ultimately, we hope this year's experience and quality research program will entice you to participate in Financial Cryptography 2011. We look forward to seeing you in Saint Lucia!

May 2010

Pino Caballero-Gil
Radu Sion

Organization

Organizing Committee

General Chair:	Pino Caballero-Gil	University of La Laguna, Spain
Program Chair:	Radu Sion	Stony Brook University, USA
Local Chair:	Candelaria Hernandez-Goya	University of La Laguna, Spain
Proceedings Chair:	Reza Curtmola	New Jersey Institute of Technology, USA
Poster Chair:	Peter Williams	Stony Brook University, USA

Local Organizing Committee

Luisa Arranz Chacon	Alcatel Espana, S.A.
Candido Caballero Gil	University of La Laguna
Amparo Fuster-Sabater	Instituto de Fisica Aplicada Madrid
Felix Herrera Priano	University of La Laguna
Belen Melian Batista	University of La Laguna
Jezabel Molina Gil	University of La Laguna
Jose Moreno Perez	University of La Laguna
Marcos Moreno Vega	University of La Laguna
Alberto Peinado Dominguez	University of Malaga
Alexis Quesada Arencibia	University of Las Palmas de Gran Canaria
Jorge Ramio Aguirre	Polytechnic University of Madrid
Victoria Reyes Sanchez	University of La Laguna

Program Committee

Ross Anderson	University of Cambridge, UK
Lucas Ballard	Google Inc., USA
Adam Barth	UC Berkeley, USA
Luc Bouganim	INRIA Rocquencourt, France
Marina Blanton	University of Notre Dame, France
Bogdan Carbunar	Motorola Labs, USA
Ivan Damgard	Aarhus University, Denmark
Ernesto Damiani	University of Milan, Italy
George Danezis	Microsoft Research, USA
Sabrina de Capitani di Vimercati	University of Milan, Italy
Rachna Dhamija	Harvard University, USA
Sven Dietrich	Stevens Institute of Technology, USA

VIII Organization

Roger Dingledine
Josep Domingo-Ferrer
Stefan Dziembowski
Simone Fischer-Hbner
Philippe Golle
Dieter Gollmann

Rachel Greenstadt
Markus Jakobsson

Rob Johnson
Stefan Katzenbeisser
Angelos Keromytis
Lars R. Knudsen
Wenke Lee
Arjen Lenstra

Helger Lipmaa
Javier Lopez
Luigi Vincenzo Mancini
Refik Molva
Fabian Monroe

Steven Murdoch
David Naccache
David Pointcheval

Bart Preneel
Josep Rifa Coma
Ahmad-Reza Sadeghi
Vitaly Shmatikov
Miroslava Sotakova
Angelos Stavrou
Patrick Traynor
Nicholas Weaver

The TOR Project, USA
University of Rovira i Virgili, Spain
University of Rome “La Sapienza”, Italy
Karlstad University, Sweden
Palo Alto Research Center, USA
Technische Universität Hamburg-Harburg,
Germany
Drexel University, USA
Palo Alto Research Center and Indiana
University, USA
Stony Brook University, USA
Technische Universität Darmstadt, Germany
Columbia University, USA
Technical University of Denmark, Denmark
Georgia Tech, USA
EPFL and Alcatel-Lucent Bell Laboratories,
Switzerland
Cybernetica AS, Estonia
University of Malaga, Spain
University of Rome “La Sapienza”, Italy
Eurecom Sophia Antipolis, France
University of North Carolina at Chapel Hill,
USA
University of Cambridge, UK
Ecole Normale Supérieure (ENS), France
Ecole Normale Supérieure (ENS) and
CNRS, France
Katholieke Universiteit Leuven, Belgium
Autonomous University of Barcelona, Spain
Ruhr University Bochum, Spain
University of Texas at Austin, USA
Aarhus University, Denmark
George Mason University, USA
Georgia Tech, USA
International Computer Science Institute
Berkeley, USA

Table of Contents

Constructive Cryptography — A Primer (Invited Paper)	1
<i>Ueli Maurer</i>	
Security Mechanisms with Selfish Players in Wireless Networks (Invited Paper)	2
<i>Jean-Pierre Hubaux</i>	
Users Do the Darndest Things: True Stories from the CyLab Usable Privacy and Security Laboratory (Invited Paper)	3
<i>Lorrie Faith Cranor</i>	
Multichannel Protocols to Prevent Relay Attacks	4
<i>Frank Stajano, Ford-Long Wong, and Bruce Christianson</i>	
A Traceability Attack against e-Passports	20
<i>Tom Chothia and Vitaliy Smirnov</i>	
Secure Computation with Fixed-Point Numbers	35
<i>Octavian Catrina and Amitabh Saxena</i>	
Implementing a High-Assurance Smart-Card OS	51
<i>Paul A. Karger, David C. Toll, Elaine R. Palmer, Suzanne K. McIntosh, Samuel Weber, and Jonathan W. Edwards</i>	
Unlinkable Priced Oblivious Transfer with Rechargeable Wallets	66
<i>Jan Camenisch, Maria Dubovitskaya, and Gregory Neven</i>	
Multiple Denominations in E-cash with Compact Transaction Data	82
<i>Sébastien Canard and Aline Gouget</i>	
What's in a Name? Evaluating Statistical Attacks on Personal Knowledge Questions	98
<i>Joseph Bonneau, Mike Just, and Greg Matthews</i>	
Cryptographic Protocol Analysis of AN.ON	114
<i>Benedikt Westermann, Rolf Wendolsky, Lexi Pimenidis, and Dogan Kesdogan</i>	
A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys	129
<i>Sven Schäge and Jörg Schwenk</i>	
Practical Private Set Intersection Protocols with Linear Complexity	143
<i>Emiliano De Cristofaro and Gene Tsudik</i>	

Design and Implementation of a Key-Lifecycle Management System	160
<i>Mathias Björkqvist, Christian Cachin, Robert Haas, Xiao-Yu Hu, Anil Kurmus, René Pawlitzek, and Marko Vukolić</i>	
Measuring the Perpetrators and Funders of Typosquatting	175
<i>Tyler Moore and Benjamin Edelman</i>	
A Learning-Based Approach to Reactive Security	192
<i>Adam Barth, Benjamin I.P. Rubinstein, Mukund Sundararajan, John C. Mitchell, Dawn Song, and Peter L. Bartlett</i>	
Embedded SFE: Offloading Server and Network Using Hardware Tokens	207
<i>Kimmo Järvinen, Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider</i>	
The Phish-Market Protocol: Securely Sharing Attack Data between Competitors	222
<i>Tal Moran and Tyler Moore</i>	
Building Incentives into Tor	238
<i>Tsuen-Wan “Johnny” Ngan, Roger Dingledine, and Dan S. Wallach</i>	
Tree-Homomorphic Encryption and Scalable Hierarchical Secret-Ballot Elections	257
<i>Aggelos Kiayias and Moti Yung</i>	
Automatically Preparing Safe SQL Queries	272
<i>Prithvi Bisht, A. Prasad Sistla, and V.N. Venkatakrishnan</i>	
PKI Layer Cake: New Collision Attacks against the Global X.509 Infrastructure	289
<i>Dan Kaminsky, Meredith L. Patterson, and Len Sassaman</i>	
Three-Round Abuse-Free Optimistic Contract Signing with Everlasting Secrecy (Extended Abstract)	304
<i>Xiaofeng Chen, Fangguo Zhang, Haibo Tian, Qianhong Wu, Yi Mu, Jangseong Kim, and Kwangjo Kim</i>	
Designing for Audit: A Voting Machine with a Tiny TCB (Short Paper)	312
<i>Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin</i>	
Attacking of SmartCard-Based Banking Applications with JavaScript-Based Rootkits (Short Paper)	320
<i>Daniel Bußmeyer, Felix Gröbert, Jörg Schwenk, and Christoph Wegener</i>	

Security Applications of Diodes with Unique Current-Voltage Characteristics (Short Paper)	328
<i>Ulrich Rührmair, Christian Jaeger, Christian Hilgers, Michael Algasinger, György Csaba, and Martin Stutzmann</i>	
Verified by Visa and MasterCard SecureCode: or, How Not to Design Authentication (Short Paper)	336
<i>Steven J. Murdoch and Ross Anderson</i>	
All You Can Eat or Breaking a Real-World Contactless Payment System (Short Paper)	343
<i>Timo Kasper, Michael Silbermann, and Christof Paar</i>	
Shoulder-Surfing Safe Login in a Partially Observable Attacker Model (Short Paper)	351
<i>Toni Perković, Mario Čagalj, and Nitesh Saxena</i>	
Using Sphinx to Improve Onion Routing Circuit Construction (Extended Abstract)	359
<i>Aniket Kate and Ian Goldberg</i>	
Secure Multiparty AES (Short Paper)	367
<i>Ivan Damgård and Marcel Keller</i>	
Modulo Reduction for Paillier Encryptions and Application to Secure Statistical Analysis (Extended Abstract)	375
<i>Jorge Guajardo, Bart Mennink, and Berry Schoenmakers</i>	
On Robust Key Agreement Based on Public Key Authentication (Short Paper)	383
<i>Feng Hao</i>	
A Formal Approach for Automated Reasoning about Off-Line and Undetectable On-Line Guessing (Short Paper)	391
<i>Bogdan Groza and Marius Minea</i>	
Signatures of Reputation (Extended Abstract)	400
<i>John Bethencourt, Elaine Shi, and Dawn Song</i>	
Intention-Disguised Algorithmic Trading (Short Paper)	408
<i>William Yuen, Paul Syverson, Zhenming Liu, and Christopher Thorpe</i>	
When Information Improves Information Security (Short Paper)	416
<i>Jens Grossklags, Benjamin Johnson, and Nicolas Christin</i>	
BetterThanPin: Empowering Users to Fight Phishing (Poster)	424
<i>Teik Guan Tan</i>	

Certification Intermediaries and the Alternative (Poster)	425
<i>Pern Hui Chia</i>	
SeDiCi: An Authentication Service Taking Advantage of Zero-Knowledge Proofs	426
<i>Ślawomir Grzonkowski</i>	
Poster Abstract: Security in Commercial Applications of Vehicular Ad-Hoc Networks	427
<i>Pino Caballero-Gil, Jezabel Molina-Gil, Cándido Caballero-Gil, and Candelaria Hernández-Goya</i>	
Domain Engineering for Automatic Analysis of Financial Applications of Cryptographic Protocols (Poster)	428
<i>Lilia Georgieva</i>	
hPIN/hTAN: Low-Cost e-Banking Secure against Untrusted Computers	429
<i>Shujun Li, Ahmad-Reza Sadeghi, and Roland Schmitz</i>	
Author Index	431

Constructive Cryptography – A Primer

Ueli Maurer

Department of Computer Science
ETH Zurich
CH-8092 Zurich, Switzerland
maurer@inf.ethz.ch

Abstract. A central paradigm in any constructive discipline is the decomposition of a complex system into simpler component systems or modules, which each may consist of yet simpler modules, and so on. This paradigm, sometimes called step-wise refinement, is useful only if the composition of modules is well-defined and preserves the relevant properties of the modules. For example, in software design, the composition operation must preserve correctness of the modules, i.e., a system consisting of correct modules must itself be correct.

In cryptography, the modules are cryptographic schemes (e.g. an encryption scheme or a message authentication code, MAC) or protocols (e.g. a zero-knowledge proof), and the composition must preserve the security of the modules. Surprisingly, for the traditional, game-based cryptographic security definitions, this composition property is unclear or at best highly non-trivial. Recall that a game-based security definition states that an adversary with certain capabilities (e.g. access to a MAC oracle) cannot win a certain game (e.g. forge a MAC) with non-negligible probability. One consequence of the lack of composability is that cryptographic protocols are often complex and lack modularity.

We propose *constructive cryptography* as a new paradigm, where the security definition of cryptographic schemes is radically different (though in many cases can be proved to be equivalent). For example, a message authentication scheme is defined to be secure if it *constructs* an authenticated communication channel from an insecure communication channel and a secret key, for a well-defined, simulation-based notion of “construct” and for well-defined definitions of an insecure and an authenticated channel. Similarly, a symmetric encryption scheme is defined to be secure if it constructs a secure communication channel from an authenticated communication channel and a secret key. The general composition property of this theory implies that the combination of a secure MAC and secure encryption scheme constructs a secure channel from an insecure channel and two secret keys (which can be constructed from a single secret key using a pseudo-random generator).

The security of public-key cryptosystems and digital signature schemes can be seen similarly in the constructive cryptography paradigm. In addition to making composition clear, the constructive cryptography approach has many other benefits. For example, it allows to investigate the intrinsic limitations of cryptography.

Security Mechanisms with Selfish Players in Wireless Networks

Jean-Pierre Hubaux

EPFL
Switzerland

<http://people.epfl.ch/jean-pierre.hubaux>

Abstract. It is frequently assumed that the parties involved in a security mechanism will behave according to everyone's expectation. However, some of them might be tempted to depart from the expected (or canonical) behavior, because such a deviation is more beneficial for them. As an illustration, we will consider that phenomenon in the framework of wireless networks. We will briefly introduce some basic background in game theory and provide an overview of several recent contributions to that field. Finally, we will consider two examples in more detail, namely revocation in high-mobility (or "ephemeral") networks and pseudonym change in mix zones.

Notes:

- Some of the material of this talk appears in the book “Security and Cooperation in Wireless Networks” by L. Buttyan and J.-P. Hubaux, Cambridge University Press, 2008, available at <http://secowinet.epfl.ch>
- A list of applications of game theory to various security (and cryptography) problems can be found at: <http://lca.epfl.ch/projects/gamesec>

Users Do the Darndest Things: True Stories from the CyLab Usable Privacy and Security Laboratory

Lorrie Faith Cranor

Carnegie Mellon University, Pittsburgh, PA
lorrie@cmu.edu

Abstract. How can we make security and privacy software more usable? The first step is to study our users. Ideally, we would watch them interacting with security or privacy software in situations where they face actual risk. But everyday computer users don't sit around fiddling with security software, and subjecting users to actual security attacks raises ethical and legal concerns. Thus, it can be difficult to observe users interacting with security and privacy software in their natural habitat. At the CyLab Usable Privacy and Security Laboratory, we've conducted a wide variety of studies aimed at understanding how users think about security and privacy and how they interact with security and privacy software. In this talk I'll give a behind the scenes tour of some of the techniques we've used to study users both in the laboratory and in the wild. I'll discuss the trials and tribulations of designing and carrying out security and privacy user studies, and highlight some of our surprising observations. Find out what privacy-sensitive items you can actually get study participants to purchase, how you can observe users' responses to a man-in-the-middle attack without actually conducting such an attack, why it's hard to get people to use high tech cell phones even when you give them away, and what's actually in that box behind the couch in my office.

Multichannel Protocols to Prevent Relay Attacks*

Frank Stajano¹, Ford-Long Wong², and Bruce Christianson^{3, **}

¹ University of Cambridge Computer Laboratory, Cambridge, United Kingdom

² DSO National Laboratories, Singapore

³ University of Hertfordshire, School of Computer Science, Hatfield, United Kingdom

Abstract. A number of security systems, from Chip-and-PIN payment cards to contactless subway and train tokens, as well as secure localization systems, are vulnerable to *relay attacks*.

Encrypting the communication between the honest endpoints does not protect against such attacks. The main solution that has been offered to date is distance bounding, in which a tightly timed exchange of challenges and responses persuades the verifier that the prover cannot be further away than a certain distance. This solution, however, still won't say whether the specific endpoint the verifier is talking to is the intended one or not—it will only tell the verifier whether the real prover is “nearby”.

Are there any alternatives? We propose a more general paradigm based on multichannel protocols. Our class of protocols, of which distance bounding can be modelled as a special case, allows a precise answer to be given to the question of whether the unknown device in front of the potential victim is a relaying attacker or the device with which the victim intended to communicate.

We discuss several instantiations of our solution and point out the extent to which all these countermeasures rely, often implicitly, on the alertness of a honest human taking part in the protocol.

1 Introduction

In a relay attack, the victims are two honest parties acting respectively as a prover (e.g. a door-opening token) and a verifier (e.g. a door-mounted token reader). In normal operation, when the prover (token) is authenticated by the verifier (door), the verifier grants some privilege (the door opens).

During a relay attack¹, a pair of communicating attackers splice themselves in the communication channel between the two victims. One of the attackers acts as a fake verifier to the victim prover and the other acts as a fake prover to the victim verifier. When the victim verifier issues a challenge, the attackers relay it unchanged to the victim prover; and when the prover issues its response to the original challenge, the attackers relay that too, unchanged, to the true verifier. The outcome is

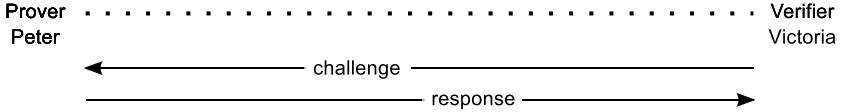
* Revision 39 of 2010-02-27 22:23:18 +0100 (Sat, 27 Feb 2010).

** On sabbatical at the University of Cambridge Computer Laboratory while the core of this research was carried out.

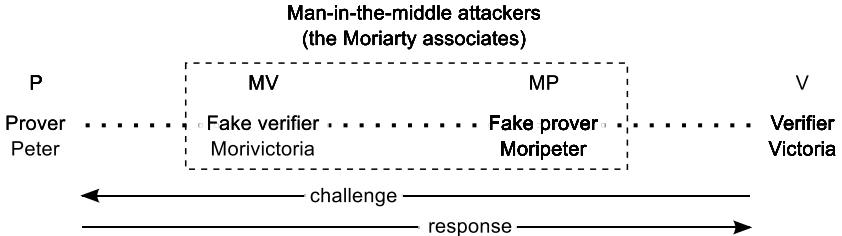
¹ Sometimes also called a *wormhole* attack, especially in secure localization contexts.

that the victim verifier grants the privilege to the fake prover, who was accepted thanks to the credentials unknowingly provided by the victim prover.

The honest participants²:



When a relay attack is taking place:



Even if the victim prover and verifier share a secret unknown to the attackers, they are still vulnerable: since their messages are relayed unchanged, the attackers succeed in fooling the verifier regardless of whether they can decrypt the messages they relay.

This problem has been known for several decades: Conway [6] described the “chess grandmaster problem”, in which an unskilled player defeats (or at least draws with) a chess grandmaster by simultaneously challenging *two* grandmasters at postal chess, one as white and one as black, and countering the moves of one grandmaster with those of the other. Beth and Desmedt [2] revisited the problem, noting that it matched the scenarios of the “mafia fraud”³ and “terrorist fraud”⁴, both previously described by Desmedt et al. [8], and they introduced

² We only show the essential core of the protocol here: clearly, in a more realistic situation, one would expect the protocol to be initiated by a preliminary request from Peter “hey, please challenge me so I can prove I’m worthy of getting the benefits”. We omit this and other non-essential messages for brevity and clarity.

³ In the mafia fraud, P is a customer who is electronically paying his restaurant bill to MV . Restaurant owner MV is a member of a mafia gang who alerts his accomplice MP to go and buy a diamond from jeweller V . Jeweller V challenges MP for his credentials, but MP and MV relay P ’s credentials to V . So P thinks he’s paying for a meal, whereas he is buying the mafiosi a diamond.

⁴ In the terrorist fraud, the verifier V is an immigration officer of country α and the fake prover MP is a terrorist who wants to enter the country. The fake prover MP is helped by P , a sympathetic citizen of α who supplies the correct answers to the questions of the immigration officer V . The main difference between this case and the mafia fraud is that the prover P is not a victim of the scam but an accomplice: he cooperates with the fake prover MP against the verifier V and therefore there is no need for a fake verifier MV .

the defensive technique of measuring the round-trip time, relying on the fact that the speed of light is finite to detect whether the actual prover is further away than expected. Brands and Chaum [3] refined that technique into a specific and more secure low-level protocol, with precomputation of single-bit challenges and responses that are then exchanged as quickly as the channel allows. More recently, Hancke and Kuhn [12] developed a distance-bounding protocol optimized for the resource-constrained RFID environment and, with colleagues [5], studied a variety of attacks on the timing measurements. Drimer and Murdoch [9] built electronic circuitry to demonstrate the relay attack⁵ against modern Chip-and-PIN bank cards and implemented the Hancke-Kuhn protocol to demonstrate its viability as a practical countermeasure. Hancke's doctoral dissertation [11] contains a good survey of the distance-bounding protocols in the literature.

The purpose of any distance-bounding protocol in such a context is to convince the honest verifier that the honest prover she is ultimately interacting with (the one who *can* respond to the challenges, whereas the attackers can't because they don't know the shared secret) is, with high probability, the prover currently in front of her. By construction, the distance bounding protocol can only give a verdict of the form "the owner of the shared secret just proved that he is no further away than d metres". If the verifier is interacting with a prover (whether genuine or fake) that is less than 1 metre away, but the distance bounding protocol says that he was unable to prove that he is within 10 metres, then the verifier should suspect that she is interacting with a relaying attacker.

Still, the distance-bounding solution does not really identify a specific principal but only its approximate location⁶. At least theoretically, depending on the spatial resolution of the distance-bounding protocol, it is still possible for attackers to go undetected if they stay within the bounds of the error margin, as in the scenario of multiple adjacent cash machines of which one is fake and performs a relay attack on another.

In this paper we propose a new paradigm for detecting and preventing relay attacks that is more general than distance bounding. Our strategy is to use a multichannel protocol [20,15,4,18,16] in which the traditional challenge-response between verifier and prover on the regular channel is augmented with an additional verification on a special channel whose main property is that it cannot be relayed.

Our multichannel approach includes the distance-bounding solution as a special case⁷. More importantly, our family of solutions includes ones that give a clear and definite "yes / no" answer to the question "is the principal in front of me really the one with whom I share this secret key, or is it just a middleperson attacker?", which the distance-bounding protocols can only answer with a less

⁵ With explicit reference to the "mafia fraud" scenario.

⁶ Within a sphere, or within the intersection of several spheres in the substantially more complicated case where one repeats the protocol from several reference points.

⁷ Insofar as you cannot relay beyond a certain distance the special channel implicitly defined by the distance-bounding procedure without being noticed by the victim endpoints.

stringent assurance such as “it probably is, provided there are no other principals within d metres of Victoria”.

Our approach also models the anti-relay alternative proposed by Damg  rd et al. [7] of somehow limiting the bandwidth with which the prover can communicate to the outside world to a value lower than the one needed in order to conduct the protocol—their arrangement implicitly relies on unrelayable channels because, by construction, at least one of the channels used in the protocol cannot be relayed to third parties outside.

We also highlight the extent to which all these anti-relay protocols, including both our new ones and the traditional ones based on distance bounding, implicitly rely on the presence of an honest human. We discuss whether they are still secure when the human takes part in the protocol without actively cheating but without thoroughly investigating all possible suspicious clues.

2 The Core Idea

Our core idea is that, although the man-in-the-middle attackers are usually able to relay the information between the two honest endpoints over whatever channels are normally used for the transaction, we might be able to augment the system with an additional special channel that the attackers won’t be able to relay. Over *that* channel, the two endpoints can verify whether they are talking directly to each other or not.

Traditionally, the authentication problem⁸ can be framed in the following terms: “I know I am talking to you; now, prove to me that you know our shared secret”. Here, instead, we examine the dual problem: “I know I am talking to someone who knows my shared secret; now, prove to me that you, the principal in front of me, are that someone”.

The intuition behind the multichannel approach is that the verifier asking that question should use the special channel to sample some physical aspect of the prover which the men in the middle are not able to relay, and then ask the prover (assumed to be honest and cooperative) to say, even over the regular channel subject to relay, what the correct value should be. Since prover and verifier already share a secret, they can use standard cryptographic techniques to protect the integrity (and confidentiality, though generally less relevant here) of the regular relay-vulnerable channel, thereby preventing the fake prover from replacing the true prover’s “model answer” with one matching the fake prover’s own physical aspect.

Since the fake prover can’t reproduce the true prover’s physical aspect (by hypothesis of unrelayability of the special channel) and can’t substitute the prover’s description with his own (because the regular channel is integrity-protected by the secret shared between the honest prover and verifier), the verifier can justifiably

⁸ According to our definition the authentication phase, which takes place repeatedly, is distinct from the preliminary “enrollment” or “pairing” phase, performed only once and under more controlled circumstances, in which the two principals establish a common secret.

deduce that the principal in front of it is the genuine prover if and only if the value sampled directly over the special channel is consistent with the one received over the integrity-protected channel. That's the core idea in a nutshell.

Looking at the problem in greater detail, the first issue is to define more precisely the “unrelayability” property, and the second is to clarify the subtle interactions between humans and their digital representatives in the course of the verification process: how much of the verification protocol can run unattended and how much of it does instead implicitly rely on human vigilance? We wish to make everything explicit.

Readers should note that using a multichannel protocol (such as acquiring a 2D barcode from a screen with a cellphone camera, as in the classic “Seeing Is Believing” protocol [15]) does *not*, by itself, prevent relay attacks. Without elaborate precautions, the auxiliary channel could itself be relayed⁹, which would totally negate its purpose. What we need is a multichannel protocol where one of the channels is by design *unrelayable*.

3 Unrelayable Channels and Protocols That Use Them

Our investigation of unrelayable channels brings to mind the work by Pappu et al. on unclonable “physical one-way functions” [17]:

These physical one-way functions are inexpensive to fabricate, prohibitively difficult to duplicate, admit no compact mathematical representation, and are intrinsically tamper-resistant.

To implement an unrelayable channel we require similar properties. In the context of a unidirectional channel in which a detector (sink) acquires information by sampling some physical aspect of an emitter (source), we need:

weak unclonability: it must be prohibitively difficult to produce a copy of a given source¹⁰;

strong unclonability: it must be prohibitively difficult to manufacture two indistinguishable sources¹¹;

⁹ For example, the on-screen barcode that Peter acquires with his cellphone could have been generated by Morivictoria by replicating the one acquired by Moripeter’s cellphone from Victoria’s screen.

¹⁰ Some will claim that this property is redundant because it is implied by each of the next two. But it is conceptually different and therefore we mention it as distinct to clarify the issues involved. By analogy, think of the source as a walnut. Weak unclonability means the attacker can’t produce another identical walnut. Strong unclonability means it’s infeasible for the attacker to produce any two walnuts that are indistinguishable. Unsimulability means the attacker can’t fool you by just showing you a photograph of your walnut.

¹¹ This would be analogous to a cryptographic “collision”. As with collision resistance, this clonability resistance property is *stronger* than the previous one, which it implies: if an attacker can’t make two identical sources of his own choice then a fortiori he can’t make a copy of a designated target source.

unsimulability: it must be prohibitively difficult to fool the sink by simulating the response of the genuine source using some other device¹²;

untransportability: it must be prohibitively difficult to manufacture a “data pipe” device capable of transporting to another location L the output of the source with sufficient fidelity that a sink at location L would not be able to distinguish whether it is sampling the genuine source or the output of the data pipe.

The unsimulability and untransportability requirements highlight the necessity of looking at the whole system, not just the source and sink endpoint devices, and of including the whole verification process in the evaluation. We must in particular clarify whether we are implicitly relying on the presence of a human verifier (e.g. to check that what is being sampled is the genuine artifact rather than, say, a box of electronics that simulates it, or a set of mirrors and prisms that reproduce its appearance) and the extent to which the overall unrelayability property depends on the care with which the human helper supervises the verification.

To help the reader follow the discussion, we shall now present several examples of unrelayable channels and associated protocols. They are not meant to be adopted as they are: take them as illustrations whose purpose is to help us think about the required properties of an acceptable solution.

To simplify matters, we deal with unidirectional authentication, with one prover and one verifier¹³. Prover and verifier are connected by a regular bidirectional channel, subject to relay attacks, and by a special unrelayable channel, which is unidirectional and goes from prover to verifier. The two principals have previously performed the pairing phase and therefore share a secret with which, using well-known cryptographic techniques, they can make the regular channel confidential and integrity-protected. Notation-wise, in the rest of this paper we shall say “lock X with K ”, written as $L_K(X)$, to mean “cryptographically protect both the integrity and the confidentiality of X using K as the key”, for example with encrypt-then-MAC.

With reference to our earlier figures, prover Peter must prove to verifier Victoria that the principal to whom Victoria is talking (and of whom Victoria can physically observe/measure/probe some physical aspect over the special channel) is Peter, i.e. the same principal that shares the secret with her. The attacker model is still that man-in-the-middle Moriarty has recruited two accomplices, Moripeter who looks like Peter and will try to fool Victoria, and Morivictoria who looks like Victoria and will try to fool Peter. Victoria wins if she can distinguish whether the principal to whom she is directly talking is Peter (who shares

¹² This property, too, implies the first one: if the attacker can't simulate the designated source using another device then a fortiori he can't make a clone of it.

¹³ We believe that what we really want in most practical applications is *mutual* authentication. For the moment, ignore possible optimizations and assume you can achieve mutual authentication by running the unidirectional protocol twice, once in each direction. Note however that this glosses over some subtle issues about the incentives of the two parties. We shall discuss them at the end of section 3.1.

a secret with her) or Moripeter (who doesn't). Conversely Moriarty wins if, after placing Moripeter next to Victoria, and Morivictoria next to Peter, he persuades Victoria that she is talking directly to Peter, even though she really isn't.

Normally, Victoria would run some kind of challenge-response protocol; she could for example ask Peter (or Moripeter, since she can't tell the difference yet) a question such as: "Here is a random nonce N . What do you obtain if you lock it with our shared secret K_{PV} ?". But, with a relay attack, Moripeter would relay the question to Morivictoria, who would ask the same question to Peter, who would provide the correct answer; then Moripeter would get the correct answer from Morivictoria and repeat it to Victoria, who would then be fooled into thinking that Moripeter knew the secret K_{PV} , whereas he didn't (and still doesn't).

3.1 Example: Banknote

In this first example, Peter's unrelayable physical characteristic is a banknote. The banknote is, by design, prohibitively difficult to duplicate (yielding weak and strong unclonability), and there are well-established methods for verifying that it is not a forgery.

Victoria now says, to the principal in front of her (Moripeter if they are under attack, or Peter under normal circumstances): "Give me a banknote."¹⁴ She checks that it's not a forgery (thereby reassuring herself that it is unclonable and that no duplicates of it exist) and then reads its serial number S and burns the banknote, making sure that that particular serial number will never be used again in any other run of this protocol¹⁵. Then she asks: "What do you obtain when you lock S , the serial number of the banknote you gave me, with our shared secret K_{PV} ?"

How can Moripeter answer that question? He could tell the serial number S to Morivictoria if it helped, but Morivictoria must run with Peter the same protocol as Victoria did with Moripeter (otherwise Peter would not respond), so she must ask for a banknote of that type from Peter, which will have a different serial number, say S_2 . Peter will lock that S_2 with the shared secret and there is no way that Morivictoria can persuade him to lock S instead, since

- the banknote is chosen by Peter; and, anyway,
- no other banknote exists with S on it: the only one that did was burnt.

So Moripeter will not be able to answer correctly and Victoria will be able to tell that she received the banknote from someone who didn't know the secret.

¹⁴ The banknote must be of a well-specified currency, issue and denomination, to avoid substitution attacks. To minimize the cost of each run of the protocol, it is OK for the banknote to be almost worthless—e.g. one from a country with runaway inflation—provided it is still unclonable. Alternatively, one might use the same technology as banknote printing to create low-value tickets with similar unclonability properties, as is sometimes done for concert or public transport tickets.

¹⁵ Burning the banknote at each protocol run makes S a nonce.

Attack: reverse pickpocketing. Now here is an attack: Moripeter and Morivictoria take a genuine banknote and make a counterfeit copy of it. The forgery is as good as it gets, but it is (by hypothesis of weak unclonability) detectable by someone who runs the proper checks. But, crucial point: Peter is the prover, not the verifier, so why should he be running any serious checks (UV light, colour-changing marker etc etc)? Do you do that on the banknotes you get from your cash machine, or as change from the supermarket? So the scam is for the Moriarty associates to “give” the forged banknote to Peter (as change in a transaction, or by letting him “find” it on the floor, or by reverse pickpocketing him, or whatever) and ensure that he will use it in the subsequent protocol run (no guarantee, but still non-negligible probability). The full run then goes as follows.

Victoria asks Moripeter for a banknote. He gives her the genuine banknote, with serial number S . She asks him to lock S with the shared secret K_{PV} , which Moripeter doesn’t know. Morivictoria asks Peter for a banknote. With some probability, she gets back the forged banknote that has the same serial number S : Peter didn’t check very carefully and never realized he had a forged banknote¹⁶ so he thinks he is handing over a genuine one. Morivictoria asks Peter to lock with K_{PV} the serial number of the banknote he just handed over; he obliges, and Morivictoria obtains $L_{K_{PV}}(S)$ which she relays to Moripeter who can then correctly answer Victoria’s challenge and pass off as Peter.

The lesson here is: who should be verifying the genuineness of the banknote? The prover or the verifier? And the correct answer is: both! If either of them doesn’t check with sufficient care, an attack is possible. (NB: if Victoria does not check that she is receiving a genuine banknote, the dual of the above scam, where Moripeter gives Victoria the forged banknote, works equally well.)

This attack scenario also highlights another systems issue we mentioned before: to what extent are we relying on humans to perform additional “implicit” sanity checks? Is it possible for the protocol to run with one machine talking to another machine, in unattended fashion¹⁷? Assume the crooked machine might exhibit a relaying artifact, e.g. a hi-res screen displaying the banknote, rather than the genuine article. In this case we see that we could in theory run this variant of the protocol in a machine-to-machine setting, provided that both the prover and the verifier contained the approved vending-machine-style technologies for checking that a banknote is not a forgery. Conversely, if we ran this protocol as person-to-machine (a human entering a high-security facility, or a human using an ATM), then it would fall upon the human to perform as careful a check of the authenticity of the banknote as the machine will do. In other words: we do indeed also need unsimulability and untransportability, as well as the strong and weak unclonability that we got from using a banknote!

¹⁶ And if Peter vaguely suspected it was a forgery, he was probably happy to get rid of it by using it in a protocol where it will be destroyed and none will be the wiser—that’s an interesting observation about the role of dishonesty in the psychology of scam victims [19] but let’s not get sidetracked for the moment.

¹⁷ Imagine for example a car interacting with a barrier, to enter a restricted zone or to pay a road toll or parking charge.

Attack: not burning the banknote. Here is another possible attack. Morivictoria asks Peter for a banknote, which he gives her. She pretends to burn it but instead she secretly passes it on to Moripeter. She also asks Peter to lock the serial number with K_{PV} , and she gives that answer to Moripeter as well. Now Moripeter can fool Victoria, using the genuine banknote and the $L_{K_{PV}}(S)$ kindly supplied by Peter! To prevent this, we must prevent Morivictoria from being able to reuse the banknote in other runs of the protocol. For example we could say she must cut it in half *and return it to Peter*¹⁸, all strictly under Peter’s nose¹⁹. The interesting problem, here, again, is that the strength of this countermeasure depends on the care with which Peter checks that he received the two halves of the same genuine banknote that he originally supplied—and not, for example, the two halves of a forgery, or of another banknote. But what’s Peter’s incentive for performing this check? If he is careless and the Moriarty associates succeed in their scam, they are fooling Victoria into opening her door (or giving away her diamond, or whatever) to Moripeter; does Peter lose anything? Not straight away, unless there are external liability issues that penalize Peter for fraudulent use of his authentication credentials. At the baseline level, though, it is Victoria’s security (not Peter’s) that depends on the care exercised by Peter, and this should be considered a vulnerability. Even though Peter is not actively dishonest, he may not go out of his way in order to protect Victoria, so long as he doesn’t lose anything himself by being slightly careless.

This attack scenario explains why we might want to develop a mutual authentication protocol in which the fate of the two parties is more closely entangled than it would be by simply running two instances of the unidirectional authentication protocol one after the other. The reason for wanting a mutual protocol is not to optimize and save on number of messages but rather to bind the incentives of the participants, so that if one of them is sloppy and the other careful then neither gets any benefit from the protocol run (as opposed to the unfair situation in which the sloppy principal is rewarded/protected because the other was careful, and the careful principal suffers because the other was sloppy).

3.2 Example: Accelerometers

In this rather different example, Peter and Victoria have 3D accelerometers that can record, at suitable resolution, a log of the accelerations to which they are

¹⁸ Returning the ashes isn’t as good, because Morivictoria might supply the ashes of another banknote and Peter would not be able to notice.

¹⁹ Otherwise another attack would be for Morivictoria to receive the banknote, go to the kitchen to fetch some scissors, pass Peter’s note to Moripeter who would then run the protocol by having it cut by the real Victoria; the two halves would be returned by Victoria to Moripeter, then to Morivictoria pretending to have just returned from the kitchen, then to Peter and neither Peter nor Victoria would be the wiser.

subjected. The accelerometers are stuck together and shaken randomly²⁰ and Victoria checks that the prover could observe the shake. The idea behind this is that “a random shake is unclonable”. The protocol runs as follows.

Victoria says: “Give me your accelerometer. Here is mine, too. I stick them together and shake them randomly for x seconds. Now have your accelerometer back. Please lock its log with our common secret and send it back to me.”

Attack: robotic arm. To comply with this request, Moripeter could observe Victoria’s shake (the challenge) with his accelerometer, give the precise details to Morivictoria from the accelerometer’s log and have Morivictoria reproduce that shake precisely in front of Peter. This last part is practically impossible for a person to do, hence our claim above that “a random shake is unclonable”. But what if Morivictoria has a high precision robotic arm that can reproduce the shake to within the required tolerances? Then Peter’s accelerometer would record a shake equivalent to that originally performed by Victoria, and Peter would lock it with the secret, and the Moriarty accomplices would win. So this highlights an implicit dependency on Peter being an “alert human” who would spot something amiss if Morivictoria’s arm were not of flesh and bones. (But would he actually pay attention to that detail? What if the arm were covered in clothes and appeared to come out of Morivictoria’s shoulder?) Thus a machine-to-machine version would not prevent relay attacks.

Attack: substituting, or tampering with, the accelerometer. Morivictoria could, by sleight of hand, substitute Peter’s accelerometer with one into which she downloaded the log communicated to her by Moripeter. No need for robotic arm, but the effect is again that of giving Peter a relayed log instead of the one of the real performance. To guard against this, Peter must ensure that the accelerometer he gets back is really his, and also that it hasn’t been tampered with (otherwise Morivictoria could upload the relayed log into Peter’s own accelerometer). Once again we raise the warning that we may be relying implicitly on the vigilance of a human Peter and that substitution or tampering might be possible in a machine-to-machine transaction.

Cameras instead of accelerometers. An alternative might be to monitor the shake with cameras, rather than accelerometers, the intention being that Peter’s cameras will never leave Peter’s trusted computing base and Morivictoria won’t be able to tamper with them. Victoria would then say, without reference to accelerometers: “I’ll shake the tip of my finger randomly for x seconds. Please lock the log of the 3D position of my finger with our common secret and send it back to me.” Setting aside the interesting but not security-critical computational geometry problem of comparing shake traces taken from different viewpoints, this solution would guard against the last two attacks (“substitute Peter’s accelerometer

²⁰ The technique of shaking together two objects instrumented with accelerometers was first proposed by Holmquist et al. [13] in the context of device pairing for ubiquitous computing. Later papers [14] perfected the necessary authentication protocols, taking into account error correction and so on.

with one containing relayed log” and “upload relayed log into Peter’s accelerometer”) but would still be subject to the “Morivictoria uses robotic arm” attack.

3.3 Example: Physical One-Way Functions

For this third example we use an instance of Pappu’s “physical one-way function”: a physical object with submicron features that are difficult to replicate exactly and that gives unpredictable but consistent “responses” when “challenged” (illuminated) with a laser. Peter holds the object (or *is* the object—think iris recognition) and Victoria challenges it. The protocol runs as follows.

Victoria shines her laser (in a random way R chosen by her, dictating parameters such as laser frequency, angle, scanning pattern etc) at Peter’s POWF object and she records the outcome $O_{Peter}(R)$. Then she tells Peter: “What is the response of your object when illuminated with R ? Lock the response under our secret and send it to me.”.

How can Moripeter answer that question? He will also have a POWF object, but by hypothesis of unclonability it must be different from Peter’s. Victoria records $O_{Moripeter}(R)$ and expects $L_K(O_{Moripeter}(R))$ but the Moriarty associates can only produce either $L_K(O_{Peter}(R))$, which has the wrong plaintext inside the outer brackets, or $L_{???}(O_{Moripeter}(R))$, where the correct plaintext is known but the correct key to lock it is unknown to Moriarty.

Attack: smoke and mirrors. In practice, the Moriarty associates could try to fool Victoria by having Moripeter use a more complex smoke-and-mirrors piece of machinery with its own lasers instead of a regular POWF object. Victoria chooses the laser parameters R and the Moriarty associates, through relay, use these same parameters to interrogate Peter’s genuine POWF. They record the response $O_{Peter}(R)$ and then make Moripeter’s smoke-and-mirrors machine respond with $O_{Peter}(R)$, rather than with anything physically generated, to Victoria’s laser challenge. Then Morivictoria asks Peter to lock the response with K , and she relays that to Moripeter, who convinces Victoria with an $L_K(O_{Peter}(R))$ that matches both the shared secret K and the response observed by Victoria.

The two assumptions upon which this attack is predicated are: first, that the Moriarty associates can build a smoke-and-mirrors machine capable of returning arbitrarily chosen laser responses regardless of the laser challenge with which it is illuminated; and second, that Victoria will just shine her laser in the prescribed way without noticing that she is interacting with a smoke-and-mirrors machine rather than with a POWF object. The first of these assumptions is fairly technology-dependent: it concerns the possibility of mounting a specific technical attack against a specific implementation. The second, instead, is once again related to the issue of whether a careful human supervisor will be overseeing the protocol or not²¹.

²¹ Note that “will be overseeing”—or, better, “is responsible for overseeing”—is quite different from simply “will be present”; in most cases a human will indeed be present, if nothing else to insert the card in the slot, but what matters here is whether the strength of the protocol depends on the degree of care that the human will exercise.

It should also be noted that in practice the attack is much harder than we casually described because Moripeter won't know Victoria's laser parameters R until Victoria actually shines the laser. There is no reason for Victoria to disclose R to the prover before shining the laser. If Victoria only discloses R after having received a laser response from the prover, then Moripeter must perform all of the following difficult tasks:

- figuring out R from the way Victoria shines the laser (instead of being told)
- reproducing those parameters at Morivictoria's end to challenge Peter
- obtaining Peter's POWF response
- relaying that back to Moripeter's smoke-and-mirrors machine

all in real time while Victoria is still operating. If the delay in Moripeter's answer makes Victoria suspicious then this is reminiscent of distance-bounding techniques (all essentially based on measuring whether the response takes longer than would be reasonable), even though conceptually we are still in a different territory. Note that it is very technology-dependent whether it is possible to (a) extract R while Victoria operates her laser and (b) relay the response piecemeal as it unwinds, rather than atomically at the end.

Note that we are now not really discussing the protocol: we are discussing whether or not the proposed special channel has the required unsimulability property.

3.4 Example: Quantum Channel (Polarized Photons)

This fourth example is even less practical than the previous ones but it is conceptually interesting, since it is based on the inherent unclonability of quantum mechanical states. We leave quantum mechanics to theoretical physicists and we just accept as a black box the assumptions (summarized in the next paragraph) of the BB84 Quantum Key Exchange protocol [1].

Under the assumptions of BB84, Alice the sender can emit photons at various polarization angles that are pairwise orthogonal (say 0, 45, 90, 135 degrees). Her encoding of 0s and 1s into these polarizations is important for BB84 but irrelevant for us. Bob the receiver cannot detect all the potential angles of the incoming photon: he must first choose one of two bases—either the rectilinear one that can distinguish between 0 and 90, or the diagonal one that can distinguish between 45 and 135. If he measures an incoming photon using a base that does not match the photon's polarization (for example measuring a 90-degree photon using the diagonal base), he will get an incorrect result (either 45 or 135, randomly). The photon is modified by the measurement; so, if eavesdropper Eve listens in on a photon with the wrong base, she "spoils" it for Bob.

We emphasize that we are *not* using (or describing) the BB84 protocol at all—only its underlying physical transmission medium. The BB84 protocol is for building a shared key between Alice and Bob, whereas in our scenario Victoria and Peter already share a key before we even start.

Our protocol runs as follows. Victoria produces a suitably long random string of the symbols {0, 45, 90, 135} and a matching string of the corresponding

polarization bases. She sends the second string (of bases) to Peter, locking it with the shared secret²², and then she sends Peter the actual polarized photons as described in the first string, which Peter can decode correctly by using the bases in the sequence he just received. Then it's Peter's job to send Victoria the string of values he read out, again locked with the shared secret. If Moripeter and Morivictoria splice themselves in, then when Moripeter listens to Victoria's photons he must choose a polarization base to receive each photon, but he won't know the right one because he could not unlock the first message, so he'll get it wrong about half the time and won't be able to tell Morivictoria the correct sequence of photons to retransmit to Peter. Therefore Peter will lock a different sequence of values and, even if they relay that, Victoria will be able to distinguish Peter from Moripeter.

Attack: relay the photons. An attack here would be for the Moriarty associates to run an optical fibre that shipped Victoria's photons to Peter, without being detected by either. If this were technically feasible, then the channel would lack the required property of untransportability and would not be suitable. However we are as usual assuming that Victoria is sufficiently alert that this attack cannot be mounted without attracting her attention: she would hopefully notice that (Mori)Peter has an extra optical fibre sticking out of the back of his coat.

Attack: extract the challenge. An over-elaborate and improbable attack sees Morivictoria use Peter as an oracle to check Morivictoria's guess of Victoria's locked sequence of bases. Victoria sends the locked sequence of bases to Moripeter. Morivictoria brute-forces it by trying each possible guess on Peter in turn, as described later. Once she has the correct guess about the bases she gives it to Moripeter, who uses to decode the real photons from Victoria. Morivictoria then sends the same sequence of photons to Peter, who provides the correct locked answer that they can relay to Victoria. (To check each guess, Morivictoria repeatedly sends Peter the same sequence of photons, polarized along the guessed base sequence; if the responses differ, then the guess was wrong, else the guess is shortlisted. She proceeds until only one guess is left.)

This attack relies on (a) the sequence being short enough that brute-forcing won't require years or millennia, (b) Victoria being patient enough to wait for the brute-force to take place between her first and second message, and (c) Peter being gullible enough to run the protocol as many times as requested without suspecting anything. It can be thwarted by having Peter include a nonce inside his locked answer so that it is different every time even if the sequence of values is the same²³.

3.5 Example: Quantum Channel (Entangled Photons)

The other seminal quantum cryptography protocol, E91 [10], uses a different underlying mechanism for quantum key establishment: an entangled pair of

²² Note that here we *are* using confidentiality, not just integrity.

²³ Of course this relies on the cryptographic implementation of “locking” not leaking information about the fact that two ciphertexts might correspond to plaintexts that share a long common portion.

photons. This mechanism, too, can be used to build another protocol in our family.

Under the assumptions of E91²⁴, some external source can prepare pairs of entangled photons and send one photon of the pair to Alice and one to Bob. Each photon can be measured using either a “blue” or a “red” machine and the outcome will be either 0 or 1. If Alice and Bob measure the two photons of an entangled pair using same-coloured machines, the outcomes will be the same; if they measure them with differently-coloured machines, they will be unrelated. Once again, we are not describing or using the E91 protocol—just its physical assumptions.

Our protocol runs as follows. Victoria generates n pairs of entangled photons and sends one photon from each pair to her correspondent (either Peter or Moripeter—she doesn’t know yet, but with our protocol she will be able to tell). Then Victoria sends Peter, over the standard channel, a random string of {red, blue} symbols—one for each of the entangled photons. Peter must then measure each photon with the machine of the specified colour and communicate the result to Victoria over the standard channel. Victoria performs the same measurements on her own photons and checks whether they match, which they should if there is no man in the middle.

In case of relay attack, Moripeter won’t be able to obtain the “challenge” string of reds and blues and therefore won’t be able to perform the correct measurements even though he has the genuine photons that are entangled with Victoria’s. Meanwhile Peter, who can perform the prescribed measurements, will be doing so on photons that are entangled with those of Morivictoria, not of Victoria, and therefore his answers won’t match those of Victoria, who will detect the difference.

Note how easy it is to specify and describe a protocol that won’t work, even if we can rely on seemingly all-powerful unclonable features such as entangled photons. Victoria generates n pairs of entangled photons and sends one from each pair to Peter. Then she also sends Peter, over the locked channel, a challenge consisting of a string of randomly chosen red and blue symbols. Peter must measure the entangled bits using machines of the prescribed colours and then report the answers to Victoria over the locked channel. But here the Moriarty associates relay the challenge from Victoria to Peter, let Peter do the measurements, relay the measurements from Peter to Victoria and appear indistinguishable from the case in which Peter answered directly.

Could you spot the subtle difference between this (broken) protocol and the almost identical one that instead works? Stop reading if you haven’t... In the working protocol, Victoria sends the photons to the guy in front of her; in the broken one, she sends them to “Peter”²⁵.

²⁴ Or rather its simplified description, by Ekert himself, at

<http://pass.maths.org.uk/issue35/features/ekert/2pdf/index.html/op.pdf>.

²⁵ The broken protocol is thus also impossible to implement: Victoria doesn’t know which principal is Peter (whole purpose of protocol); so how could she send *him* the photons?

Note that the “relay the photons” attack (cfr 3.4) applies to this setting as well, with the same caveats.

3.6 Why Our Multichannel Approach Works

The key insight of our approach is that the standard channel (think radio) connects Victoria to Peter (even if she doesn’t know where he really is) and that the special unrelayable channel connects Victoria to the principal in front of her. Victoria challenges Peter over the standard channel and Peter issues conceptually the same response over both channels. The Moriarty associates can only get it right on one channel at a time (they can relay the standard channel or they can “prove presence” over the unrelayable channel²⁶) but they can’t issue a consistent response over both. All the protocols shown so far are variations of this principle.

4 Conclusions and Further Work

We presented a novel paradigm: a family of multichannel protocols featuring a special channel that is unrelayable. We discussed the properties of unrelayable channels and illustrated possible channels and protocols with imaginative (if not always realistic) examples, chosen to explore the subtleties of the possible attacks, including the crucial role of the human principal in checking for unexpected hardware. We trust readers will recognize this framework as a conceptually new approach to developing protocols that prevent relay attacks.

What we need next is one or more robust and practical implementations of the unrelayable channel, using appropriate physical phenomena and transducers, and suitable protocols from this family to accompany them. Another useful contribution would be a formal analysis of the properties of these protocols.

We see great potential in this new line of authentication protocol research and hope that others will join us in bringing it to fruition for real-world applications.

References

1. Bennett, C., Brassard, G.: Quantum cryptography: Public-key distribution and coin tossing. In: Proc. IEEE ICCSSP (1984)
2. Beth, T., Desmedt, Y.: Identification Tokens — or: Solving the Chess Grandmaster Problem. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 169–176. Springer, Heidelberg (1991)
3. Brands, S., Chaum, D.: Distance-Bounding Protocols. In: Helleseth, T. (ed.) EU-ROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
4. Christianson, B., Li, J.: Multi-channel Key Agreement using Encrypted Public Key Exchange. In: Proc. Security Protocols Workshop 2007. LNCS, vol. 5964. Springer, Heidelberg (2007)

²⁶ . . . which, for all its wonderful properties, does not need to be particularly versatile: for example, you may not even be able to choose what bits the source will transmit!

5. Clulow, J., Hancke, G., Kuhn, M., Moore, T.: So Near and Yet So Far: Distance-Bounding Attacks in Wireless Networks. In: Buttyán, L., Gligor, V.D., Westhoff, D. (eds.) ESAS 2006. LNCS, vol. 4357, pp. 83–97. Springer, Heidelberg (2006)
6. Conway, J.: On numbers and games. Academic Press, London (1976)
7. Damgård, I., Nielsen, J.B., Wichs, D.: Isolated Proofs of Knowledge and Isolated Zero Knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 509–526. Springer, Heidelberg (2008)
8. Desmedt, Y., Goutier, C., Bengio, S.: Special Uses and Abuses of the Fiat-Shamir Passport Protocol. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 21–39. Springer, Heidelberg (1988)
9. Drimer, S., Murdoch, S.: Keep your enemies close: distance bounding against smart-card relay attacks. In: Proc. USENIX Security 2007 (2007)
10. Ekert, A.: Quantum cryptography based on Bell’s theorem. Physical Review Letters 67(6), 661 (1991)
11. Hancke, G.: Security of proximity identification systems. Tech. Rep. 752, University of Cambridge (2009)
12. Hancke, G., Kuhn, M.: An RFID Distance Bounding Protocol. In: Proc. IEEE Securecomm 2005 (2005)
13. Holmquist, L., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H.: Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) UbiComp 2001. LNCS, vol. 2201, p. 116. Springer, Heidelberg (2001)
14. Mayrhofer, R., Gellersen, H.: Shake well before use: Intuitive and Secure Pairing of Mobile Devices. IEEE Trans. Mobile Computing 8(6), 792–806 (2009)
15. McCune, J., Perrig, A., Reiter, M.: Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In: Proc. IEEE Security and Privacy 2005 (2005)
16. Nguyen, L., Roscoe, A.: Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey (2009) (manuscript)
17. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical One-Way Functions. Science 297(5589), 2026–2030 (2002)
18. Pavlovic, D., Meadows, C.: Deriving Authentication for Pervasive Security. In: Proc. ACM ISTPS 2008 (2008)
19. Stajano, F., Wilson, P.: Understanding scam victims: seven principles for systems security. Tech. rep. 754, University of Cambridge (2009)
20. Wong, F., Stajano, F.: Multi-channel Protocols. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2005. LNCS, vol. 4631, pp. 112–127. Springer, Heidelberg (2007); See also the extended and revised version in IEEE Pervasive Computing 6(4), 31–39 (2007)

A Traceability Attack against e-Passports

Tom Chothia* and Vitaliy Smirnov

School of Computer Science, University of Birmingham, Birmingham, UK

Abstract. Since 2004, many nations have started issuing “e-passports” containing an RFID tag that, when powered, broadcasts information. It is claimed that these passports are more secure and that our data will be protected from any possible unauthorised attempts to read it. In this paper we show that there is a flaw in one of the passport’s protocols that makes it possible to trace the movements of a particular passport, without having to break the passport’s cryptographic key. All an attacker has to do is to record one session between the passport and a legitimate reader, then by replaying a particular message, the attacker can distinguish that passport from any other. We have implemented our attack and tested it successfully against passports issued by a range of nations.

1 Introduction

New technologies lead to new threats. Traditionally security protocols have been analysed for a range of security and authenticity goals, however the introduction of small, promiscuous Radio Frequency Identifier (RFID) tags have raised new concerns. For instance, can a person’s movements be traced using the RFID tags that have been inserted into the items they are carrying? As RFID tags will respond to any signal broadcast to them, and originally replied with a unique identifier, Benetton’s proposal to place RFID tag in clothes caused a public outcry for precisely this reason [BB]; similar traceability concerns have also affected the New York area E-Zpass system [Cal]. Now RFID tags are being placed in passports.

The use of RFID tags in passports was primarily motivated by the desire to provide storage for bio-metric information such as fingerprints or iris scans [ICA06]. A suite of cryptographic protocols protects the data on the tag. Read access to the data on the passport is protected by the *Basic Access Control* (BAC) protocol. This protocol produces a session key by using another key derived from the date of birth, date of expiry and the passport number printed on the document. The aim of this protocol is to ensure that only parties with physical access to the passport can read the data. All data on the tag is signed by a document signing key which is in turn signed by a country key from the state that issued it. The public country verification keys are publicly available from the International Civil Aviation Organisation (ICAO)¹. This process of

* This work is partly supported by EPSRC grant EP/F033540/1: Verifying Interoperability Requirements in Pervasive Systems.

¹ Currently at <https://pkddownloadsg.icao.int>

ensuring the integrity of the data is referred to as *Passive Authentication*. A third protocol, *Active Authentication*, ensures that the passport has not been copied by signing a nonce (a new random number) from the reader, using a signing key stored securely on the tag. The verification key, signed by the issuing country, can then be read from the tag and the passport verified by the reader. Both BAC and Active Authentication are specified as optional although BAC seems to be universally used². We only observed Active Authentication on a few of the passports we looked at (e.g. the Irish passport).

In 2006 a second generation of e-passports were announced [ICA06] which included a new *Extended Access Control* protocol that would establish a session key based on a longer secret and would authenticate the reader to the tag using the country signing keys. This protocol would be run after the BAC protocol. A third generation of e-passport protocols are currently under discussion [BG08], although they have not yet been finalised by the ICAO.

The BAC protocol ensures that the data on the e-passport can only be read by someone who knows the key derived from the date of birth, date of expiry and number on the passport. Our attack lets someone who does not know this key trace a passport, i.e., if an attacker can observe a run of a particular passport then they can build a device that detects whenever the same passport comes into range of the reader. RFID tags receive their power via a signal from the reader; FCC regulations [FCC] limit the power of the readers, leading to an effective range of about 9cm. However, if the attacker disregards these regulations, they can power up the tag from a much greater distance, Kfir and Wool calculate that this is possible from a distance of up to 50cm [KW05]. If another reader powers the tag up, messages can be sent to and received from a tag to a range of several meters [Yos04, Han06]. This would make it easy to eavesdrop on the required message from someone as they used their passport at, for instance, a customs post. Furthermore, the RFID tags in passports are “always on” and give no indication to their owner that they are sending data.

A traceability attack does not lead to the compromise of all data on the tag, but it does pose a very real threat to the privacy of anyone that carries such a device. Assuming that the target carried their passport on them, an attacker could place a device in a doorway that would detect when the target entered or left a building. Juels et al. [JMW05] point out, rather melodramatically, that such an attack would make it possible to program a bomb that would explode in the presence of a particular person. More benignly, it could also be used to make a device that would tell a blind person whenever someone they had met before was close by. Such tracing attacks may also apply to other contactless devices. However, we believe that a traceability attack against e-passports is particularly severe because unlike, for instance, Bluetooth devices they cannot be turned off and also because a passport is a government mandated identity document and carrying one is compulsory when crossing a border or when resident in certain countries.

² Early US and Belgian e-passports did not have BAC, however BAC is now implemented.

The BAC protocol was closely based on ISO 11770-2 mech. 6 [ISO96]. It sets up a secure session key that the reader then uses to access the data. During a run of the BAC protocol, the passport generates a nonce that the reader must encrypt using the passport's unique encryption key. This ensures that messages are not being replayed to the passport. The reader and passport also generate Message Authentication Codes (MACs) for each message, using the passport's unique MAC key. This guarantees that the messages are received correctly and the MAC is checked before the nonce is looked at. This protocol protects the data on the passport, as any replayed or corrupted message will be rejected.

Our examination of actual passports has shown that it is possible to tell the difference between a message that was rejected because of an incorrect nonce and a message that was rejected because of a failed message authentication check. To trace a passport we eavesdrop on a legitimate session between a passport and a reader, and record the encrypted message that contains the passport's nonce. Then, when we want to identify a particular passport, we replay this message. If this replayed message is rejected because the MAC check failed then we know this is not the same passport, as the MAC key is unique to each passport. On the other hand, if the message is rejected because of the nonce check failed, we know that the MAC check using the unique passport key succeeded and therefore we have found the same passport again. In the case of the French passport different error messages are given in response to a failed MAC or an incorrect nonce. In the case of all other nationalities we tested, the rejection messages are the same but a failed MAC check is reported noticeably sooner than a failed nonce.

Many authors (e.g. [JMW05, CLRPS06, AKQ08]) have pointed out that the entropy used to seed the BAC keys is low, and in the case of countries where passport numbers are partly predictable it may be possible to guess the keys. However, passports are now being issued with a passport number made up of letters and numbers, rather than just numbers, which will increase the possible key entropy. It has also been pointed out that once a reader is given access to a passport it cannot be revoked [JMW05]. Richter et al. [RMP08] showed that the error messages issued by a passport were different for each country and so it was possible to uniquely identify the nationality of a passport drawn from a group of 10 European countries. Ours is the only attack on e-passports that allows an attacker to remotely trace an individual passport, in real-time, for any passport numbering scheme, without having to know the BAC keys.

Our attack has a relatively simple fix; the error messages issued by the passports must be standardised and response times must be padded so as to remove the information leak. One way to do this would be to make e-passports decrypt messages even if the MAC check fails. For the tens of millions of passports already issued it is too late, however future passports can be made safe.

In the next section we describe the protocols used by e-passports and discuss other analysis of these protocols in Section 2.2. We present a protocol based attack against the French e-passport in Section 3 and extend this to a timing attack against all e-passports in Section 4. We discuss ways in which this attack may be stopped and conclude in Section 5.

2 The e-Passport Protocols

An e-passport³ is an identification document combining a traditional passport with an RFID tag capable of performing cryptographic operations, storing biometric data and other bearer related information. The specification for e-passports is published by the International Civil Aviation Organization (ICAO) [ICA06] and more than 60 states have started issuing their own e-passports based on this standard.

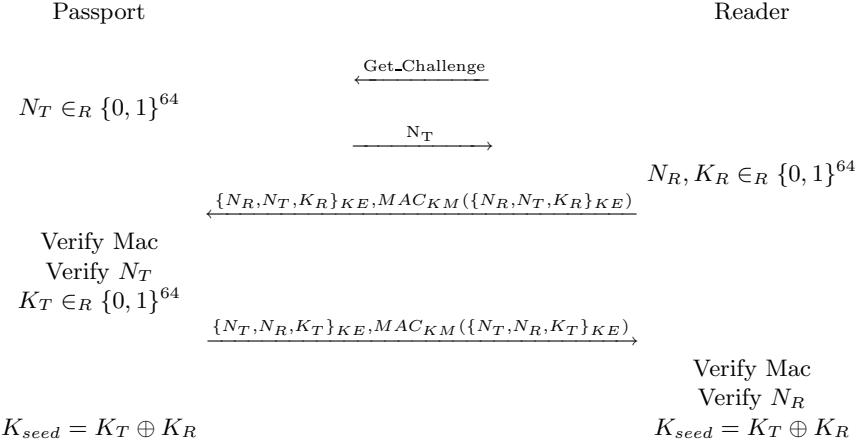
The ICAO specification requires that passports use the contactless card standard ISO 14443 [ISO01] for hardware level communication. This standard defines how the reader should power up the card and select a particular tag to communicate with; if more than one tag is present, each card broadcasts a unique ID and the reader selects one, with which to establish a session. The ICAO specification recommends that the UID is randomised to avoid the possibility of it being used to trace a particular passport [ICA08, page 22]. If a country chooses to ignore this advice, then a passport will be easily traceable. All the passports we have looked at, so far, use randomised UIDs. ISO 14443 defines two ways in which radio signals can be used to communicate with the cards (Type A and Type B). E-passports may implement either method.

On top of the ISO 14443 communication, the ICAO specification states that the passports should implement some of the commands and error codes defined in the standard for contact-based smart cards ISO 7816 [ISO95]. As well as giving a detailed description of the layout of the data on the passport, it specifies that the passport should support the ISO 7816 commands SELECT FILE and READ BINARY for accessing the data on the tag. The instructions GET CHALLENGE, MUTUAL AUTHENTICATION and INTERNAL AUTHENTICATION are used for BAC and Active Authentication. The passports also use ISO 7816 error codes, such as “6A80: Incorrect parameters” or “6300: No information given”.

2.1 The Passport Protocols

The data on the passport is organised into 16 *data groups*, that can be read using the ISO 7816 SELECT FILE and READ BINARY commands. The ICAO specification defines what each data group should be used for: DG1 and DG2 are compulsory for all passports and store the machine-readable data printed on the passport and the passport photo respectively. DG3 to DG16 are for optional data, such as fingerprints (DG3, which we found on a recent German passport). The contents of some of these data groups have been defined but are not yet used in practice, such as iris scans (DG4), holder’s signature (DG7) and the address of someone to contact in an emergency (DG16). Data groups 11 and 12 are for optional additional information depending on the country, for example,

³ For the rest of this document we will use “passport” to mean “e-passport”, rather than a passport without an RFID tag, and only use e-passport when we want to underline the difference between the two.

**Fig. 1.** The Basic Access Control Protocol

the French passport uses these to store the height⁴ of the passport holder, their home address and the address of the police station where the passport was issued. According to the specification, the data groups are read-only. The hash of the data groups, which has been signed by the issuing state, is stored on the passport; checking this ensures that the passport is not forged.

Read access to the data on the passport is protected by the *Basic Access Control* protocol (BAC). This protocol uses a key generated from the date of birth, date of expiry and passport number printed on the passport and establishes a new session key to protect all following communication with the reader. The aim of this protocol is to prevent eavesdropping and skimming attacks by ensuring that only someone who has seen the information page of the passport can access the data on the tag. While other authors have criticised this design as less secure than, say, making the reader authenticate to the tag using a certificate, it does have the advantage of allowing moderately skilled users to see what is on their own passport.

BAC is a key establishment protocol, as shown in Figure 1. Here $\{-\}_K$ denotes Triple-DES encryption with the key K and $MAC_K(\cdot)$ denotes a cryptographic checksum according to ISO 9797-1 Message Authentication Code Algorithm 3. The passport stores two keys: KE and KM , and the reader derives these keys using the machine-readable information on the passport, which has, in theory, been scanned before the wireless communication begins.

The reader initiates the protocol by sending a challenge to the tag and the tag replies with a random 64-bit string N_T . The reader then creates its own random nonce and some new random key material, both 64-bits. These are encrypted,

⁴ We found cases where a French passport overestimated the height of its owner, this seems to be because the height measurement is not checked by the passport issuing organisation and so reflects the height that the passport holder would like to think of themselves as, rather than their true height.

along with the tag’s nonce and sent back to the reader. A MAC is computed using the *KM* key and sent along with the message, to ensure the message is received correctly.

The tag receives this message, verifies the MAC, decrypts the message and checks that its nonce is correct; this guarantees to the tag that the message from the reader is not a replay of an old message. The tag then generates its own random 64-bits of key material and sends this back to the reader in a similar message, except this time the order of the nonces is reversed, in order to stop the reader’s message being replayed directly back to the reader. The reader checks the MAC and its nonce, and both the tag and the reader use the xor of the key material as the seed for a session key, with which to encrypt the rest of the session.

This protocol guarantees that only parties who know the keys derived from the machine-readable zone can learn the session key and message freshness is guaranteed by the nonces. However, we observe that this protocol does not guarantee a fresh session key to the reader: as the passport picks its key material after it sees the reader’s key material, and the material is xor-ed together, the passport may pick its material in such a way as to force a particular key seed. Although this does not seem to lead to an attack, concatenating the key material would have meant that both parties were guaranteed a fresh key.

Active Authentication is an optional protocol designed to prevent cloning attacks. The protocol is based on public key cryptography; the tag proves the possession of a private key with a straightforward challenge-response protocol. If the passport supports the Active Authentication protocol, the public key is stored in Data Group 15, which is signed along with the rest of the passport data. In 2006, the ICAO proposed a new set of protocols called Extended Access Control (EAC). These protocols are commonly used to protect sensitive biometric data, and require the reader to authenticate itself to the passport using a certificate signed by a country signing key. We observed EAC on a recent German passport, where it was used to protect fingerprints, and information on the EAC parameters was stored in data group 14. Both Active Authentication and EAC are optional and run after BAC, so, as our attack is against BAC, the additional security these protocols provide does nothing to stop our attack.

2.2 Related Work

Many papers have been written about the e-passport specification. One of the most popular themes is the low entropy of the BAC key seed. The original ICAO documentation points out that the ideal entropy of 73-bits is probably closer to 56-bits due to non-random passport numbers. A series of authors have then analysed the passport numbers of particular countries. For instance, Juels et al. [JMW05] pointed out the US passport only offers 54-bits of entropy, Carluccio et al. [CLRPS06] put the German passport’s entropy at 55-bits, and Avoine et al. [AKQ08] put the Belgian passport at 38-bits. Most of these authors go on to assume that the attacker knows the birthday of their victim and so subtract another 15-bits from the key entropy. We note that all of these calculations are

based on the assumption that the random part of the passport numbers only contain digits. This is no longer true: the passport number on German passports issued since, at least, 2008 include letters as well as numbers. Therefore, the entropy is now likely to be much higher than Carluccio et al. estimate.

The Belgian passports have such low entropy because the passport numbers are mostly numeric and issued sequentially, Avoine et al. show that an eavesdropping attack can find the key in about a second, whereas an online attack against only a passport could take a few weeks, in the worst case. Carluccio et al. [CLRPS06] and Liu et al. [LKLRP07] both present hardware architectures that can speed up the cracking process, however they also assume that the attacker has some previous knowledge about the victim, such as their birthday and has observed a correct run of the protocol. In contrast to this work, our attack is an attack on the protocol itself, rather than an attack against the weak key seed. We do not need to assume that the attacker knows the age of the victim and our attack works, in real-time against any passport numbering scheme.

Hoepman et al. [HHJ⁺06] also discuss the low BAC entropy and point out that a passport would be traceable if it does not randomise its ISO 14443 UID. All the passports we have looked at do randomise their UIDs, although we have been told that passports from Italy and New Zealand do not.

Perhaps the most similar work to ours is that of Danev et al. [DHBv09] who show that a passport can be identified by its hardware characteristics with an error rate of 2% to 4%. However, to collect their readings they must place the passport in a specially constructed wood frame, therefore they suggest they that their method is better suited to detecting counterfeit passports than it is to tracing people.

2.3 Experimental Framework

To interact with the passports we used an ACR122U reader from Advanced Card Systems Limited. This is one of the cheapest (~\$50) RFID readers on the market and while more expensive reader could collect more accurate timing data and performed tests faster, using such a reader underlines the fact that our attack does not need specialist hardware.

Adam Laurie's RFID Input/Output Tools (RFIDiot) project [Lau06] has developed a number of tools to make interacting with RFID tags easy. We found these tools very useful when initially experimenting with e-passports, and we have made use of Laurie's libraries when writing the code to perform our attack.

We ran our tests with passports volunteered by members of our lab and their families. We tested 10 passports in total: 3 UK, 2 German, 1 Russian, 2 French, 1 Irish and 1 Greek. We would like to extend our thanks to all of the volunteers that offered their passports for testing, and we were particularly pleased that no country had chosen to make their passports lock up after a set number of failed runs of the BAC protocol.

When taking a large number of time samples from a continuously powered passport we noticed that after around 100 readings in a row the response times from the passport would start to slow down by about 1ms every 20 readings. To

RFID tag	ATR value
UK Passport	3B898001097877D4020000900048
French Passport	3B8E80011177B3A7028091E16577010103FF61
Irish Passport	3B848001043833B1BB
German Passport, (numric passport number, no fingerprints), German Passport (alpha-numeric passport number, fingerprints)	3B8E8001107833D4020064041101013180FFBD
Dubai Metro pass	3B898001097877C4020000900058
Mifare (e.g. Oyster card, Univ. Id)	3B8F8001804F0CA0000003060300030000000068 3B8F8001804F0CA000000306030001000000006A

Fig. 2. ATR values from various RFID tags

ensure that our sampled data was independent and identically distributed we powered down the tag between each time measurement.

2.4 Passport FingerPrinting via Answer to Reset

While the ICAO defines the specification for e-passports, all of the countries we have looked at have built their own implementations. Richter et al. [RMP08] exploit this fact, to show that it is possible to deduce which country issued a passport by the error messages it gives. They also mention other possible ways to detect the issuing country of a passport including the ISO 14443 “Answer to Select” or “File Control Information” message. We also found that the passports of different nations gave distinctive error messages, however we received different error messages to the ones reported by Richter et al., this may have been due to using different parameters in the ISO 7816 commands.

Contact-based ISO 7816 chips will respond to a reset with an “Answer to Reset” (ATR) message, which includes data on the chip’s manufacturer and how the chip should be read. In the interests of compatibility, the Interface Device Handler (the firmware and/or drivers) for contactless card readers construct an ATR message for ISO 14443 tags [Wor07, Sec. 3.1.3.2.3]. These handler constructed ATR messages have a standard prefix, followed by the historical data from the “Answer to Select” for ISO 14443 Type A tags, or the application data and protocol information for ISO 14443 Type B tags. Furthermore, this constructed ATR message is generated when the reader initiates contact with the tag, and is therefore much easier to find than a complete set of error codes.

Out of the passports we tested, we found that each country had its own unique constructed ATR value, we also found that a range of mifare classic cards all issue the same ATR, see Figure 2. The German passport was recently updated to include an alpha-numeric passport number and the fingerprints of the owner. We found that these updated passports had a different ATR to the earlier version. Therefore, the ATR provides an easy way to identify, not just the issuing nation, but also the version of a passport. This is an additional weakness in the passport because if it is possible to narrow down the issue date of a passport it becomes easier to guess the BAC key. Some of the observed ATRs were very close so,

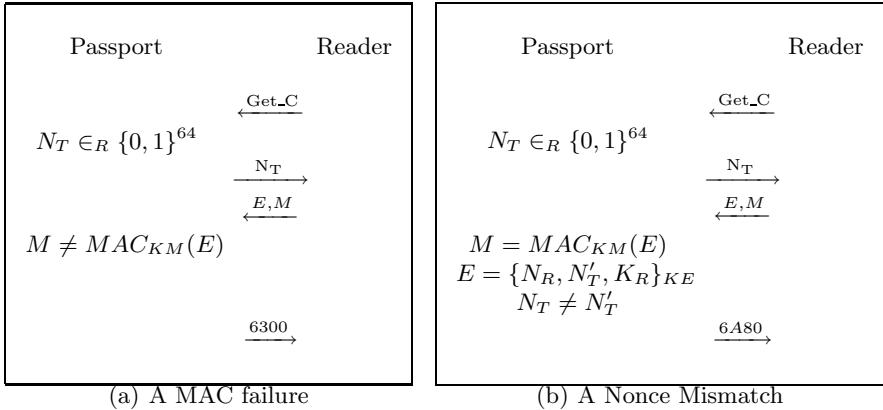


Fig. 3. The Basic Access Control Protocol

just as with error messages, there is a possibility of two different tags having the same profile. Hence, further research is needed before we can be sure that this is a good identification technique.

3 An Attack against French e-Passports

The ICAO passport specification states that the passport must always respond to a message, returning an error message if the message was incorrect or unexpected. The fault in the French passport’s BAC protocol becomes apparent when we consider the error messages that the passport generates in response to erroneous messages from the reader.

To find these error messages we power up the passport, according to ISO 14443, we then send a GET CHALLENGE message to initiate the BAC protocol to which the passport replies with a nonce. The reader should send the tag’s nonce back to the passport, along with some keying material and its own nonce. This message should be encrypted with the passport’s unique encryption key and sent with a MAC generated using the passport’s unique MAC key. To find the error messages we tried broadcasting a message to the tag with an incorrect MAC, and found that the French passport replied with a “6300: No information given” error (Figure 3(a)). Next we formed a message with a correct MAC but with an incorrect nonce. This message was replied to with a “6A80: Incorrect parameters” error (Figure 3(b)).

These different error messages can be used to trace a passport, even by an attacker that does not have the passport encryption and MAC keys. First the attacker must observe a run of the passport with a reader that knows the passport key, for instance, while going through customs. The attacker records the message from the reader that contains the encrypted and MACed nonces and keying material. Later, when the attacker comes across another passport, they can use this recorded message to test if it is the same passport as they observed before: the

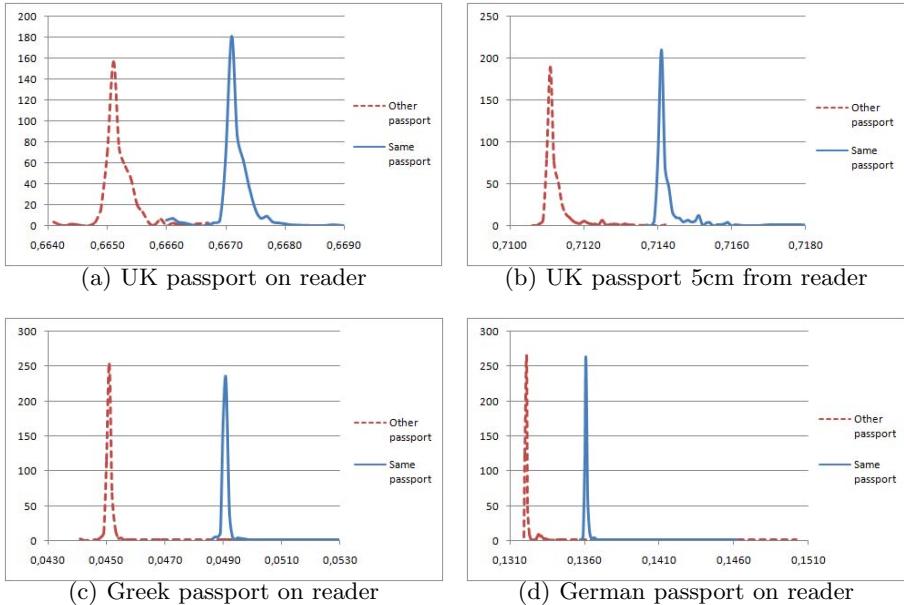


Fig. 4. Sampled Times from Replaying a Message to the Same or a Different Passport

attacker broadcasts a GET CHALLENGE message, to which the tag responds with a nonce. The attacker then replays the message they recorded from the previous run. If the tag responds with a 6300 error message then we know that the MAC check failed, therefore the passport we are currently looking at used a different MAC key from the original passport and is not the same one. If, on the other hand, we get a 6A80 message then we know that the MAC check must have succeeded, and so the current passport is the passport we are trying to trace.

4 A Time-Based Traceability Attack

Out of all the passports we tested, only the French passport responded to a failed MAC check and a mismatched nonce with different error messages; all the other passports issued the same error code, usually “6300”. So it seemed that this attack only affected French passports. However, examining the passports further, we noticed that the time it took for a passport to issue these error messages was not constant.

Figure 4(a) shows the time it took for a UK passport to issue the error message (to 4 decimal places). We sent 500 messages we knew would fail the MAC check (shown in dashed, red) and 500 replayed messages, with the correct MAC key, but with an incorrect nonce (shown in solid, blue). It is clear from this data that a failed MAC elicits a reply more quickly than a failed nonce. Looking at the protocol specification, it seems that this is because the passport rejects a message with an incorrect MAC straightaway, whereas if the MAC is correct, the MAC

check is performed, the message is then decrypted and only after that can the nonce be checked. The additional time it takes to reply to a replayed message is the time it takes the passport to decrypt the message and check the nonce. After checking several passports, we found that the exact time difference depended mainly on which country issued the passport. For our particular reader, UK passports took around 2.8 milliseconds longer to respond to a replayed message, German, Greek and Irish passports took 4ms to 5ms and a Russian passport we tested took a sluggish 7ms.

We retested a UK passport, this time placing the passport 5cm away from the reader (Figure 4(b)). This data set clearly shows the time difference between a message replayed to the passport that generated it and a message replayed from a different passport. However, placing the passport away from the reader leads to all the messages taking longer. The time it takes the radio waves to cross the extra distance is of the order of 10^{-10} seconds so this slowdown is most likely explained by less power being supplied to the RFID tag. Such variations in response times mean that it is not possible to trace a passport with a single replayed message. Instead, the attacker must send a message they know will fail the MAC check, then send the replayed message and compare the response times.

The exact attack could be performed in a number of different ways. If a passport is known to be stationary then the attacker could send one completely random message and then replay the message from the passport they wish to trace. If the time difference is more than some value the attacker could decide that it is the same passport as before, and if it is less than that value the attacker could decide that it is a different passport. This test could be repeated for additional accuracy, the attacker could also use different lower and upper bounds, or attempt to work out the nationality of the passports via the ATR (as described in Section 2.4) and then pick the most efficient cutoff for that country. When the passport is moving it is necessary to send a number of different random messages interleaved with a number of replayed messages and then take the average. We find the error rates and efficiencies of these different methods using a statistical analysis of the response times.

Statistical Analysis of Passport Response Times. The response times in Figure 4 appear to follow a normal distribution. Due to the limited accuracy of our measuring framework, we round our data to 4 decimal places. This makes our data discrete by placing the results into a number of bins, (e.g. all time measurements between 0.66505 and 0.66515 are placed in the 0.6651 bin). Therefore we can verify that the data is well modelled by a normal distribution using a χ^2 goodness of fit test. This test defines a test statistic:

$$X = \sum_{i=1,\dots,k} \frac{(O_i - E_i)^2}{E_i} \quad (1)$$

where O_i is the observed number in bin i and E_i is the number predicted by the distribution. The sampled data is well modelled by a normal distribution if the X statistic is consistent with a $\chi^2_{(k-3)}$ distribution (see e.g. [SC89]). We carried out

this test and found that the X statistic was within the 95% confidence interval for the British, German, Greek and Irish passports, both when the passport is directly on the reader or when placed 5cm away from it. We note that this does not mean that the distribution is exactly normal, but rather it means that a normal distribution is a reasonable model for the sampled data and is therefore useful in order to estimate the error rates.

The Russian e-passport was not consistent with a normal distribution. The time graphs for a 100 samples are given in Figure 5 (only 100 samples were taken due to limited access to the passport). As well as not following a normal distribution, the passport would not let us access any data after we have performed BAC, which suggests that the passport might not be fully compatible with the ICAO standard (EAC, if used, should only protect bio-metric data). Information on the Russian passport specification is sparse, and mostly in Russian (see e.g. [Min03, Eva05]), so this calls for further study. The time gap between random and replayed messages was the biggest we have seen for any passport and with no overlap at all; therefore our attack would work against Russian passports with a very high degree of certainty.

Looking at the timings that follow a normal distribution, we can calculate the rates of false positives and false negatives for particular tests. We know that the difference between a value from a distribution $\mathcal{N}(m_1, v_1)$ and a value from the distribution $\mathcal{N}(m_2, v_2)$ will come from the distribution $\mathcal{N}(m_1 - m_2, v_1 + v_2)$. Therefore, the difference in response times in milliseconds, for a random message and a message replayed to the same UK passport it came from will come from the distribution $\mathcal{N}(2.8, 0.63)$, whereas the difference in response times for a different passport, one that did not generate the message being replayed, would come from the distribution $\mathcal{N}(0, 0.62)$. The distributions of these differences for all of the different passports are shown in the first 2 columns of Figure 6.

A false positive occurs when we test a different passport and decide that it is the one that generated the message we are replaying, whereas a false negative occurs when we test the passport that generated the message we are replaying

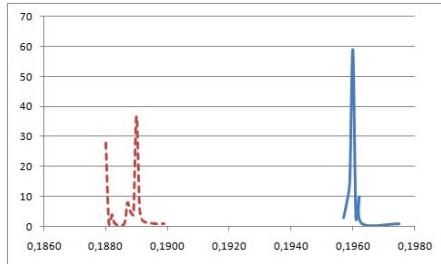


Fig. 5. Russian Sample Times

Passport Country	Same Passport	Different Passport	Prob. False Pos. at 1.7ms	Prob. False Neg. at 1.7ms
UK	$\mathcal{N}(2.8, 0.63)$	$\mathcal{N}(0, 0.62)$	0.015	0.084
German	$\mathcal{N}(3.9, 0.124)$	$\mathcal{N}(0, 0.52)$	0.009	0.024
Greek	$\mathcal{N}(4.0, 1.57)$	$\mathcal{N}(0, 1.21)$	0.061	0.033
Irish	$\mathcal{N}(5.2, 0.79)$	$\mathcal{N}(0, 1.52)$	0.084	0.00004

Fig. 6. Distribution of Time Differences and the Error Rates

but fail to identify it as the same passport. The simplest test is for the attacker to send one random message and one replayed message. Using the distributions in Figure 6 we calculated that if the attacker decides that it is the same passport when the time difference is more than 1.7ms and a different passport when the difference is less than 1.7ms, then the worst false positive probability is 0.084 and the worst false negative rate is 0.084. If the attacker repeats this test, taking the *best out of 3* the false positive and negative probabilities fall to 0.02 and for the *best out of 5* the error rates are 0.005.

If the attacker decides that the passport is the same when the difference is more than 2.8ms, a different passport when the difference is less than 1.0ms and runs another test when the difference is in between these values, we find that the probability of a false negative is 0.011 and a false positive is 0.012 and the expected number of trials is 4.8. This suggested that, for the passports we tested, the most efficient test, that balances the false positives and false negative, is to use 1.7ms as a cutoff and running extra trials to get the accuracy required. We implemented this test, using the best out three, and wrote a program that tested a passport against a database of replay messages from each of the 10 passports we examined, in turn. In 20 tests our program correctly identified every passport from the time delay of its replay message.

To test the feasibility of our attack against a moving target we tried taking a number of readings from a passport while it was moved across the reader. We averaged these readings, and found that some readings would take up to a few seconds and have a disproportional effect on our averages, therefore we discarded any time measurements that were more than one second. Used the single time cutoff, as described above, we found that with just one test the false positive and negative probabilities were as high as 0.32, however with 50 tests these probabilities fell to 0.21. With 100 observations, taking less than a minute, the error probabilities were as low as 0.1, suggesting that this attack is feasible against a moving target. The reader we used was the cheapest hardware we could find; we expect that more advanced readers with specialised hardware may be able to perform these attacks far more quickly and to a higher degree of accuracy.

5 Conclusion

Our work shows the inherent dangers of using RFID tags in personal items. The e-passport specification was developed by experts over many years and since its publication has been the subject of dozens of academic studies. During this time e-passports have been issued to over 30 million people, all of whom may be at risk of being traced using our attack. As future work we would like to examine more passports and test our attack against other RFID enabled identity documents.

The fix for our attack is relatively simple. First, all e-passports must standardise their error messages. The required error messages in all possible situations should be specified by the ICAO (in e.g. [ICA08]). Second, in the BAC protocol, after a MAC check fails, the passport should try to decrypt the message and check the nonce anyway before sending the error message. Care must also be

taken when implementing new protocols, as our attack might work against any protocol that requires an RFID tag to first check a MAC before decrypting and processing some data.

Our attack is only feasible because the e-passports contain an RFID tag. If e-passports used, for instance, a contact based smart card, then such attacks would not be possible. The reasons for making the e-passports wireless is not immediately clear, the ICAO documentation [ICA06] mentions that reasons for choosing RFID include high data transfer rates, reduced wear and tear on the document and that contact based readers do not fit the shape of the passport. However, contact-based smart cards are quite capable of transferring the data on the card in a reasonable amount of time, and the BAC protocol requires the contact based reading of the passport number and date of birth and expiry, so these reasons seem weak.

Worryingly, the protocols that are used in e-passports are also to be used in some national identity cards, such as the proposed UK ID card scheme [Bog09]. While we have not been able to confirm if these cards will be RFID or contact based, it is possible that our attack will also work against these. It is quite possible that, at some point in the future, it will become a legal requirement for people to carry such an RFID enabled cards and their use will become common to, for instance, access health care, prove identity at an airport or a bank, prove age at a bar, etc. The use of our attack in such a possible future would make it possible for anyone to trace the movements of anyone else.

Acknowledgments. We would like to thank Henryk Plötz for helpful comments regarding how the Interface Device Handler constructs the ATR message.

References

- [AKQ08] Avoine, G., Kalach, K., Quisquater, J.-J.: ePassport: Securing International Contacts with Contactless Chips. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 141–155. Springer, Heidelberg (2008) [cited p. 22, 25]
- [BB] Boycott benetton, <http://www.boycottbenetton.com/> (retrieved 26/8/2009) [cited p. 20]
- [BG08] BSI-Germany. Advanced security mechanisms for machine readable travel documents. Technical report, Federal Office for Information Security (2008) [cited p. 21]
- [Bog09] Boggan, S.: New id cards are supposed to be unforgeable. Daily Mail (August 2009), <http://www.dailymail.co.uk/news/article-1204641> [cited p. 33]
- [Cal] Caldwell, C.: A pass on privacy? The New York Times (July 17, 2005) [cited p. 20]
- [CLRPS06] Carluccio, D., Lemke-Rust, K., Paar, C., Sadeghi, A.-R.: E-passport: The Global Traceability or How to Feel Like an UPS Package. In: Workshop on RFID Security – RFIDSec (2006) [cited p. 22, 25, 26]
- [DHBv09] Danev, B., Heydt-Benjamin, T.S., Čapkun, S.: Physical-layer Identification of RFID Devices. In: Proceedings of the 18th USENIX Security Symposium – USENIX 2009 (2009) [cited p. 26]

- [Eva05] Evangel, A.: Biometric passport: the whole world under control (2005) (in Russian),
<http://www.pcweek.ru/themes/detail.php?ID=69892> [cited p. 31]
- [FCC] Title 47–telecommunication, chapter i–federal communications commission, part 15–radio frequency devices [cited p. 21]
- [Han06] Hancke, G.P.: Practical attacks on proximity identification systems. In: Symposium on Security and Privacy, pp. 328–333 (2006) [cited p. 21]
- [HHJ⁺06] Hoepman, J.-H., Hubbers, E., Jacobs, B., Oostdijk, M., Schreur, R.W.: Crossing Borders: Security and Privacy Issues of the European e-Passport. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 152–167. Springer, Heidelberg (2006) [cited p. 26]
- [ICA06] ICAO. Machine Readable Travel Documents. Doc 9303. Part 1. Technical report, International Civil Aviation Organization (2006) [cited p. 20, 21, 23, 33]
- [ICA08] ICAO. Supplement to doc 9303. Technical report, International Civil Aviation Organization (2008) [cited p. 23, 32]
- [ISO95] Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange, ISO/IEC 7816-4 (1995) [cited p. 23]
- [ISO96] Information technology – Security techniques – Key management – Part 2: Mechanisms using symmetric techniques, ISO/IEC 11770-2 (1996) [cited p. 22]
- [ISO01] Identification cards – Contactless integrated circuit cards – Proximity cards, ISO/IEC 14443 (2001) [cited p. 23]
- [JMW05] Juels, A., Molnar, D., Wagner, D.: Security and Privacy Issues in E-passports. In: SecureComm (2005) [cited p. 21, 22, 25]
- [KW05] Kfir, Z., Wool, A.: Picking Virtual Pockets Using Relay Attacks on Contactless Smartcard Systems. In: SecureComm. IEEE, Los Alamitos (2005) [cited p. 21]
- [Lau06] Laurie, A.: RFIDIOt (2006), <http://rfidiot.org/> [cited p. 26]
- [LKL RP07] Liu, Y., Kasper, T., Lemke-Rust, K., Paar, C.: E-Passport: Cracking Basic Access Control Keys. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part II. LNCS, vol. 4804, pp. 1531–1547. Springer, Heidelberg (2007) [cited p. 26]
- [Min03] Minkin, V.: Myths and realities of biometric passport system (2003) (in Russian), <http://www.elsys.ru/review7.php> [cited p. 31]
- [RMP08] Richter, H., Mostowski, W., Poll, E.: Fingerprinting Passports. In: NLUUG Spring Conference on Security (2008) [cited p. 22, 27]
- [SC89] Snedecor, G.W., Cochran, W.G.: Statistical Methods, 8th edn. Iowa State University Press, Iowa (1989) [cited p. 30]
- [Wor07] The PC/SC Workgroup. Interoperability Specification for ICCs and Personal Computer Systems. Part 3 (2007) [cited p. 27]
- [Yos04] Yoshida, J.: Tests reveal e-passport security flaw. Electronic Engineering Times 1336, 1 (2004) [cited p. 21]

Secure Computation with Fixed-Point Numbers

Octavian Catrina and Amitabh Saxena

Dept. of Computer Science, University of Mannheim, Germany
`{catrina,saxena}@uni-mannheim.de`

Abstract. Secure computation is a promising approach to business problems in which several parties want to run a joint application and cannot reveal their inputs. Secure computation preserves the privacy of input data using cryptographic protocols, allowing the parties to obtain the benefits of data sharing and at the same time avoid the associated risks. These business applications need protocols that support all the primitive data types and allow secure protocol composition and efficient application development. Secure computation with rational numbers has been a challenging problem. We present in this paper a family of protocols for multiparty computation with rational numbers using fixed-point representation. This approach offers more efficient solutions for secure computation than other usual representations.

Keywords: Secure multiparty computation, secure fixed-point arithmetic, secret sharing.

1 Introduction

Secure computation provides cryptographic protocols that enable a group of parties to run joint applications and preserve the privacy of their inputs. For instance, parties P_1, P_2, \dots, P_n can use these protocols to evaluate a function $f(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_n)$, where P_i has private input x_i and output y_i . Roughly speaking, the protocols ensure that the output is correct and the computation does not reveal anything else besides the agreed upon output.

Secure computation can solve business problems where input data belongs to different parties and cannot be revealed or shared with other parties. For example, information sharing and collaborative decision making can substantially improve supply chain performance. However, the supply chain partners are not willing to share the necessary sensitive data (e.g., production costs and capacity), since the risks associated with revealing it exceed the benefits gained. Secure computation can offer the benefits of data sharing and at the same time avoid the risks of disclosing private data. Solutions based on secure computation have been studied for various business problems, including privacy-preserving supply chain planning [2], different types of auctions [9,4], benchmarking [3], and collaborative linear programming [20].

A basic requirement of these applications is a protocol family that provides operations with all primitive data types and allows secure protocol composition

and efficient application development. The protocols proposed so far offer subsets of operations with boolean and integer data or/and specialized solutions for particular problems. Our goal is to provide practical protocols for secure computation with rational numbers.

Our contribution. We present a family of protocols for multiparty computation with rational numbers using fixed-point representation. The protocols are constructed using secure computation based on secret sharing and provide perfect or statistical privacy in the semi-honest model. The protocol family offers arithmetic and comparison with signed fixed-point numbers and evaluation of boolean functions. Secure addition, subtraction, and comparison of fixed-point numbers are trivial extensions of the integer operations. We present new protocols for scaling, multiplication, and division of fixed-point numbers. We also discuss the methods used to optimize the building blocks of these protocols, including a more efficient solution for bit decomposition.

Related Work. We use standard techniques for constructing multiparty computation protocols based on secret sharing, similar to [7,8,19,6]. However, the solutions presented in [8,19] aim at providing perfect privacy and constant round complexity, while our goal is to obtain efficient protocols for secure computation with fixed-point numbers of typical size (≤ 128 bits). For many building blocks we obtain important performance gains using a combination of techniques that includes additive hiding with statistical privacy (instead of perfect privacy), protocols with logarithmic round complexity (instead of constant round complexity), optimized data encoding (especially for binary values), and non-interactive generation of shared random values.

Related protocols focus on secure computation with field (or ring) elements, binary values, and integers. Protocols for secure division (the most complex task) were developed for particular applications and offer only partial solutions. The division protocol in [15] was designed for two-party computation of statistics and relies on a particular structure of the inputs. The multiparty reciprocal protocol in [1] is restricted to positive integers with known range, $2^{k-1} \leq x < 2^k$. This approach based on the Newton-Raphson method (and its extension to division in [14]) is closer to ours. However, our goal is a general division protocol for signed fixed-point numbers. We present a protocol constructed with more accurate and efficient components and using an algorithm that can better take advantage of their properties. In particular, the absolute error of our integer truncation protocol (division by 2^m) is $|\delta| \leq 0.5$ with high probability (rounding to the nearest integer) and $|\delta| < 1$ in the worst case. The approximate truncation protocol in [1] has absolute error $|\delta| \leq n + 1$, where n is the number of parties.

Secure computation with rational numbers has been a challenging problem. An interesting method was proposed in [13] for addition and multiplication of rational numbers using Paillier homomorphic encryption. This method works only for a limited number of consecutive operations (without decryption), depending on the size of the operands and the modulus of the encryption scheme (e.g., 15 operations for 1024-bit modulus and 32-bit numerator and denominator). Our

approach based on fixed-point representation does not have such limitations and offers a complete protocol family for arithmetic and comparison.

Protocols for multiplication and reciprocal of fixed-point numbers were first presented in [5], together with two more general building blocks, reduction modulo 2^m and division by 2^m with rounding down. The fixed-point arithmetic solutions in this paper are more efficient and accurate.

2 Preliminaries

2.1 Secure Computation Framework

Basic framework. Consider a group of $n > 2$ parties, P_1, \dots, P_n , that communicate on secure channels. For $1 \leq i \leq n$, party P_i has private input x_i and output y_i , function of all inputs. Multiparty computation based on secret sharing proceeds as follows. The parties use a linear secret sharing scheme to distribute their private inputs to the group, creating a distributed state of the computation where each party has a share of each secret variable. Certain subsets of parties can reconstruct a secret by pooling together their shares (when needed), while any other subset cannot learn anything about it. Moreover, the properties of the secret sharing scheme allow the parties to compute with shared variables. The protocols used for this purpose take on shared inputs and return shared outputs. This provides the basis for secure protocol composition.

Let X and Y be random variables with finite sample spaces V and W and $\Delta(X, Y) = \frac{1}{2} \sum_{v \in V \cup W} |Pr(X = v) - Pr(Y = v)|$ the statistical distance between them. We say that the distributions are perfectly indistinguishable if $\Delta(X, Y) = 0$ and statistically indistinguishable if $\Delta(X, Y)$ is negligible in some security parameter κ . Our protocols offer perfect or statistical privacy, in the sense that the views of protocol execution (consisting of all values learned by an adversary) can be simulated such that the distributions of real and simulated views are perfectly or statistically indistinguishable, respectively.

We assume a basic framework that uses Shamir secret sharing over a finite field \mathbb{F} . This framework allows secure arithmetic in \mathbb{F} with perfect privacy against a passive threshold adversary able to corrupt t out of n parties. Essentially, in this model, the parties do not deviate from the specified protocol and any $t + 1$ parties can reconstruct a secret, while t or less parties cannot distinguish it from random uniform values in \mathbb{F} . We assume $|\mathbb{F}| > n$, to enable Shamir sharing, and $n > 2t$, for multiplication of secret-shared values. We refer the reader to [7] for a more formal and general presentation of this approach to secure computation.

Complexity metrics. In this framework, the running time of the protocols is (usually) dominated by the communication between parties. We evaluate protocol complexity using two metrics that reflect different aspects of the interaction between parties. Communication complexity measures the amount of data sent by each party. For our protocols, a suitable abstract metric of communication complexity is the number of invocations of a primitive during which every party sends a share (field element) to the others, e.g., the multiplication protocol.

Table 1. Secure arithmetic in a finite field \mathbb{F}

Operation	Purpose	Rounds	Invocations
$[c]^{\mathbb{F}} \leftarrow [a]^{\mathbb{F}} + [b]^{\mathbb{F}}$	Add secrets	0	0
$[c]^{\mathbb{F}} \leftarrow [a]^{\mathbb{F}} + b$	Add secret and public	0	0
$[c]^{\mathbb{F}} \leftarrow [a]^{\mathbb{F}} b$	Multiply secret and public	0	0
$[c]^{\mathbb{F}} \leftarrow [a]^{\mathbb{F}} [b]^{\mathbb{F}}$	Multiply secrets	1	1
$a \leftarrow \text{Output}([a]^{\mathbb{F}})$	Reveal a secret	1	1

Round complexity measures the number of sequential invocations. This metric is relevant for the inherent network delay, independent of the amount of data sent. Invocations that can be executed in parallel count as a single round.

We denote $[x]$ a Shamir sharing of x and $[x]^{\mathbb{F}}$ a sharing in a particular field \mathbb{F} . Table 1 summarizes the secure arithmetic operations in the basic framework.

2.2 Data Representation

The next step toward secure computation using this approach is to map the application data to field elements. The reverse mapping is performed to extract the application data after the computation. We consider the following data types: boolean values, signed integers, and signed fixed-point numbers.

Fixed-point representation. Fixed-point numbers are rational numbers represented as a sequence of digits split into integer and fractional parts by a virtual radix point. For binary digits, a fixed-point number can be written $\tilde{x} = s \cdot (d_{e-2} \dots d_0.d_1 \dots d_f)$ and its value is $\tilde{x} = s \cdot \sum_{i=-f}^{e-2} d_i 2^i$, where $s \in \{-1, 1\}$, e is the length of the integer part (including the sign bit), and f is the length of the fractional part. Denote $\bar{x} = s \cdot \sum_{i=0}^{e+f-2} d_i 2^i$ and observe that $\tilde{x} = \bar{x} \cdot 2^{-f}$, hence \tilde{x} is encoded as an integer \bar{x} scaled by the factor 2^{-f} .

We define a fixed-point data type as follows. Let k , e , and f be integers such that $k > 0$, $f \geq 0$, and $e = k - f \geq 0$. Denote $\mathbb{Z}_{(k)} = \{x \in \mathbb{Z} \mid -2^{k-1} + 1 \leq x \leq 2^{k-1} - 1\}$. The fixed-point data type with resolution 2^{-f} and range 2^e is the set $\mathbb{Q}_{(k,f)} = \{\tilde{x} \in \mathbb{Q} \mid \tilde{x} = \bar{x} \cdot 2^{-f}, \bar{x} \in \mathbb{Z}_{(k)}\}$. Intuitively, $\mathbb{Q}_{(k,f)}$ is obtained by sampling the range of real values $[-2^{e-1} + 2^{-f}, 2^{e-1} - 2^{-f}]$ at 2^{-f} intervals.

We use the following compact notation for a range of integers: $[A..B] = \{x \in \mathbb{Z} \mid A \leq x \leq B\}$ and $[A..B) = \{x \in \mathbb{Z} \mid A \leq x < B\}$.

Data encoding in a field. Any secret value in a secure computation has a data type which is public information. Data types are encoded in a field \mathbb{F} as follows.

Denote $0_{\mathbb{F}}$ and $1_{\mathbb{F}}$ the additive and multiplicative identities of \mathbb{F} . Logical values *false*, *true* and bit values 0, 1 are encoded as $0_{\mathbb{F}}$ and $1_{\mathbb{F}}$, respectively. \mathbb{F} can be a small binary field \mathbb{F}_{2^m} or prime field \mathbb{Z}_q . This encoding allows secure evaluation of boolean functions using secure arithmetic in \mathbb{F} . Encoding in \mathbb{F}_{2^m} is more efficient, because XOR is a local operation; we can take $m = 8$, which is sufficient for Shamir sharing with $n < 256$ parties.

Table 2. Complexity of the building blocks (the default field is \mathbb{Z}_q)

Protocol	Rounds	Invocations	Field
$[r] \leftarrow \text{PRandBit}()$	1	1	\mathbb{Z}_q
$[r] \leftarrow \text{PRandBitL}()$	2	2	\mathbb{Z}_{q_1}
$[r]^{\mathbb{F}_{2^8}}, [r] \leftarrow \text{PRandBitD}()$	2	2	\mathbb{Z}_{q_1}
$[r] \leftarrow \text{PRandInt}(k)$	0	0	0
$[a] \leftarrow \text{BitF2MtoZQ}([a]^{\mathbb{F}_{2^8}})$	2	2	\mathbb{Z}_{q_1}
	1	1	\mathbb{F}_{2^8}
$[c_{k-1}]^{\mathbb{F}}, \dots, [c_0]^{\mathbb{F}} \leftarrow \text{PreOR}([a_{k-1}]^{\mathbb{F}}, \dots, [a_0]^{\mathbb{F}})$	$\log(k)$	$\frac{k}{2} \log(k)$	\mathbb{F}
$[c_{m-1}]^{\mathbb{F}}, \dots, [c_0]^{\mathbb{F}} \leftarrow \text{BitAdd}(a, [b_{m-1}]^{\mathbb{F}}, \dots, [b_0]^{\mathbb{F}})$	$\log(m)$	$m \log(m)$	\mathbb{F}
$[a_{m-1}]^{\mathbb{F}_{2^8}}, \dots, [a_0]^{\mathbb{F}_{2^8}} \leftarrow \text{BitDec}([a], k, m)$	1 2 $\log(m)$	1 $2m$ $m \log(m)$	\mathbb{Z}_q \mathbb{Z}_{q_1} \mathbb{F}_{2^8}
$[s] \leftarrow \text{LTZ}([a], k)$	1 2 $\log(k) + 1$	1 $2k$ $2k - 3$	\mathbb{Z}_q \mathbb{Z}_{q_1} \mathbb{F}_{2^8}

Signed integers are encoded in \mathbb{Z}_q using the function $\text{fld} : \mathbb{Z}_{\langle k \rangle} \mapsto \mathbb{Z}_q$, $\text{fld}(\bar{x}) = \bar{x} \bmod q$, for $q > 2^k$. For any integers $\bar{a}, \bar{b} \in \mathbb{Z}_{\langle k \rangle}$ and operation $\odot \in \{+, -, \cdot\}$ we have $\bar{a} \odot \bar{b} = \text{fld}^{-1}(\text{fld}(\bar{a}) \odot \text{fld}(\bar{b}))$. Moreover, if $\bar{b} \mid \bar{a}$ then $\bar{a}/\bar{b} = \text{fld}^{-1}(\text{fld}(\bar{a}) \cdot \text{fld}(\bar{b})^{-1})$. Secure arithmetic with signed integers can thus be computed using secure arithmetic in \mathbb{Z}_q .

A secret fixed-point number \tilde{x} of type $\mathbb{Q}_{\langle k, f \rangle}$ is represented as a secret integer $\bar{x} = \tilde{x}2^f$ encoded in \mathbb{Z}_q and public parameters that specify the resolution and the range, f and e (or $k = e + f$). We define the map $\text{int}_f : \mathbb{Q}_{\langle k, f \rangle} \mapsto \mathbb{Z}_{\langle k \rangle}$, $\text{int}_f(\tilde{x}) = \tilde{x}2^f$. Note that the fixed-point representation allows very efficient encoding of a secret rational number, as a single field element.

We also use (when required) a bitwise encoding of integers, where each bit of the binary representation is encoded and shared in a field \mathbb{F} as described above.

We distinguish different representations of a number using the following simplified notation: we denote \tilde{x} a rational number of some fixed-point type $\mathbb{Q}_{\langle k, f \rangle}$ and $\bar{x} = \tilde{x}2^f \in \mathbb{Z}_{\langle k \rangle}$ the integer value of its fixed-point representation; for secure computation using secret-sharing we denote $x = \bar{x} \bmod q \in \mathbb{Z}_q$ the field element that encodes \bar{x} (and hence \tilde{x}) and $[x]$ a sharing of x .

2.3 Building Blocks

We provide an overview of several building blocks and techniques used in fixed-point arithmetic protocols. Their complexity is summarized in Table 2.

Shared random values. The protocols often use secret sharing together with additive or multiplicative hiding, taking advantage of their combined capabilities for computing with secret data and efficient conversion methods. For example, given a shared variable $[x]$ the parties can jointly generate a shared random value

$[r]$, compute $[y] = [x] + [r]$, and reveal $y = x + r$. This is similar to one-time pad encryption of x with key r .

For a secret $x \in \mathbb{Z}_q$ and random uniform $r \in \mathbb{Z}_q$ we obtain $\Delta(x+r \bmod q, r) = 0$, hence perfect privacy. Alternatively, for $x \in [0..2^k)$, random uniform $r \in [0..2^{k+\kappa})$, and $q > 2^{k+\kappa+1}$ we obtain $\Delta(x+r \bmod q, r) < 2^{-\kappa}$, hence statistical privacy with security parameter κ . The variant with statistical privacy can substantially simplify the protocols by avoiding wraparound modulo q , although it requires larger q (hence larger shares) for a given data range. Statistical privacy also holds for other distributions of r that can be generated more efficiently or/and meet particular requirements: (1) $r = \sum_i r_i$, where $r_i \in [0..2^{k+\kappa})$ are random uniform integers; (2) $r = 2^k r'' + r'$, where $r'' = \sum_i r''_i$ and $r''_i \in [0..2^\kappa)$ and $r' \in [0..2^k)$ are random uniform integers.

We use Pseudo-random Replicated Secret Sharing (PRSS) [6] to generate without interaction shared random values in \mathbb{F} with uniform distribution and random sharings of zero. Also, we use the integer variant of PRSS (RISS) [10] to generate shared random integers in a given interval, and the ideas in [11] for bit-share conversions (e.g., BitF2MtoZQ converts bit shares from \mathbb{F}_{2^8} to \mathbb{Z}_q).

To enable these techniques, we assume in the remainder of the paper that numbers are encoded in \mathbb{Z}_q as specified in Section 2.2 and $q > 2^{k+\kappa+\nu+1}$, where k is the required integer bit-length, κ is the security parameter, $\nu = \lceil \log(\binom{n}{t}) \rceil$, n is the number of parties, and t is the corruption threshold.

Protocol PRandBit generates a random bit shared in \mathbb{Z}_q by combining the protocol RandBit in [8] and protocols in [6]. A random uniform integer $r \in [0..2^k)$ is constructed from shared random bits as $[r] = \sum_{i=0}^{k-1} 2^i [r_i]$. Note that RandBit includes an exponentiation that significantly increases the running time when generating many random bits for large q . PRandBitL generates a shared random bit in a small field \mathbb{Z}_{q_1} to reduce complexity, then converts its shares to the target field \mathbb{Z}_q (e.g., $\lceil \log(q_1) \rceil = 64$). PRandBitD uses a similar technique to generate a random bit shared in both \mathbb{Z}_q and \mathbb{F}_{2^8} . Bits shared in \mathbb{Z}_q are used to construct a random uniform integer, while bits shared in \mathbb{F}_{2^8} are used for binary computation. PRandInt(k) generates without interaction a shared random integer $r \in [0..2^{k+\nu})$ distributed as sum of $\binom{n}{t}$ random uniform integers in $[0..2^k)$.

Bit decomposition. Protocol 2.1, BitDec, is a general tool that provides a bridge between secure computation with integers shared in \mathbb{Z}_q and with integers bitwise-shared in \mathbb{Z}_q or \mathbb{F}_{2^8} . The inputs are $[a] = [\text{fld}(\bar{a})]$ and the public integers k and m , where $\bar{a} \in \mathbb{Z}_{(k)}$ and $0 < m \leq k$. The output is an array of shared bits equal to the m least significant bits of the 2's complement binary representation of \bar{a} . The protocol follows the idea in [8,18,19] for bit decomposition of \mathbb{Z}_q elements, but offers a more efficient solution for bounded integers and statistical privacy. Protocol 2.1 extracts m bits in $\log(m) + 3$ rounds with $m \log(m) + 2m + 1$ invocations, while the variant with perfect privacy and constant round complexity [19] extracts $k = \lceil \log(q) \rceil$ bits in 51 rounds with $56k \log(k) + 30k$ invocations.

Correctness. Let $\ell = k + \kappa + \nu$. The protocol generates a random integer $0 \leq r < 2^\ell$ and computes $c = (2^\ell + 2^k + a - r) \bmod q$. For $q > 2^{\ell+1}$ we have $(2^k + a) \bmod q = 2^k + \bar{a}$ and $c = 2^\ell + 2^k + \bar{a} - r$. If $\bar{a} \geq 0$ then $(r+c) \bmod 2^k = \bar{a}$

and if $\bar{a} < 0$ then $(r + c) \bmod 2^k = 2^k - |\bar{a}|$, hence $(r + c) \bmod 2^k$ is equal to the 2's complement representation of \bar{a} . The protocol computes the $m \leq k$ least significant bits of \bar{a} using the binary addition protocol **BitAdd**.

Protocol 2.1. $([a_{m-1}]^{\mathbb{F}_{2^8}}, \dots, [a_0]^{\mathbb{F}_{2^8}}) \leftarrow \text{BitDec}([a], k, m)$

```

1 foreach  $i \in [0..m-1]$  do parallel
2    $[r_i]^{\mathbb{F}_{2^8}}, [r_i] \leftarrow \text{PRandBitD}();$ 
3    $[r'] \leftarrow \sum_{i=0}^{m-1} 2^i \cdot [r_i];$ 
4    $[r''] \leftarrow \text{PRandInt}(\kappa + k - m);$ 
5    $[r] \leftarrow 2^m \cdot [r''] + [r'];$ 
6    $c \leftarrow \text{Output}(2^{k+\kappa+\nu} + 2^k + [a] - [r]);$ 
7    $([a_{m-1}]^{\mathbb{F}_{2^8}}, \dots, [a_0]^{\mathbb{F}_{2^8}}) \leftarrow \text{BitAdd}((c_{m-1}, \dots, c_0), ([r_{m-1}]^{\mathbb{F}_{2^8}}, \dots, [r_0]^{\mathbb{F}_{2^8}}));$ 
8 return  $([a_{m-1}]^{\mathbb{F}_{2^8}}, \dots, [a_0]^{\mathbb{F}_{2^8}});$ 

```

Security. Protocol **BitDec** can leak information in step 6 when it reveals c . The other building blocks provide perfect privacy or statistical privacy with security parameter κ . Since $\Delta(c, r) < 2^{-\kappa}$ we conclude that **BitDec** provides statistical privacy with security parameter κ .

Complexity. Table 3 shows the complexity of **BitDec** and its building blocks. Observe that most of the invocations are in small fields, \mathbb{Z}_{q_1} or \mathbb{F}_{2^8} . The double-shared random bits can be generated in parallel in 2 rounds and precomputed. The random integer r is constructed such that to minimize the number of shared random bits. **BitAdd** uses standard algorithms and is designed to offer a good trade-off between round and communication complexity (a variant with minimal communication needs $2 \log(m)$ rounds).

3 Secure Fixed-Point Arithmetic

The protocols for arithmetic with fixed-point numbers are constructed using secure integer arithmetic and scaling. Let \tilde{a}, \tilde{b} be fixed-point numbers. We denote $\tilde{a} + \tilde{b}, \tilde{a} - \tilde{b}, \tilde{a} \cdot \tilde{b}, \tilde{a}/\tilde{b}$ the exact arithmetic operations (the result is a real number). The output of a protocol may differ from the exact result, either because the value is truncated to obtain a given fixed-point representation, or because the algorithm computes an approximation of the result.

We present a secure arithmetic operation in three steps: we first give an algorithm for exact arithmetic; then, we derive an algorithm for inputs and output of given fixed-point types and limited precision arithmetic, and evaluate its error; finally, we use this algorithm to obtain a protocol with secret inputs and output. The second algorithm takes as inputs $\bar{a} = \tilde{a}2^f, \bar{b} = \tilde{b}2^f$ and computes the result $\bar{c} = \tilde{c}2^f$ using integer arithmetic. For secure computation, fixed-point numbers are encoded in \mathbb{Z}_q and secret-shared. Let $a = \bar{a} \bmod q, b = \bar{b} \bmod q, c = \bar{c} \bmod q$ the encoded numbers. On input the secret-shared values $[a]$ and $[b]$ the protocol computes the secret-shared output $[c]$ using secure arithmetic in \mathbb{Z}_q . Table 3 summarizes the complexity of the protocols presented in this section.

Table 3. Complexity of the fixed-point arithmetic protocols

Protocol	Rounds	Invocations	Field
$[d] \leftarrow \text{TruncPr}([a], k, m)$	1	1	\mathbb{Z}_q
	2	$2m$	\mathbb{Z}_{q_1}
TruncPr after precomputation	1	1	\mathbb{Z}_q
$[c] \leftarrow \text{FPMul}([a], [b], k, f)$	2	2	\mathbb{Z}_q
	2	$2f$	\mathbb{Z}_{q_1}
FPMul after precomputation	2	2	\mathbb{Z}_q
$[y] \leftarrow \text{FPDiv}([a], [b], k, f)$ $(e = f, k = 2f, \theta \text{ iterations})$	$2\theta + 8$	$4\theta + 8$	\mathbb{Z}_q
	2	$2k\theta + 6.5k$	\mathbb{Z}_{q_1}
	$3\log(k) + 2$	$1.5k\log(k) + 4k - 2$	\mathbb{F}_{2^8}
FPDiv after precomputation	$2\theta + 8$	$4\theta + 8$	\mathbb{Z}_q
$[w] \leftarrow \text{AppRcr}([b], k, f)$ after precomputation	$3\log(k) + 2$	$1.5k\log(k) + 4k - 2$	\mathbb{F}_{2^8}
	7	7	\mathbb{Z}_q

3.1 Scaling

The purpose of scaling is to convert a given number to a fixed-point type with different resolution. Let $\tilde{a}_1 = \bar{a}_1 2^{-f_1}$ and suppose we want to convert this value to $\tilde{a}_2 = \bar{a}_2 2^{-f_2}$. Let $m = f_2 - f_1$. We distinguish two cases. If $m \geq 0$ we have to scale up \tilde{a}_1 by computing $\bar{a}_2 = \bar{a}_1 2^m$. We obtain $\tilde{a}_2 = \tilde{a}_1$ (same value with higher resolution). If $m < 0$ we have to scale down (truncate) \tilde{a}_1 . Let $\text{trunc}(\bar{x}, m) = \bar{x}/2^m - \delta_t$, where δ_t is the absolute error of the truncation operation. We scale down \tilde{a}_1 by computing $\bar{a}_2 = \text{trunc}(\bar{a}_1, m)$ and obtain $\tilde{a}_2 \approx \tilde{a}_1$ with absolute error $\delta = \tilde{a}_1 - \tilde{a}_2 = \delta_t 2^{-f_2}$. For example, if $\text{trunc}(\bar{x}, m)$ rounds down (discards m bits) then $0 \leq \delta_t < 1$. If it rounds to the nearest integer then $-0.5 < \delta_t \leq 0.5$.

A secret number $[a_1]$ is scaled up without interaction by computing $[a_2] = [a_1]2^m$. Truncation is more complicated. We present an accurate and efficient solution. Let $\bar{a} \in \mathbb{Z}_{(k)}$ and $0 < m < k$. Protocol 3.1, TruncPr , takes as inputs $[a]$ and the public integers k and m and returns $[d]$ such that $\bar{d} = \lfloor \bar{a}/2^m \rfloor + u$, where u is a random bit and $\Pr(u = 1) = (\bar{a} \bmod 2^m)/2^m$. Therefore, the protocol rounds $\bar{a}/2^m$ to the nearest integer with probability $1 - \alpha$, where α is the distance between $\bar{a}/2^m$ and the nearest integer.

Correctness. A signed integer \bar{a} is encoded in \mathbb{Z}_q as $a = \text{fld}(\bar{a}) = \bar{a} \bmod q$. Step 1 maps $\bar{a} \in [-2^{k-1}..2^{k-1}]$ to $b \in [0..2^k)$ by computing $b = (2^{k-1} + a) \bmod q = 2^{k-1} + \bar{a}$. Observe that $b' = b \bmod 2^m = \bar{a} \bmod 2^m$ for any $0 < m \leq k$. Denote $\ell = k + \kappa + \nu$. Steps 2-6 generate a random secret $r \in [0..2^\ell)$ and reveal $c = (b + r) \bmod q$. For $q > 2^{\ell+1}$ we have $q > b + r$ and hence $c = b + r$. Let $c' = c \bmod 2^m$ and $r' = r \bmod 2^m$. We see that $c' = (b' + r') \bmod 2^m = b' + r' - u \cdot 2^m$, where $u \in \{0, 1\}$. Therefore, steps 1-9 compute $a' = (\bar{a} \bmod 2^m) - u \cdot 2^m$.

Let $d' = (a - a') \bmod q$ and observe that $d' = (\bar{a} - (\bar{a} \bmod 2^m) + u \cdot 2^m) \bmod q = ([\bar{a}/2^m] \cdot 2^m + u \cdot 2^m) \bmod q$. The protocol returns $d = d'(2^{-m} \bmod q) \bmod q$. We have $d = ([\bar{a}/2^m] + u) \bmod q = \text{fld}([\bar{a}/2^m] + u)$, hence the output is correct.

Protocol 3.1. $[d] \leftarrow \text{TruncPr}([a], k, m)$

```

1   $[b] \leftarrow 2^{k-1} + [a];$ 
2  foreach  $i \in [0..m-1]$  do parallel
3     $[r_i] \leftarrow \text{PRandBitL}();$ 
4   $[r'] \leftarrow \sum_{i=0}^{m-1} 2^i \cdot [r_i];$ 
5   $[r''] \leftarrow \text{PRandInt}(q, \kappa + k - m);$ 
6   $[r] \leftarrow 2^m \cdot [r''] + [r'];$ 
7   $c \leftarrow \text{Output}([b] + [r]);$ 
8   $c' \leftarrow c \bmod 2^m;$ 
9   $[a'] \leftarrow c' - [r'];$ 
10  $[d] \leftarrow ([a] - [a'])2^{-m};$ 
11 return  $[d];$ 

```

Probabilistic rounding. Observe that $b' + r' \in [0..2^{m+1} - 2]$ and that $u = 1$ if $b' + r' \geq 2^m$ and $u = 0$ if $b' + r' < 2^m - 1$. It follows that $\Pr(u = 1) = \Pr(r' \geq 2^m - b')$. We see that $p(b') = \Pr(u = 1)$ grows with b' from $p(0) = 0$ to $p(2^m - 1) \approx 1$. For example, if r' is random uniform in $[0..2^m - 1]$, we obtain $p(b') = b'/2^m$, hence $p(0) = 0$, $p(2^m/2) = 1/2$, and $p(2^m - 1) = 1 - 2^{-m}$. Note that deterministic rounding is too expensive because it requires comparisons.

Security. Protocol TruncPr can leak information in step 7 when it reveals $c = b+r$. The other building blocks provide perfect privacy or statistical privacy with security parameter κ . Since $\Delta(c, r) < 2^{-\kappa}$ we conclude that TruncPr provides statistical privacy with security parameter κ .

Complexity. All random bits are generated in parallel in 1 or 2 rounds depending on the protocol used, PRandBit or PRandBitL , and can be precomputed. The construction of r minimizes the number of shared random bits generated by the protocol. Table 3 shows the complexity of the variant using PRandBitL .

Extensions. Observe that $b' = c' - r' + u \cdot 2^m$ and that $u = 1$ if $c' < r'$ and $u = 0$ if $c' \geq r'$. We can compute $[u]$ using a comparison protocol for bitwise-shared integers and obtain a protocol $\text{Trunc}([a], k, m)$ that computes $\bar{d} = \lfloor \bar{a}/2^m \rfloor$, i.e., truncates m bits and rounds down. This is the truncation protocol used in [5]. Note that TruncPr is substantially more efficient than Trunc , since it avoids an expensive bitwise comparison, and at the same time reduces the rounding error with high probability to $|\delta_t| < 0.5$.

Furthermore, if $\bar{a} < 0$ then $\lfloor \bar{a}/2^{k-1} \rfloor = -1$ and if $\bar{a} \geq 0$ then $\lfloor \bar{a}/2^{k-1} \rfloor = 0$. Therefore, we can determine the sign of a secret integer by computing $[s] = -\text{Trunc}([a], k, k-1)$. This is the comparison protocol $\text{LTZ}([a], k)$ in Table 2.

3.2 Addition, Subtraction, and Comparison

We specify addition and subtraction for values of the same fixed-point type. Values of different types have to be converted to the same type. Let $\tilde{a}, \tilde{b} \in \mathbb{Q}_{\langle k, f \rangle}$ and $\tilde{c} = \tilde{a} + \tilde{b}$. Since $\tilde{c} = (\bar{a} + \bar{b})2^{-f}$, we obtain the representation of \tilde{c} with

resolution 2^{-f} by computing $\bar{c} = \bar{a} + \bar{b}$. For secret-shared values we compute $[c] = [a] + [b]$ and $[c] = a + [b]$ without interaction. The output is the exact result and has the same type as the inputs. Subtraction is similar.

Integer comparison operators with secret inputs and output can be constructed using the following two protocols: $\text{EQZ}([a])$, that computes $\bar{a} \stackrel{?}{=} 0$, and $\text{LTZ}([a])$, that computes $\bar{a} \stackrel{?}{<} 0$. For example, $\text{EQ}([a], [b]) = \text{EQZ}([a] - [b])$ computes $\bar{a} \stackrel{?}{=} \bar{b}$ and $\text{GE}([a], [b]) = 1 - \text{LTZ}([a] - [b])$ computes $\bar{a} \stackrel{?}{\geq} \bar{b}$. We can also use these protocols for fixed-point inputs of the same type and obtain exact results.

3.3 Multiplication

We first consider multiplication of two numbers of the same fixed-point type, $\tilde{a}, \tilde{b} \in \mathbb{Q}_{(k,f)}$. Let $\tilde{c} = \tilde{a}\tilde{b}$. Since $\tilde{c} = \bar{a}\bar{b}2^{-2f}$, we obtain the representation of the exact result \tilde{c} with resolution 2^{-2f} by computing $\bar{c} = \bar{a}\bar{b}$ (if overflow does not occur). For secret-shared values we compute $[c] = [a][b]$ with complexity 1 round and 1 invocation. If an input is public then $[c] = a[b]$, without interaction.

The output of a multiplication is usually scaled down to resolution 2^{-f} in order to obtain a value with the same type as the inputs and to limit the size of the integers that encode the fixed-point numbers. Thus, for a typical multiplication we compute $\bar{d} = \text{trunc}(\bar{a}\bar{b}, f)$ and obtain $\tilde{d} \approx \tilde{a}\tilde{b}$ with absolute error $\delta = \tilde{a}\tilde{b} - \bar{d} = \delta_t 2^{-f}$. The secure computation is shown in Protocol 3.2. Observe that the output overflows when the intermediate value $\bar{a}\bar{b}$ reaches $k + f$ bits. Therefore, \mathbb{Z}_q must support integers of at least $k + f$ bits in order to avoid overflow of $\bar{a}\bar{b}$ for all valid outputs.

Protocol 3.2. $[d] \leftarrow \text{FPMul}([a], [b], k, f)$

- 1 $[c] \leftarrow [a][b];$
 - 2 $[d] \leftarrow \text{TruncPr}([c], 2k, f);$
 - 3 **return** $[d];$
-

Fixed-point multiplication with inputs and outputs of different types can be computed using similar protocols, with the same complexity and accuracy. For example, if the inputs are $\tilde{a} = \bar{a}2^{-f_a}$, $\tilde{b} = \bar{b}2^{-f_b}$ and the output is $\tilde{d} = \bar{d}2^{-f}$, where $f \leq f_a + f_b$, the computation is $\bar{d} = \text{trunc}(\bar{a}\bar{b}, f_a + f_b - f)$. We obtain $\tilde{d} \approx \tilde{a}\tilde{b}$ with absolute error $\delta = \delta_t 2^{-f}$. Truncation is not necessary if one input is an integer and the other one has the same resolution as the output.

We point out two optimizations that improve the efficiency and accuracy of the protocols by reducing the number of truncations. We assume inputs and outputs of the same type $\mathbb{Q}_{(k,f)}$ and $|\delta_t| < 1$.

We can evaluate the inner product $\sum_{i=1}^m \tilde{a}_i \tilde{b}_i$ with error $\delta = \delta_t 2^{-f}$ by computing $\bar{d} = \text{trunc}(\sum_{i=1}^m \bar{a}_i \bar{b}_i, f)$. Computing $\bar{d}' = \sum_{i=1}^m \text{trunc}(\bar{a}_i \bar{b}_i, f)$ is both inefficient and less accurate, since the cumulated errors can reach $|\delta'| < m2^{-f}$. A double multiplication $\bar{a}\bar{b}\bar{c}$ can be evaluated with absolute error $\delta = \delta_t 2^{-f}$ by computing $\bar{d} = \text{trunc}(\bar{a}\bar{b}\bar{c}, 2f)$, if the data representation supports integers of $k + 2f$ bits. On the other hand, if we compute $\bar{d}' = \text{trunc}(\text{trunc}(\bar{a}\bar{b}, f)\bar{c}, f)$, the error becomes $\delta' \approx \delta_t 2^{\ell-2f}$, assuming $\bar{c} \in [2^{\ell-1}..2^\ell]$.

3.4 Division

Secure division with secret dividend and public divisor follows immediately from fixed-point multiplication. Let $\tilde{a}, \tilde{b} \in \mathbb{Q}_{(k,f)}$, and assume \tilde{a} is secret and \tilde{b} is public. We can obtain the secret quotient $\tilde{y} \approx \tilde{a}/\tilde{b} \in \mathbb{Q}_{(k,f)}$ with error $\delta = \delta_t 2^{-f}$ by computing the reciprocal $\tilde{x} \approx 1/\tilde{b}$, $\tilde{x} \in \mathbb{Q}_{(k+f,k)}$, and then $[y] = \text{TruncPr}([a] \cdot \text{fld}(\text{int}_k(\tilde{x})), k)$. Therefore, if the divisor is public the division protocol has the same complexity as the truncation. For example, this protocol may be sufficient for secure evaluation of statistics like sample mean $\mu = \frac{1}{N} \sum_{i=1}^N x_i$ and variance $\sigma^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2$ since N is usually public.

The problem becomes difficult when the divisor is secret. The algorithms for dividing fixed-point numbers follow two main approaches: digit recurrence (subtractive division) and functional iteration (multiplicative division) [12]. Functional iteration is more suitable for secure computation, because the algorithms converge faster and are simpler to implement with the available building blocks. These algorithms fall into two main classes: algorithms that use the Newton-Raphson method for computing the reciprocal and algorithms that use series expansion, in particular Goldschmidt's method. Both methods require a suitable initial approximation, which is the main hurdle for secure computation. Moreover, both offer quadratic convergence and the iterations have similar complexity. The Newton-Raphson iterations are self correcting (truncation errors in an iteration decrease quadratically during next iterations), but the multiplications are dependent and have to be computed sequentially. For Goldschmidt's method, the multiplications can be computed concurrently, but truncation errors cumulate during the iterations. We developed and evaluated protocols for both methods. We present in this paper a protocol based on Goldschmidt's method that offers better efficiency (for similar accuracy).

Goldschmidt's method for computing a/b can be described as follows [16]. Let w_0 be an initial approximation of $1/b$ with relative error $\epsilon_0 < 1$, and let $a_0 = a$, $b_0 = b$. For $i \geq 1$ the algorithm computes: $a_i = a_{i-1}w_{i-1}$, $b_i = b_{i-1}w_{i-1}$, $w_i = 2 - b_i$. Denote $r_i = \prod_{j=0}^i w_j$ and observe that:

$$\frac{a}{b} = \frac{aw_0 \dots w_{i-1}}{bw_0 \dots w_{i-1}} = \frac{a_i}{b_i} = \frac{a_i w_i}{b_i w_i} = \frac{a_{i+1}}{b_{i+1}} = \frac{ar_i}{br_i}.$$

The relative error of the initial approximation is $\epsilon_0 = 1 - bw_0$. It can be shown (by induction) that $b_i = 1 - \epsilon_0^{2^{i-1}}$ and $w_i = 1 + \epsilon_0^{2^{i-1}}$. Observe that if $\epsilon_0 < 1$ then b_i converges to 1 and hence a_i converges to the quotient a/b and r_i to the reciprocal $1/b$. Denoting $e_i = \epsilon_0^{2^i}$, the recurrence relations that approximate the quotient can be written as follows: $a_1 = aw_0$; $a_{i+1} = a_i(1 + e_{i-1})$, $e_i = e_{i-1}^2$. After i iterations we obtain $a_{i+1} = (a/b)(1 - \epsilon_0^{2^i})$, hence $a_{i+1} \approx a/b$ with relative error $\epsilon_0^{2^i}$. We can obtain similar recurrence relations for $1/b$.

Initial approximation. A critical issue is to determine an initial approximation that ensures fast convergence. The usual method is to compute a normalized input $c \in [0.5, 1)$ and then find an approximation of $1/c$. We use the linear

approximation $w_0 = 2.9142 - 2c$ with relative error $\epsilon_0 < 0.08578$ (3.5 initial bits) [12]. This approximation offers sufficient accuracy for our purposes and can be computed without interaction for secret c .

More accurate initial approximations can be obtained by table lookup [17]. For example, a piece-wise linear approximation using a table with 2^k entries offers initial approximations with accuracy of $2k + 2$ bits. A reciprocal with 64-bit accuracy can thus be computed in 2 iterations, with an initial approximation based on a table with only 128 entries. However, the efficiency gain is reduced by the additional cost of the table lookup with secret index.

Division algorithm. The division protocol performs the computation described above using the building blocks in the previous sections. We give an algorithm for positive inputs and then show how to extend it to signed inputs.

Let $\tilde{a}, \tilde{b} \in \mathbb{Q}_{(k,f)}^+$ and assume that $2^{\ell-f-1} \leq \tilde{a} < 2^{\ell-f}$ and $2^{m-f-1} \leq \tilde{b} < 2^{m-f}$, for some $\ell \leq k$ and $m \leq k$. Our goal is to compute $\tilde{y} \in \mathbb{Q}_{(k,f)}^+$ such that $\tilde{y} \approx \tilde{a}/\tilde{b}$ and the maximum absolute error is close to the resolution 2^{-f} of the output. We describe the exact computation (without truncations) followed by the computation with limited precision carried out by the protocol:

1. Computation of the initial approximation $\tilde{w} \approx 1/\tilde{b}$:

Exact arithmetic: Normalize \tilde{b} to obtain $\tilde{c} \in (0.5, 1)$. The normalized divisor is $\tilde{c} = \tilde{b}2^{f-m} = \tilde{b}2^{u-e}$, where $u = k-m = e+f-m$. Let $\tilde{d} = 2.9142 - 2\tilde{c}$ be the initial approximation of $1/\tilde{c}$. The initial approximation of $1/\tilde{b}$ is $\tilde{w} = \tilde{d}2^{u-e}$.

Approximate arithmetic: Assume \tilde{b} with resolution 2^{-f} , \tilde{c} with resolution 2^{-k} and \tilde{w} with resolution 2^{-f} . Let $\bar{b} = \tilde{b}2^f$, $\bar{c} = \tilde{c}2^k$, $\bar{w} = \tilde{w}2^f$. Compute: $\bar{c} = \bar{b}2^u$; $\bar{d} = \text{int}_k(2.9142) - 2\bar{c}$; $\bar{w} = \text{trunc}(\bar{d}2^u, 2e)$.

2. Computation of $\tilde{y} \approx \tilde{a}/\tilde{b}$:

Exact arithmetic: Let $\tilde{y}_0 = \tilde{a}\tilde{w}$ and $\tilde{x}_0 = 1 - \tilde{b}\tilde{w}$ (note that \tilde{x}_0 is the relative error of \tilde{w} and $0 \leq \tilde{x}_0 < 1$). For $1 \leq i < \theta$ do: $\tilde{y}_i = \tilde{y}_{i-1} + \tilde{y}_{i-1}\tilde{x}_{i-1}$; $\tilde{x}_i = \tilde{x}_{i-1}\tilde{x}_{i-1}$. Let $\tilde{y} = \tilde{y}_\theta = \tilde{y}_{\theta-1} + \tilde{y}_{\theta-1}\tilde{x}_{\theta-1}$ (last iteration). We obtain $\tilde{y} \approx \tilde{a}/\tilde{c}$ with relative error $\epsilon_\theta < \epsilon_0^2$.

Approximate arithmetic: Assume $\tilde{a}, \tilde{b}, \tilde{w}, \tilde{y}_i$ with resolution 2^{-f} , and \tilde{x}_i with resolution 2^{-2f} . Denote $\bar{a} = \tilde{a}2^f$, $\bar{b} = \tilde{b}2^f$, $\bar{y}_i = \tilde{y}_i2^f$, $\bar{w} = \tilde{w}2^f$, and $\bar{x}_i = \tilde{x}_i2^{2f}$. Let $\bar{x}_0 = \text{int}_{2f}(1.0) - \bar{b}\bar{w}$ and $\bar{y}_0 = \text{trunc}(\bar{a}\bar{w}, f)$. For $1 \leq i < \theta$ do: $\bar{y}_i = \bar{y}_{i-1} + \text{trunc}(\bar{y}_{i-1}\bar{x}_{i-1}, 2f)$; $\bar{x}_i = \text{trunc}(\bar{x}_{i-1}\bar{x}_{i-1}, 2f)$. Let $\bar{y} = \bar{y}_{\theta-1} + \text{trunc}(\bar{y}_{\theta-1}\bar{x}_{\theta-1}, 2f)$ (last iteration).

Correctness. Since $2^{m-1} \leq \tilde{b}2^f < 2^m$, we have $2^{k-1} \leq \tilde{b}2^f2^{k-m} < 2^k$ and $2^{-1} \leq \tilde{b}2^{f-m} < 1$, so the normalized divisor is $\tilde{c} = \tilde{b}2^{f-m} = \tilde{b}2^{u-e}$.

The initial approximation of $1/\tilde{c}$ is $\tilde{d} = 2.9142 - 2\tilde{c}$. From $\tilde{d} \approx 1/(\tilde{b}2^{u-e})$ it follows that $\tilde{w} = \tilde{d}2^{u-e} \approx 1/\tilde{b}$. The relative error of \tilde{w} is $\tilde{x} = (1/\tilde{b} - \tilde{w})/(1/\tilde{b}) = 1 - \tilde{b}\tilde{w}$. For the fixed-point types in the algorithm we obtain: $\tilde{c}2^k = \tilde{b}2^{-f}2^{u-e}2^k = \tilde{b}2^u$ hence $\bar{c} = \bar{b}2^u$; $\tilde{d}2^k = (\text{int}_k(2.9142)2^{-k} - 2\bar{c}2^{-k})2^k$ hence $\bar{d} = \text{int}_k(2.9142) - 2\bar{c}$; $\tilde{w}2^f = \tilde{d}2^{-k}2^{u-e}2^f = \tilde{d}2^{u-2e}$ hence $\bar{w} = \text{trunc}(\bar{d}2^u, 2e)$; and $\tilde{x}_02^{2f} = (\text{int}_{2f}(1.0)2^{-2f} - \bar{b}2^{-f}\bar{w}2^{-f})2^{2f}$ hence $\bar{x}_0 = \text{int}_{2f}(1.0) - \bar{b}\bar{w}$.

The iterations follow the simple recurrence relations presented earlier. Correctness of the computation with limited precision is easy to verify. Observe that the two fixed-point multiplications in an iteration can be computed in parallel, and in the last iteration it is sufficient to compute \tilde{y}_θ .

Signed inputs. Since $\tilde{x}_i + 1 \geq 0$ the division algorithm works for $\tilde{a} \leq 0$ without modification. The extension to $\tilde{b} < 0$ affects only the initial approximation algorithm, which is modified to return $\tilde{w} \approx 1/\tilde{b}$ with the correct sign. Thus, $\tilde{y}_0 = \tilde{a}\tilde{w}$ is initialized with the correct sign, and the iterations preserve it.

Accuracy. The quotient error has two main components: the approximation error of the method, which depends on the initial approximation and the number of iterations, and the truncation error due to computation of the iterations with limited precision. The accuracy is limited by the resolution 2^{-f} of the output.

For exact computation of the iterations, the relative error after iteration θ is $\epsilon_\theta < \epsilon_0^{2^\theta}$, where ϵ_0 is the relative error of the initial approximation of $1/\tilde{b}$. For example, we use a linear approximation with $\epsilon_0 < 0.08578$, so the approximation error of \tilde{y}_5 is $\epsilon_5 < 7.4 \cdot 10^{-35}$. This implies 113 exact quotient bits, hence an absolute error less than 2^{-f} for $k = 2f \leq 112$ bits.

For θ iterations, the cumulated absolute error δ_T due to truncations is upper bounded by $\theta 2^{-f}$. This error is essentially caused by truncation of \tilde{y}_i , which adds an error $|\delta_{T_y}| < 2^{-f}$ per iteration. Truncation of \tilde{x}_i introduces a negligible error $|\delta_{T_x}| < 2^{-2f} \ll |\delta_{T_y}|$. Assuming sufficient iterations for an approximation error 2^{-f} , the overall error of the algorithm is bound by $(\theta + 1)2^{-f}$. The error bound can be reduced to 2^{-f} by slightly increasing the resolution of \tilde{y}_i .

We note that the average accuracy of the truncations is better than the worst case considered above. The error bound observed in experiments with an implementation of the division protocol is actually close to 2^{-f} .

Protocols. Let $\tilde{a}, \tilde{b} \in \mathbb{Q}_{(k,f)}$ and $\tilde{b} \neq 0$. On input $[a], [b]$ the Protocol 3.3, FPDiv, computes $[y]$ such that $\tilde{y} \in \mathbb{Q}_{(k,f)}$ and $\tilde{y} \approx \tilde{a}/\tilde{b}$, using the algorithm described above. Protocol 3.4, AppRcr, provides the initial approximation of $1/\tilde{b}$. It takes as input the divisor $[b]$ and returns $[w]$ such that $\tilde{w} \in \mathbb{Q}_{(k,f)}$ and $\tilde{w} \approx 1/\tilde{b}$. The linear approximation is computed using the normalized value of the divisor obtained by Protocol 3.5, Norm.

Correctness. The correctness of FPDiv and AppRcr is easy to verify based on the algorithm description. Protocol Norm takes as input $[b]$, for $\tilde{b} \in \mathbb{Q}_{(k,f)}$ and computes the secret integer values $[c]$ and $[v']$ such that $2^{k-1} \leq \bar{c} < 2^k$ and $\bar{c} = \bar{b}\bar{v}'$. Suppose that $2^{m-1} \leq |\bar{b}| < 2^m$, $m \leq k$. Observe that $|\bar{v}'| = 2^{k-m}$ and $|\bar{c}| = (|\bar{b}|2^{f-m})2^k$. Therefore, \bar{c} is the representation of the normalized input $\bar{b}2^{f-m} \in [0.5, 1)$ with resolution 2^{-k} and \bar{v}' is the signed scale factor. Steps 1-2 compute the sign of \bar{b} as a secret integer $\bar{s} \in \{-1, 1\}$ using the protocol LTZ, and then the absolute value of the input $\bar{x} = \bar{s}\bar{b} = |\bar{b}|$. Steps 3-10 determine the scale factor 2^{k-m} using bit decomposition and the protocol PreOR, which returns all prefixes $[y_i] = \bigvee_{j=i}^{k-1} [x_j]$, for $0 \leq i < k$. Finally, steps 11-12 compute (in parallel) the normalized input $\bar{c} = \bar{x}2^{k-m}$ and the signed scale factor $\bar{v}' = \bar{s}2^{k-m}$.

Protocol 3.3. $[y] \leftarrow \text{FPDiv}([a], [b], k, f)$

```

1  $(\theta, \alpha) \leftarrow (\lceil \log \frac{k}{3.5} \rceil, \text{fld}(\text{int}_{2f}(1.0)))$ ;
2  $[w] \leftarrow \text{AppRcr}([b], k, f)$ ;
3  $[x] \leftarrow \alpha - [b][w]$ ;
4  $[y] \leftarrow [a][w]$ ;
5  $[y] \leftarrow \text{TruncPr}([y], 2k, f)$ ;
6 for  $i \in [1.. \theta - 1]$  do
7    $[y] \leftarrow [y](\alpha + [x])$ ;
8    $[x] \leftarrow [x][x]$ ;
9    $[y] \leftarrow \text{TruncPr}([y], 2k, 2f)$ ;
10   $[x] \leftarrow \text{TruncPr}([x], 2k, 2f)$ ;
11   $[y] \leftarrow [y](\alpha + [x])$ ;
12   $[y] \leftarrow \text{TruncPr}([y], 2k, 2f)$ ;
13 return  $[y]$ ;

```

Protocol 3.4. $[w] \leftarrow \text{AppRcr}([b], k, f)$

```

1  $\alpha \leftarrow \text{fld}(\text{int}_k(2.9142))$ ;
2  $([c], [v]) \leftarrow \text{Norm}([b], k, f)$ ;
3  $[d] \leftarrow \alpha - 2[c]$ ;
4  $[w] \leftarrow [d][v]$ ;
5  $[w] \leftarrow \text{TruncPr}([w], 2k, 2(k - f))$ ;
6 return  $[w]$ ;

```

Protocol 3.5. $([c], [v']) \leftarrow \text{Norm}([b], k, f)$

```

1  $[s] \leftarrow 1 - 2 \cdot \text{LTZ}([b], k)$ ;
2  $[x] \leftarrow [s][b]$ ;
3  $([x_{k-1}]^{\mathbb{F}_{2^8}}, \dots, [x_0]^{\mathbb{F}_{2^8}}) \leftarrow \text{BitDec}([x], k, k)$ ;
4  $([y_{k-1}]^{\mathbb{F}_{2^8}}, \dots, [y_0]^{\mathbb{F}_{2^8}}) \leftarrow \text{PreOR}([x_{k-1}]^{\mathbb{F}_{2^8}}, \dots, [x_0]^{\mathbb{F}_{2^8}})$ ;
5 foreach  $i \in [0..k - 1]$  do parallel
6    $[y_i] \leftarrow \text{BitF2MtoZQ}([y_i]^{\mathbb{F}_{2^8}})$ ;
7 foreach  $i \in [0..k - 2]$  do
8    $[z_i] \leftarrow [y_i] - [y_{i+1}]$ ;
9    $[z_{k-1}] \leftarrow [y_{k-1}]$ ;
10   $[v] \leftarrow \sum_{i=0}^{k-1} 2^{k-i-1} [z_i]$ ;
11   $[c] \leftarrow [x][v]$ ;
12   $[v'] \leftarrow [s][v]$ ;
13 return  $([c], [v'])$ ;

```

Security. The division algorithm performs the same sequence of operations regardless of the secret values. The loop counters depend on the desired accuracy of the division operation and the fixed-point representation, which are public parameters. The three protocols do not reveal any secret-shared variable and

all their sub-protocols provide either perfect or statistical privacy. We conclude that **FPDiv** provides statistical privacy.

Complexity. The round and communication complexity of the protocols **FPDiv** and **AppRcr** are shown in Table 3 for $k = 2f$. Observe that most of the invocations are in a small field, \mathbb{Z}_{q_1} or \mathbb{F}_{2^8} , so their communication and computation overhead is low. All shared random bits used in **FPDiv** and its subprotocols can be generated in parallel in 2 rounds. An iteration is computed in 2 rounds (two fixed-point multiplications in parallel).

The complexity of **FPDiv** is clearly dominated by the initial approximation, especially the normalization step. For example, if $k = 112$ and $\theta = 5$ (≈ 112 bits accuracy), steps 3-12 of **FPDiv** are computed in 12 rounds, and **AppRcr** adds 29 rounds (27 rounds for **Norm**), giving a total of 43 rounds. A variant of **FPDiv** with positive divisor is sufficient in many applications and can be computed in 33 rounds by skipping the steps 1, 2, and 12 of **Norm**.

Note that the building blocks in Table 2 are optimized for low communication complexity. The round complexity of **FPDiv** can be reduced using building blocks that trade off higher communication complexity for a lower number of rounds.

4 Conclusions

Business applications of secure computation need a protocol family that provides operations with all primitive data types and allows secure protocol composition and efficient application development. We presented a protocol family that fills an important gap by enabling secure computation with rational numbers.

Fixed-point representation offers the most efficient encoding of rational numbers as well as efficient protocols for the most frequent operations: addition, subtraction, multiplication, and comparison. Division is simple for public divisor, but becomes quite complex when the divisor is secret. On the other hand, secure arithmetic with floating-point numbers is clearly not practical.

The protocols have been implemented in Java and tested in complex applications like secure linear programming using Simplex (with a variant of the division protocol that was optimized for multiple divisions with the same divisor).

On-going work focuses on improving the efficiency of division and adding protocols for secure evaluation of other mathematical functions.

Acknowledgements. Part of the work presented in this paper was funded by the European Commission through the grant FP7-213531 to the SecureSCM project. The authors thank the anonymous reviewers for their helpful comments.

References

1. Algesheimer, J., Camenish, 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)

2. Atallah, M., Blanton, M., Deshpande, V., Frikken, K., Li, J., Schwarz, L.: Secure Collaborative Planning, Forecasting, and Replenishment (SCPFR). In: Multi-Echelon/Public Applications of Supply Chain Management Conference (2006)
3. Atallah, M., Bykova, M., Li, J., Frikken, K., Topkara, M.: Private Collaborative Forecasting and Benchmarking. In: Proc. WPES 2004, Washington (2004)
4. Brandt, F.: How to obtain full privacy in auctions. International Journal of Information Security 5(4), 201–216 (2006)
5. Catrina, O., Dragulin, C.: Multiparty Computation of Fixed-Point Multiplication and Reciprocal. In: Proc. 20th International Workshop on Database and Expert Systems Application (DEXA 2009), pp. 107–111. IEEE Computer Society, Los Alamitos (2009)
6. Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 342–362. Springer, Heidelberg (2005)
7. Cramer, R., Damgård, I., Maurer, U.: General Secure Multi-Party Computation from any Linear Secret-Sharing Scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
8. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J., 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)
9. Damgård, I., Nielsen, J., Toft, T., Pagter, J.I., Jakobsen, T., Bogetoft, P., Nielsen, K.: A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006)
10. Damgård, I., Thorbek, R.: Non-interactive Proofs for Integer Multiplication. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 412–429. Springer, Heidelberg (2007)
11. Damgard, I., Thorbek, R.: Efficient Conversion of Secret-shared Values Between Different Fields. In: Cryptology ePrint Archive, Report 2008/221 (2008)
12. Ercegovac, M.D., Lang, T.: Digital Arithmetic. Morgan Kaufmann, San Francisco (2003)
13. Fouque, P., Stern, J., Wackers, G.: CryptoComputing with Rationals. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 136–146. Springer, Heidelberg (2003)
14. From, S.L., Jakobsen, T.: Secure Multi-Party Computation on Integers. Master's thesis, Univ. of Aarhus, Denmark, BRICS, Dep. of Computer Science (2006)
15. 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)
16. Markstein, P.: Software Division and Square Root Using Goldschmidt's Algorithms. In: Proc. 6th Conference on Real Numbers and Computers, pp. 146–157 (2004)
17. Masayuki Ito, N.T., Yajima, S.: Efficient Initial Approximation for Multiplicative Division and Square Root by a Multiplication with Operand Modification. IEEE Transactions on Computers 46(4) (1997)
18. 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)
19. Toft, T.: Primitives and Applications for Multi-party Computation. PhD dissertation, Univ. of Aarhus, Denmark, BRICS, Dep. of Computer Science (2007)
20. Toft, T.: Solving Linear Programs Using Multiparty Computation. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 90–107. Springer, Heidelberg (2009)

Implementing a High-Assurance Smart-Card OS

Paul A. Karger¹, David C. Toll¹, Elaine R. Palmer¹, Suzanne K. McIntosh¹,
Samuel Weber^{1,*}, and Jonathan W. Edwards²

¹ IBM Thomas J. Watson Research Center,
P.O. Box 704, Yorktown Heights, NY 10598, USA

`karger@watson.ibm.com, {toll,erpalmer,skranjac,jone}@us.ibm.com`

² IBM Global Business Services
1500 Aristides Blvd., Lexington, KY 40511, USA

Abstract. Building a high-assurance, secure operating system for memory constrained systems, such as smart cards, introduces many challenges. The increasing power of smart cards has made their use feasible in applications such as electronic passports, military and public sector identification cards, and cell-phone based financial and entertainment applications. Such applications require a secure environment, which can only be provided with sufficient hardware and a secure operating system. We argue that smart cards pose additional security challenges when compared to traditional computer platforms. We discuss our design for a secure smart card operating system, named Caernarvon, and show that it addresses these challenges, which include secure application download, protection of cryptographic functions from malicious applications, resolution of covert channels, and assurance of both security and data integrity in the face of arbitrary power losses.

1 Introduction

The design of higher security operating systems has been studied since the 1960s. However, most of these designs have been for relatively large computer systems. This paper examines a series of issues that a high-security operating system must face to be able to run in an extremely memory-limited environment, such as a smart card, a cell phone, a small PDA, or other constrained pervasive devices. For a good overview of smart card technology in general, see [11].

As the prime uses of smart cards are identification, authorization and encryption, it is crucial that sufficient trust be established between different applications executing on the same card. The lack of a trusted secure operating system for smart cards has resulted in some users having a “necklace of cards”, each one hosting a single application. The Caernarvon project was started to create such a secure smart card operating system. A very high-level overview of the Caernarvon system can be found here [14]. In contrast, this paper focuses on a

* Now with the National Science Foundation, 4201 Wilson Boulevard, Arlington, Virginia 22230, `samweber@acm.org`

number of challenges faced in actually implementing a high-assurance operating system on a smart card.

Most existing smart card systems have required that all applications be written together and loaded onto the card prior to the card being issued, because smart card processors did not support internal security controls, and all applications had to be mutually trusting. However, with the development of new smart card processors with internal security features (see Section 1.1), much stronger security could be provided. Thus, one primary goal of the Caernarvon project was to build a smart card operating system capable of supporting the download of applications that might be actively hostile, both to each other and to the underlying operating system. To be able to protect against such potentially hostile applications, the Caernarvon security policy was chosen to be mandatory access controls (MAC), because only such controls can effectively deal with applications that may contain Trojan horses. (See Section 3.)

This paper will focus on the specific challenges that must be faced to build a high-assurance operating system for such memory-constrained devices as smart cards. Sections 4 and 5 examine in detail the following security aspects of the operating system that are particularly different from previously described work on high-assurance operating systems: a hierarchical file system structure to reduce memory consumption, elimination of global address space covert channels that are unique to smart cards, reducing memory consumption of mandatory access classes, capability-based discretionary access controls, reliable persistent storage in the presence of power failures and memory write errors, secure application download, secure cryptographic implementations without trusted applications, and secure chip initialization without slowing down manufacturing lines.

1.1 Feasibility

The first question in the Caernarvon project was “Would it be feasible at all to build such a system?”. When IBM Research first considered the project, the answer was, “No.” Early smart card chips did not have adequate hardware support for security, such as separate supervisor and user states and memory protection.

The project only became possible when the Philips (now NXP) SmartXA chip was introduced as the first smart card processor to meet those needs. Karger, Toll and McIntosh [9] discuss these hardware requirements in much more depth. Since then, other vendors have also introduced chips that meet the requirements to support the Caernarvon operating system. However, as discussed in [9], not all chips that claim to support memory protection can do so without introducing covert channel problems.

The current SmartXA2 chip supports a relatively large amount of memory for a smart card chip: 7 Kbytes of RAM, 256 Kbytes of ROM, and 144 Kbytes of EEPROM. However, compared to most contemporary secure computer system projects, that amount of memory is extremely tiny. Note that the numbers are in kilobytes, not megabytes or gigabytes, and there is no external peripheral memory, such as a disk. All memory must fit on the single chip.

2 Applications of the Technology

Many applications could benefit from a high assurance smart card operating system. Generally, those applications have data or software from multiple parties co-residing on the same card, and require some level of data sharing between the parties. The trust relationship of those parties ranges from friendly to mistrustful to hostile. The threats addressed range from honest mistakes in software to attacks by financially-motivated cardholders to industrial espionage to comprehensive logical and physical attacks by hostile adversaries and insiders. Below is a list of sample applications:

- an electronic passport issued by one government, with electronic entry/exit timestamps added by other (possibly hostile) governments, described in [8].
- a corporate/school campus card, with multiple application providers for copiers, vending machines, public transit, and building access
- an ID card for coalition military forces for building or computer access
- a subscriber identity module for mobile phones to hold credentials for financial institutions, governments, and phone service providers, etc.

There are roadblocks hindering the commercialization of a high assurance smart card operating system, such as: significant investment in time and funding is required by multiple institutions; the skills required cross several domains; some existing smart card application specifications have mandated protocols that preclude a high level of security. For example, the electronic passports specified by the International Civil Aviation Authority require the use of weak cryptographic authentication protocols, and the protocols of the Federal Employee Personal Identity Verification program require some very sensitive information to be transmitted in unencrypted form. In attempting to resolve these issues, the Caernarvon development led to a clearer understanding that in order to meet many security goals, privacy goals must also be met [8] .

3 Background – Security Policy and Authentication

The Caernarvon system builds on previous work on mandatory security policies to provide multi-level security. Mandatory security was chosen specifically because a major goal of the Caernarvon system is to support downloading of multiple applications from multiple application providers, who may be mutually hostile. Caernarvon system security is enforced using a mandatory security policy described more completely in [12] that is based on modifications of the Bell and LaPadula secrecy model [2] and the Biba integrity model [3].

Enforcement of a meaningful security policy requires that there be a secure mechanism to ensure that the use of the desired access classes is valid and correct. This authentication must be performed by the smart card's operating system itself and not by an application, so that the operating system is guaranteed, and can guarantee to others, that the authentication has been correctly completed. The smart card operating system can use this knowledge to safely grant or deny

the host system access to files and other system objects on the card. Space does not permit including the full description of this authentication protocol which is available in [13].

4 Security Design Challenges

4.1 File System

The Caernarvon system implements a smart card file system. Besides the challenges caused by the specification and hardware restrictions, the file system must also enforce the system's security policy. There also must be a quota mechanism and support for memory-mapped files. The file system is the major repository of system state, and hence security is crucial to its design and implementation.

Caernarvon implements an ISO 7816-4 File System, with certain security extensions described in this section. This file system is implemented by two separate components, namely the Persistent Storage Manager (PSM) and the file system abstraction layer.

The PSM provides and manages memory objects, that is, blocks of persistent storage. Smart cards can have their power sources removed unexpectedly, corrupting in-progress memory writes. An important purpose of the PSM is to maintain the integrity of the memory objects, allowing other system components to ignore power interruption issues.

The PSM is not exposed to user mode applications. The file system layer sits on top of the PSM, and it is visible to applications.

File System Structure. The ISO 7816-4 standard defines a hierarchical file system, which has a single MF (Master File, equivalent to “root” in Unix), which contains DFs (Dedicated Files, otherwise known as directories) and/or EFs (Elementary Files). The Caernarvon system extends this model by defining another file type, an “XF” (Executable File), which are executable programs.

Unlike most other file systems, the Caernarvon MF and DFs have no table of the files that they contain; instead, each DF and EF (including each XF) in the system has a pointer to its parent, as shown in Figure 1. This saves the space that would be occupied by the list of file names and the reference to each file; it also

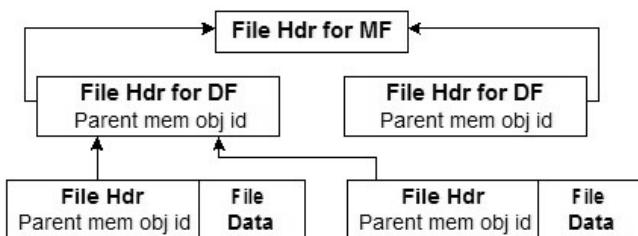


Fig. 1. Directory Structure Implementation

means that, when files are created or modified, there is no need to update the corresponding DF entry for that file. However, this design has the drawback that file system searches require examining every file in the system. Since the amount of persistent memory is extremely limited (< 256 Kbytes), there can be only a small number of files in any given smart card, and this extended search cannot create a performance problem. Obviously, this space/time tradeoff algorithm does not scale to large memories with lots of files. Eventually, smart cards will have much larger persistent memories available. In that case, the highly modular and layered design of Caernarvon would make switching to a more conventional directory structure quite easy.

An obvious, but incorrect optimization for the linear search would be to cache the information about recently opened files. However, such a cache could not fit in RAM (only 7 kbytes total) and would itself consume the scarce persistent memory that we are trying to save. Furthermore, the open file table itself serves as a small cache, as long as the file remains open. Due to the slow communications speeds of smart cards, switching applications consumes more time than any file system caching would save.

The MF, each DF, and the headers and data areas of all files, are each held in a PSM memory object, the size and location of which are defined by the PSM's object descriptors, as shown in Figure 2.

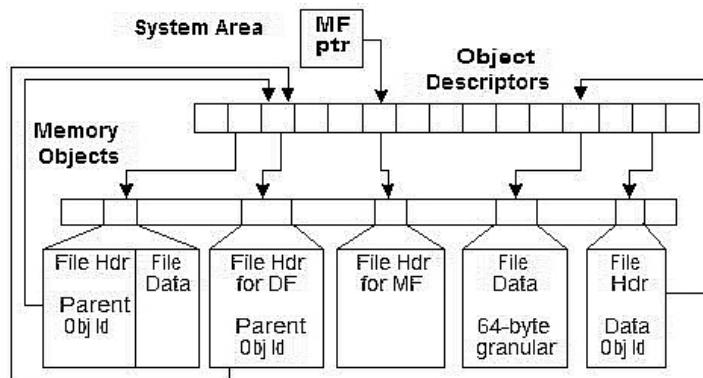


Fig. 2. PSM Memory Objects for Files

In general, files are packed so as to minimize the space and number of memory blocks required for their storage. In the case of most data files, the file header and the file data area are packed together into a single contiguous memory area. However, Caernarvon provides a facility for files to be memory mappable - this is used, in particular, for the code of executable programs. Data that is memory mappable must lie on a physical memory boundary and have a size granularity specified by the processor's memory protection unit. This means that mappable files have one memory object for the header and another for the data. Figure 2 shows the arrangement of the memory objects for both types of file.

Global Address Spaces—DFNames. The ISO 7816-4 file system names all files as numbers. While that simplifies the file system, it makes it difficult for end users to select an application by name. Remembering the file numbers is not likely to be acceptable. As a result, ISO 7816-4 defines the concept of a DF Name that is used to select an executable program from outside the card. A DF name is a string name assigned to the DF containing the application. The DF Names are unique to the card, and, (as defined in ISO 7816-4) constitute a global address space.

Global address spaces cause two different operational problems. First, if two different application developers happen to choose the same DFName, then the first such name loaded onto a particular card will win. Since ISO 7816-4 assumed all applications would be preloaded onto the card, this was never a problem. Once you have multiple application providers downloading applications to a card after issuance, the name collision problem can become serious. Second, such name collisions could be used as a covert channel to bypass mandatory access controls.

Caernarvon avoids these problems by making the DF Name space into a name space that is private to the current access class. The ISO 7816-4 rules for DF Names are then applied within each access class, rather than system wide. Within an access class, DF Names must be unique, but the same DF Name may be repeated in a different access class.

Storing Access Classes in the File System. In most previous operating systems that supported mandatory access controls, the access classes of files and directories are stored with their associated files and directories, either in file headers or in the directory branches. Furthermore, Caernarvon access classes are designed to support multi-organizations, as discussed in [14, Section 3.1]. These multi-organizational access classes can be quite large to represent, and every file and directory may have its own unique access class. In a smart card with very limited amounts of memory, storing large numbers of access classes could be a severe problem.

To reduce access class memory usage in Caernarvon, two steps have been taken. First, we note that due to restricted memory for storing applications and data; no one smart card will need to use more than a small number (< 256) of distinct access classes. Therefore, the File System only stores each multi-organizational access class once in a table, and need store only an 8-bit index into that table with each file or directory.

Caernarvon follows the Multics practice [15] that while a child directory (DF) may have a different (higher) access class than its parent DF, ordinary files (EFs and XFs) must have the same access class as their parent DF. This allows us to save additional memory by requiring that files (EFs and XFs) do not have associated access classes, while directories (DFs) may but are not required to. DFs, EFs and XFs without an access class inherit the access class of their nearest parent DF that does have its own access class.

A program, if it has appropriate access to a file, may change the access class of that file, for example to raise its secrecy level or to lower its integrity level, but

doing so may remove the program's access to the file. In this case, any open file handles for the file in question are marked, and then the program can perform no more operations such as read and write using the file handle until it has closed the file and re-opened it. If the program no longer has access to the file then the re-open will fail. This re-open approach avoids the kernel complexity of the Multics revocation approach and originates in the design of the DEC A1-secure virtual machine monitor [10].

This facility to change the access class of a DF (and hence of all the files within it) can be used to move data from one access class to another. Thus if a program at AC a wishes to move a DF (also at AC a) to AC b then the program must change the DF to the AC $a+b$. Note that this is quite legitimate - a program may always change a file to a higher secrecy level. Having done this, the program, which is still at AC a , no longer has access to the DF. At this stage, a special guard process must be run; this is an evaluated application that has been certified as fit to perform the downgrade of secrecy level $a+b$ to b . This guard program would verify that it is indeed legitimate to downgrade the secrecy of the DF, and if all is well, change the AC of the DF to b . Note that moving a DF must also move disk quota as discussed below.

Quota. In order to protect against denial of service attacks when one application takes all the persistent storage on the card, Caernarvon provides a quota facility. This also enables the card issuer to control (and charge for) the amount of space on the card used by each application provider. Covert channels whereby a Trojan horse could signal by either allocating all memory or freeing some are prevented.

Most contemporary operating systems use special quota accounts to charge for file space. However, such quota accounts require a great deal of management software. Instead, the Caernarvon system uses a very old approach (described below) from the Multics operating system [15, section 3.7.3.1], that is simpler to implement. The Multics approach was replaced by quota accounts, because such accounts were easier for time-sharing system users to manage. However, Caernarvon quota will be managed by application providers, not end-users, so the slightly harder management interface should not be a problem.

Each DF may (but need not) have a quota; if the DF does not have its own quota then that DF and all the files within it are charged against the quota of the nearest parent DF that does have a quota. When a new top-level DF is created for an application, then that DF would normally be allocated its own quota. The application can quite legitimately give some of its quota to a DF below its own top-level DF. If a DF is moved from one AC to another (as described above), then the quota occupied by that DF and all the files within it is also moved to the new parent DF of the DF that has been moved. The combined atomic operation of moving a DF from one AC to another together with its quota is totally new and is needed to avoid covert storage channels. Most existing mandatory access control systems avoid the channels by restricting such moves to human users through a trusted path, but a smart card has no direct human interface.

Discretionary Security Policy - Capabilities. As discussed above in Section 3, the primary security policy of the Caernarvon system is a mandatory access controls. This is because mandatory access controls are specifically designed to deal with malicious applications code. However, mandatory access controls do not provide the very fine-grained control that users can get from discretionary access controls, such as access control lists or permission bits. As the Caernarvon design progressed, we became concerned that including both a powerful mandatory access control mechanism as well as a full access control list implementation would exceed the very limited amount of memory that can be committed to the smart card operating system.

As a result, we chose to implement a compromise discretionary access control system, based on a very restricted form of capabilities [4]. These are discretionary security policy rules that may be associated with an individual executable program (an XF). These capabilities take two forms that are based on our assessment of the most common needs of smart card developers and are easy to implement in only a small amount of code:

1. a bit array that specifies whether that program is permitted to issue certain supervisor calls (SVCs). For example, there is a special, evaluated, Admin Application issued with the system that is used for the administration of (in particular, the creation of) Access Classes and top-level DFs for applications. This program uses certain special SVCs for the administration of ACs; the capability bit for this group of SVCs is unset for *every* other XF in the system, so that no other program can issue those SVCs.
2. there can be special access rules to allow or forbid access to individual files by this program.

It is important to note that these capabilities are not a fully general capability system, as defined by Dennis and Van Horn [4]. In particular, Caernarvon capabilities cannot be passed from one process to another.

4.2 Persistent Storage Manager - PSM

In the Caernarvon system, the physical blocks of storage are managed by the Persistent Storage Manager (PSM). The principal client of the PSM is the File System; the PSM is also used by the Access Class Manager and the Key Management system.

Figure 3 shows how the PSM divides persistent storage. The System Area at the beginning of EEPROM contains information that the operating system uses to locate its internal data structures and persistent state. Items in the System Area are expected to be set up when the initial EEPROM image is built, and subsequently is never changed. Everything else is held in a memory object; every object has a unique ID, and is always referenced via this ID. Every memory object is in turn described (in particular, its location and size) by an object descriptor; a memory object is located by searching the object descriptors for the descriptor with the requested ID. This allows for the possibility of either the memory object and/or its descriptor moving in physical storage.

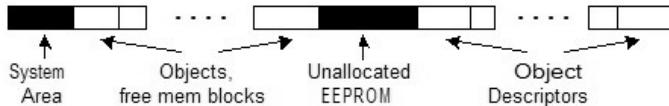


Fig. 3. PSM Use of Persistent Storage

Management of Storage Objects. Figure 2 shows how file system objects are stored in PSM memory objects. In older smart card systems, where the file system was created during card personalization or initialization, the file system was effectively static, that is there was no file creation or deletion after issuance. However, a Caernarvon file system is dynamic—the number of files and/or their sizes will change since the Caernarvon system allows for the installation or removal of applications, and the creation and deletion of files after the card has been issued.

This means that the PSM must manage any possible fragmentation of memory objects. The Caernarvon PSM avoids this completely by always ensuring that any memory object is stored in a single contiguous area of memory. However, this in turn causes another problem, in that as files are created, extended (or shrunk), or deleted, free storage will become fragmented. When a file is to be created, there may not be a single block of free memory available that is large enough. Alternatively when enlarging a file, there may be no immediately adjacent free memory.

A traditional approach to this problem would be to do a garbage collection: move in-use objects to contiguous storage and coalesce the free areas. In Caernarvon, we do not do a complete collection. Instead, when a memory allocation is attempted but no suitably-sized memory block is available, the PSM determines the minimal number of occupied blocks that need to be moved to create a large enough free memory space. After this compaction operation, memory may still be fragmented, but the number of memory writes will have been kept low in order to preserve the lifetime of the memory locations. It should be noted that compaction will certainly require multiple read and write operations to persistent storage, both of the memory objects and of the descriptors. These operations are time consuming, so minimizing their number and frequency is also important to maintain acceptable performance.

Weaknesses of Persistent Storage. Write operations to EEPROM or Flash memory are slow, usually taking 4 to 6 milliseconds each. Further, the write block size for EEPROM is limited, for example, to 128 bytes. Thus writing any significant amount of data to a file is likely to take multiple write operations, plus additional writes to update the control block information. There is also the problem that the smart card is powered only when it is in the reader. These factors mean there is a significant risk of a file write operation being interrupted and not fully completed.

A solution to the power interruption problem is to ensure that all memory transactions, for example a request to extend a file and update its contents, be

treated as a single atomic operation. That is, the entire transaction, including all of the multiple write operations required, must be completed in its entirety, or not performed at all. The PSM maintains a *backtrace buffer* where, when persistent storage is to be updated, the old values are stored before the new data is written. This is done for every step of the operation - the backtrace buffer entries are cleared only when the entire transaction is completed. When the card is powered-up, if the backtrace buffer is not empty, the items in the backtrace buffer are restored one-by-one, in the reverse order to which the original steps were performed. When this is complete, the state of the memory is as if the transaction had never been started. There is one additional complication, in that, when powered up, the smart card cannot immediately start these backtrace operations, but is required to respond to the smart card reader within 40,000 processor clocks. This allows no time to perform the pending backtrace buffer operations. However, when the reader sends the first command to the card, the card can request a “waiting time extension” to delay its response to the reader; this request can be repeated as many times as are necessary to complete the pending backtrace operations.

Smart card persistent memory, EEPROM or Flash, has a limited number of write cycles before it starts to fail, for example between 100,000 and 500,000 for EEPROM, and only 10,000 for Flash. The PSM takes two measures to compensate for memory corruption due to the cells wearing out. First, once every write to persistent memory is completed and before control is returned, the low-level code that did the write compares the updated contents of memory with the data in the caller’s buffer (in RAM). If a mismatch is detected, an error is returned; in this case, the data that was to be written is still available in the buffer. Second, the PSM places a checksum on every memory object under its management, including the control blocks or descriptors that define the memory objects. This checksum is verified on read operations to enable the detection of memory failures—an attempt is then made to recover the lost data byte(s). Once a memory error is detected, the memory area in question is marked as bad, and the data is re-written to a different location.

The backtrace buffer for atomic transactions is well known in data base design [6], but its use throughout a file system in conjunction with mechanism to recover from memory wearing out is unprecedented in smart cards. Of course, the backtrace buffer is itself an area of memory that has frequent write operations, and hence can wear out. When errors occur within the backtrace buffer, the PSM will attempt to move the backtrace buffer to a new area of memory.

4.3 Implementing Application Download

A primary aim of the Caernarvon system is to allow for the secure download of applications in the field. The card issuer can allow or forbid the download of applications, and when download is permitted, can control which organizations are allowed to install their applications on the card and how much file quota they may use. Note that in the context of download, the term “application” is

not limited to just the XFs; it may also encompass the associated DFs, EFs, file quota, keys, etc.

The download process can be divided into two main steps, namely the creation of access classes (with the appropriate file quota) for organizations that currently are not present on the card, and then the download of application files (including executable programs) for an organization that is present on the card. Obviously, download of an application for a new organization requires the completion of both of these steps.

Creation of Access Classes. The creation of a new access class is a tricky operation on a smart card, because the card is physically in the possession of an end user who may not be privileged to create access classes. The smart card also does not have a system administrator or security officer who can perform such operations. Requiring the card holder to carry the card back to the card issuer would be unacceptable to most customers.

Instead, the Caernarvon operating system includes secure cryptographic protocols to (a) create the Access Class and (b) to create the necessary top-level DF associated with the new access class, and set its allocated quota. These protocols will require a full future paper to explain and are not further discussed here.

File Download. Once an organization has been authorized to be present on a Caernarvon card, that is, once any necessary access class(es) have been created, then that organization may download such files as it needs, subject to the file space the quota imposed by the card issuer.

A file is downloaded simply by authenticating at the appropriate access class, running a program to create any required DFs and EFs, and writing the appropriate data to those files. An executable file, once it is downloaded, must be “activated” to convert the file from an EF to an XF.

The card issuer may wish to restrict the programs that are run. For example, only approved applications or Common Criteria evaluated applications might be allowed. To implement this level of control, the Caernarvon kernel will check for digital signatures, either from the card issuer or the Common Criteria certifier or both.

4.4 Cryptographic Challenges

The Caernarvon system includes a cryptographic library to ensure that the cryptographic algorithms, such as DES, triple-DES, AES, RSA, DSA, and ECC are correctly and securely implemented in a side-channel free fashion. This is discussed in section 3.6 of [14].

In addition to proper cryptographic algorithm implementation, it is essential that cryptographic key management also be implemented securely. If the application handles the key itself, it may inadvertently leak information (for example, some bits of the key) by such simple operations as copying the key from one memory location to another. Further, there is nothing to prevent a malicious program from deliberately leaking the key to outside the smart card.

Caernarvon provides secure key management facilities within the kernel. Keys can be loaded into the card by the kernel, so that the application never sees the key; the application refers to the key by a name (actually a handle) of its choosing. The keys are effectively stored in the file system with file IDs for names and hierarchical file paths, the same as for regular files. This avoids covert channel problems that could arise in the names of keys, if the keys were stored in a flat file system. However, the key names are a separate name space from the file names, and, to ensure security of the key, these key “files” cannot be accessed as regular files. In addition, keys can be marked by purpose, such as to be encryption keys or signing keys; the kernel can then prevent a signing key from being used for encryption, or vice versa. This prevents certain cryptographic weaknesses where a key is used for more than one purpose.

Unfortunately, some smart card standards (such as the Global Platform standard [1]) require that the keys be visible to applications (or in the Global Platform case, the application security domain). To satisfy this requirement, Caernarvon also supports a “raw” key mode, where the keys are handled entirely by the application. Access to the crypto co-processor must still be mediated by the kernel to avoid both object re-use and covert channel issues. While applications may find this mode a necessity, its use is strongly discouraged, since Caernarvon cannot ensure any security for these raw keys.

Another problem that can arise is that a program can develop its own cryptographic code, for example to implement an algorithm devised specially for that application. Running such code on top of a high security kernel provides no guarantee of the quality of the implementation of the cryptography, in particular immunity to side channel attacks. Again, the only way to avoid the problem is to design the application to use only the strong crypto (and secure key management) provided by the Caernarvon kernel.

5 Chip Initialization

Smart card chips containing the Caernarvon system are intended to be high security devices. Therefore, it is imperative that each individual chip be secure right from the point of manufacture, with no opportunity for the chip or its contents to be compromised while in the factory or during delivery. Manufacture and initialization are the most security-sensitive stages in the chip’s entire lifecycle, because the chip is in its most vulnerable, exposed state, and it is during these stages that important roles and security parameters are set for the remainder of the chip’s lifecycle. A fundamental assumption is that the manufacturing line is secure, which requires the chip manufacturer to assure that it is safe from tampering, collusion, theft, and other threats, including those from insiders.

In a typical smart card chip manufacturing facility, manufacturing test software is built into each chip to assure the viability of the chip. The test software tests the processor, memory subsystem, internal peripherals, and other subsystems such as cryptographic accelerators. These tests typically destroy the contents of writable memory. Thus, the chips cannot be initialized with unique

persistent data until all manufacturing tests are complete. Once the manufacturing tests have completed successfully, the test software downloads a copy of the initial file system *for that chip*, decrypting it with a strong cryptographic key held in read-only memory, and used only once (during manufacture). The image of each chip’s initial file system is pre-calculated by the chip manufacturer by filling in the values of security-relevant data items in predefined locations. Some of these items include certificates, private and public keys, Diffie-Hellman [5] key parameters used for authentication, a chip-unique seed for random number generation, initial access classes, and uncertified application binary files. Because it is difficult for the smart card chip’s processor to meet the demanding speed required by the manufacturing line, these security-relevant items are not typically generated on-chip. Instead, they must be generated and digitally signed in advance in hardware security modules such as the IBM 4764 [7], and injected into each smart card chip at very high speeds. The chip manufacturer also digitally signs a certificate unique to each chip, thus enabling off-chip applications to verify (as part of an interactive authentication protocol) that communications come from an authentic Caernarvon chip, and not an imposter. This chip certificate includes a serial number and public key unique to each chip, a chip type and configuration code, the Caernarvon software hash value, version number, and evaluation assurance level.

The chip makes use of a public key hierarchy to establish identities and public keys of the actors that set the final configuration of the chip’s software and data. Actors include the chip manufacturer, the smart card enabler, the smart card personalizer, the smart card issuer, the application certifying body, and others. During initialization, some of the public keys and roles of these actors are set by the chip manufacturer. Others are initialized later in the chip’s lifecycle, and can only be set by an actor authenticated in a specific role.

After the test code has completed the initialization of a chip, it must securely disable itself so that it can never be run again, even in the face of physical attacks on the chip. At this point in the chip’s lifecycle, the OS is fully functional and secure. Thus, when the chip is first powered up for any purpose outside of the manufacturing line (for example, for personalization of the smart card for the end user), the Caernarvon system is in control. In particular, full system authentication is required to perform any operations such as personalization or the installation of applications.

6 Conclusion

The Caernarvon operating system project has shown the feasibility of building smart card systems with much higher levels of security than is common practice. In particular, it is possible for a smart card to download and execute applications from multiple mutually-distrusting sources, but still prevent them from interfering with each other or with the operating system itself.

These goals required reconsideration of many traditional smart card software practices (see Section 4.1), as well as solving many security problems that are

not present in larger-scale computers. In particular, new security approaches had to be developed to deal with the extreme memory constraints of a smart card, and these new techniques had to be applied throughout the operating system. Despite these challenges, the Caernarvon operating system was completely free of covert storage channels.

The present status of the Caernarvon system is discussed in [14, Section 7]. An alpha test version has been implemented and runs on a smart card hardware emulator. However, commercial viability is still undetermined, because the complete OS has not been released as a product. Several major pieces of the design have also been transferred to other IBM projects, as discussed in [14, Section 6].

What this paper has shown is how such a high-security system can actually be implemented in the extremely memory constrained environment of a smart card, yet still support a very general mandatory access control model that can protect against essentially arbitrary malware attacks.

Acknowledgements

The Caernarvon project involved work by a number of people in addition to the authors of this paper, and we wish to acknowledge the contributions of, from IBM: Vernon Austel, Ran Canetti, Suresh Chari, Vince Diluocco, Günter Karjoth, Gaurav Kc, Rosario Gennaro, Hugo Krawczyk, Mark Lindemann, Tal Rabin, Josyula Rao, Pankaj Rohatgi (now with Cryptography Research), Helmut Scherzer (now with Giesecke & Devrient), and Michael Steiner; from atsec: Helmut Kurth; from Philips: Hans-Gerd Albertsen, Christian Brun, Ernst Haselsteiner, Stefan Kuipers, Thorwald Rabeler, and Thomas Wille; from the University of Augsburg: Wolfgang Reif, Georg Rock and Gerhard Schellhorn; from the DFKI: Axel Schairer and Werner Stephan; and from the German BSI: Stefan Wittmann. We must also thank Wietse Venema for his very useful suggestions on improving the paper.

References

- [1] Béguelin, S.Z.: Formalisation and verification of the GlobalPlatform card specification using the B method. In: Barthe, G., Grégoire, B., Huisman, M., Lanet, J.-L. (eds.) CASSIS 2005. LNCS, vol. 3956, pp. 155–173. Springer, Heidelberg (2006)
- [2] Bell, D.E., LaPadula, L.J.: Computer Security Model: Unified Exposition and Multics Interpretation. In: ESD-TR-75-306, The MITRE Corporation, Bedford, MA, HQ Electronic Systems Division, Hanscom AFB, MA (June 1975), <http://csrc.nist.gov/publications/history/bell76.pdf>
- [3] Biba, K.J.: Integrity Considerations for Secure Computer Systems. In: ESD-TR-76-372, The MITRE Corporation, Bedford, MA, HQ Electronic Systems Division, Hanscom AFB, MA (April 1977), <http://handle.dtic.mil/100.2/ADA039324>
- [4] Dennis, J.B., Van Horn, E.C.: Programming semantics for multiprogrammed computations. ACM Commun. 9(3), 143–155 (1966)

- [5] Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Trans. on Information Theory* IT-22(6), 644–654 (1976)
- [6] Gray, J.N.: Notes on Data Base Operating Systems. LNCS, vol. 60, pp. 393–481. Springer, Berlin (1978)
- [7] IBM 4764 Model 001 PCI-X Cryptographic Coprocessor. Data Sheet G221-9091-05, http://www-03.ibm.com/security/cryptocards/pdfs/4764-001_PCIX_Data_Sheet.pdf
- [8] Karger, P.A., Kc, G.S., Toll, D.C.: Privacy is essential for secure mobile devices. *IBM Journal of Research and Development* 53(2) (2009)
- [9] Karger, P.A., Toll, D.C., McIntosh, S.K.: Processor requirements for a high security smart card operating system. In: Eighth e-Smart Conference, Eurosmart, Sophia Antipolis, France, September 19-21 (2007), IBM Research Div. Rpt. RC 24219 (W0703-091), <http://domino.watson.ibm.com/library/CyberDig.nsf/Home>
- [10] Karger, P.A., Zurko, M.E., Bonin, D.W., Mason, A.H., Kahn, C.E.: A retrospective on the VAX VMM security kernel. *IEEE Trans. on Software Eng.* 17(11), 1147–1165 (1991)
- [11] Rankl, W., Effing, W.: Smart Card Handbook: Third Edition. John Wiley & Sons, Chichester (2003)
- [12] Schellhorn, G., Reif, W., Schairer, A., Karger, P., Austel, V., Toll, D.: Verification of a formal security model for multiapplicative smart cards. In: Cuppens, F., Deswarthe, Y., Gollmann, D., Waidner, M. (eds.) ESORICS 2000. LNCS, vol. 1895, pp. 17–36. Springer, Heidelberg (2000)
- [13] Scherzer, H., Canetti, R., Karger, P.A., Krawczyk, H., Rabin, T., Toll, D.C.: Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 181–200. Springer, Heidelberg (2003)
- [14] Toll, D.C., Karger, P.A., Palmer, E.R., McIntosh, S.K., Weber, S.: The Caernarvon secure embedded operating system. *Operating Systems Review* 42(1), 32–39 (2008)
- [15] Whitmore, J., Bensoussan, A., Green, P., Hunt, D., Kobziar, A., Stern, J.: Design for Multics security enhancements. In: ESD-TR-74-176, Honeywell Information Systems, Inc., HQ Electronic Systems Division, Hanscom AFB, MA (December 1973), <http://csrc.nist.gov/publications/history/whit74.pdf>

Unlinkable Priced Oblivious Transfer with Rechargeable Wallets

Jan Camenisch¹, Maria Dubovitskaya^{1,2,3}, and Gregory Neven¹

¹ IBM Research – Zurich

² IBM Russian Systems and Technology Laboratory

³ National Research Nuclear University MEPhI, Russia

Abstract. We present the first truly unlinkable priced oblivious transfer protocol. Our protocol allows customers to buy database records while remaining fully anonymous, i.e., (1) the database does not learn who purchases a record, and cannot link purchases by the same customer; (2) the database does not learn which record is being purchased, nor the price of the record that is being purchased; (3) the customer can only obtain a single record per purchase, and cannot spend more than his account balance; (4) the database does not learn the customer’s remaining balance. In our protocol customers keep track of their own balances, rather than leaving this to the database as done in previous protocols. Our priced oblivious transfer protocol is also the first to allow customers to (anonymously) recharge their balances. Finally, we prove our protocol secure in the standard model (i.e., without random oracles).

1 Introduction

Suppose you want to buy a digital item from a website that charges per purchased item, and that sells different items at different prices. You have reasons to believe, however, that the website is making a lucrative parallel business out of selling information about your shopping behavior to your competitors. For example, you may work for a pharmaceutical company and buy information about particular DNA genome sequences from a database, or you may work for a high-tech company and buy patents from a patent database. The list of purchased records from either of these databases certainly reveals precious information about your company’s research strategies. How do you prevent the database from gathering information about your shopping behavior while still allowing the database to correctly charge you for the purchased items?

What we need here is a *priced oblivious transfer* (POT) protocol [1], where customers load an initial amount of money into their pre-paid accounts, and can then start downloading records so that (1) the database does not learn which record is being purchased, nor the price of the record that is being purchased; (2) the customer can only obtain a single record per purchase, and cannot spend more than his account balance; and (3) the database does not learn the customer’s remaining balance. All known POT protocols require the database to maintain customer-specific state information across the different purchases by the same customer to keep track of his (encrypted or committed) account balance. Different transactions by the same customer thereby necessarily become linkable. Thus, none of these protocols allows the customer to purchase records

anonymously: even if an anonymous payment system is used to pre-charge the initial balance, the customer could be at most pseudonymous, partially defeating the purpose of protecting the customer’s privacy. For example, the database still learns the number of records bought by each customer, the time that these records were bought, and their average price. This clearly reveals information about the customer and might lead to identification of the customer or of the records she’s buying. To overcome this, we further require that the POT additionally guarantees that (4) the database does not learn any information about who purchases a record.

Existing POT protocols also lack a recharge functionality: once a customer’s balance does not contain enough credit to buy a record, but is still positive, the customer cannot use up the balance, but will have to open a new account for further purchases. Even if the protocol can be extended so that the customer can reveal and reclaim any remaining credit, he will leak information about his purchases by doing so.

In this paper, we propose the first truly anonymous priced oblivious transfer protocol with recharge functionality. Rather than having the database keep track of account balances, in our protocol the customers maintain their own balance. Of course, precautions are taken to ensure that they cannot tamper with their balance, or rewind it to a previous state. Furthermore, we offer a protocol that allows customers to recharge their balances. Lastly, we present an enhanced protocol where records are transferred using an optimistic fair exchange protocol [2,3], thereby preventing a cheating database from decreasing a customer’s wallet without sending the desired record.

1.1 Construction Overview

We consider a database where each record may have a different price. The database provider encrypts each record with a key that is derived from not only its index but also from its price. It then publishes the entire encrypted database.

To be able to access records, a customer first contacts the provider to create a new, empty wallet. Customers can load more money into their wallet at any time. The payment mechanism used to recharge customers’ wallets is outside the scope of this paper; for full customer anonymity, we advise the use of an anonymous e-cash scheme.

When a customer wants to purchase a record with index σ and price p from the database, the provider and the customer essentially run a two-party protocol, at the end of which the customer will have obtained the decryption key for the record σ as well as an updated wallet with a balance of p units less. This is done in such a way that the provider does not learn anything about σ or p . More precisely, we model wallets as one-time-use anonymous credentials with the balance of the wallet being encoded as an attribute. When the customer buys a record (or recharges her wallet), she basically uses the credential and gets in exchange a new credential with the updated balance as an attribute, without the provider learning anything about the wallet’s balance. The properties of one-time-use credentials ensure that a customer cannot buy records worth more than what she has (pre-)paid to the provider. We prove our protocol secure in the standard model (i.e., without random oracles).

1.2 Related Work

Relative to the enormous body of work that has appeared on oblivious transfer, only few priced oblivious transfer protocols have been proposed. The concept of POT was introduced by Aiello et al. [1] who present a scheme based on homomorphic encryption and symmetrically private information retrieval (SPIR) [24]. The protocol by Tobias [27] is based on ElGamal, and a recent protocol by Rial et al. [29] builds on the OT protocol of [15]. The protocols of [1,27] come only with heuristic security considerations, while that of [29] was proved secure in the universal composability (UC) model [19]. All three of these protocols share a common principle that the database maintains an encryption or commitment of each customer’s balance that gets updated at each purchase. Purchases by the same customer are therefore necessarily linkable, as the database has to know which ciphertext or commitment to use. Neither of these protocols enables customers to recharge their wallets.

While by itself not being a POT protocol, the recent work by Coull et al. [21] could be cast into one. They propose an OT scheme where access to records is controlled by using a state graph. With each access a user transitions from one state to another, where the allowed records are defined by the possible transitions from the current state. One could implement a (fully anonymous) POT protocol by defining a separate state for each possible balance in a customer’s wallet. The allowed transitions between states b and b' are those records with price exactly $b - b'$. When using this approach however, the size of the encrypted database is $O(b_{\max} \cdot N)$, where b_{\max} is the maximum wallet balance and N is the number of records in the database, as opposed to $O(b_{\max} + N)$ in our scheme.

Our paper builds on ideas from recent work by Camenisch et al. [10] who extend the OT protocol of [15] with anonymous access control. One could in fact combine their and our ideas to achieve POT with access control. This would allow for price differentiation among customers while maintaining full customer privacy, e.g., offer a cheaper price to holders of a loyalty card, without leaking whether the customer has one.

2 Definition of UP-OT

2.1 Syntax

Let $\kappa \in \mathbb{N}$ be a security parameter and let ε be the empty string. All algorithms described here are probabilistic polynomial-time (PPT); we implicitly assume they all take an extra input 1^κ . A function $\nu : \mathbb{N} \rightarrow [0, 1]$ is *negligible* if for all $c \in \mathbb{N}$ there exists a $\kappa_c \in \mathbb{N}$ such that $\nu(\kappa) < \kappa^{-c}$ for all $\kappa > \kappa_c$.

An unlinkable priced oblivious transfer (UP-OT) scheme is parameterized by a security parameter $\kappa \in \mathbb{N}$, a maximum wallet balance $b_{\max} \in \mathbb{N}$ and a maximum record price $p_{\max} \leq b_{\max}$. We consider a setting with one database and one or more customers. A database consists of a list of N couples $((R_1, p_1), \dots, (R_N, p_N))$, containing database records $R_1, \dots, R_N \in \{0, 1\}^*$ and associated prices $p_1, \dots, p_N \in \{0, \dots, p_{\max}\}$. A UP-OT scheme is a tuple of polynomial-time algorithms and protocols $\mathcal{UP\text{-}OT} = (\text{DBSetup}, \text{CreateWallet}, \text{Recharge}, \text{Purchase})$ run between customers C_1, \dots, C_M and a database provider DB in the following way:

- **DBSetup** : $\text{DB} : (DB = (R_i, p_i)_{i=1, \dots, N}) \xrightarrow{\$} ((pk_{\text{DB}}, ER_1, \dots, ER_N), sk_{\text{DB}})$
 The database provider executes the randomized DBSetup algorithm to initiate a database containing records R_1, \dots, R_N with corresponding prices p_1, \dots, p_N . It generates a pair of a secret and corresponding public key $(sk_{\text{DB}}, pk_{\text{DB}})$ for security parameter κ , and uses it to encrypt the individual records. The encrypted database consists of the public key pk_{DB} and the encrypted records ER_1, \dots, ER_N . The encrypted database is made available to all customers, e.g., by publishing it on a website or by distributing it on DVDs.¹ The database provider keeps the secret key sk_{DB} to himself.
- **CreateWallet** : $\text{DB} : (pk_{\text{DB}}, sk_{\text{DB}}) \rightarrow (\varepsilon); C : (pk_{\text{DB}}) \rightarrow W_0 \text{ or } \perp$
 A customer creates an empty wallet with a zero balance, signed by the database provider, by engaging in the CreateWallet protocol with the database provider. The provider's public key pk_{DB} is a common input, the corresponding secret key sk_{DB} is a secret input to the provider. At the end of the protocol, the customer outputs an empty wallet W_0 , or \perp to indicate failure.
- **Recharge** : $\text{DB} : (pk_{\text{DB}}, m, sk_{\text{DB}}) \rightarrow (\varepsilon); C : (pk_{\text{DB}}, m, W_i) \rightarrow W_{i+1} \text{ or } \perp$
 When the customer wants to add money to her wallet W_i (which may or may not be her initial wallet W_0) she can engage in a Recharge protocol with the database provider. The database's public key pk_{DB} and the amount of money m that the customer wants to add to her balance are common inputs. The database's secret key sk_{DB} and the customer's current wallet W_i are private inputs to the database and the customer, respectively. Eventually the customer outputs the new wallet W_{i+1} or \perp to indicate failure.
- **Purchase** : $\text{DB} : (pk_{\text{DB}}, sk_{\text{DB}}) \rightarrow (\varepsilon); C : (pk_{\text{DB}}, \sigma, ER_\sigma, p_\sigma, W_i) \rightarrow (R_\sigma, W_{i+1}) \text{ or } (\perp, W_{i+1}) \text{ or } (R_\sigma, \perp) \text{ or } (\perp, \perp)$
 To purchase a record from the database, a customer engages in a Purchase protocol with the database provider. The database's public key pk_{DB} is a common input. The customer has as a private input her selection index $\sigma \in \{1, \dots, N\}$, the encrypted record ER_σ and its price p_σ , and her current wallet W_i . The database provider uses its secret key sk_{DB} as a private input. At the end of the protocol, the customer outputs the database record R_σ and an updated wallet W_{i+1} . An output containing $R_\sigma = \perp$ or $W_{i+1} = \perp$ indicates that the record transfer or the wallet update failed, respectively.

We assume that all communication links are private and anonymous, so that cheating customers cannot eavesdrop on honest customers' conversations, and so that the database does not know which customer he's interacting with.

2.2 Security

We define security of an UP-OT protocol through indistinguishability of a real-world and an ideal-world experiment reminiscent of the universal-composability (UC)

¹ We assume that each customer obtains a copy of the *entire* encrypted database. It is impossible to obtain our strong privacy requirements with a single database server without running into either computation or communication complexity that is linear in the database size. In this paper we focus on *amortizing* the complexity of the purchase protocol to keep it constant, i.e., independent of N .

framework [19] and the reactive-systems security model [25,28]. The definitions we give, however, do not entail all formalities necessary to fit either of these frameworks; our goal here is solely to prove security of our scheme.

We summarize the ideas underlying these models. In the real world the honest players and the adversary A who controls the dishonest players run cryptographic protocols with each other. The environment \mathcal{E} provides the inputs to the honest players and receives their outputs, and interacts arbitrarily with the adversary. In the ideal world, the players do not run any cryptographic protocols but interact through an ideal trusted party T . A (set of) cryptographic protocol(s) is said to securely implement a functionality if for every real-world adversary A and every environment \mathcal{E} there exists an ideal-world simulator A' (controlling the same parties as A) such that \mathcal{E} cannot distinguish with non-negligible probability whether it is run in the real world while interacting with A or whether it is run in the ideal world while interacting with A' .

THE REAL WORLD. We first describe how the real world algorithms presented in §2.1 are orchestrated when all participants are honest, i.e., honest real-world customers C_1, \dots, C_M and an honest database DB. Parties controlled by the real-world adversary A can arbitrarily deviate from the behavior described below.

Upon receiving $(\text{initdb}, DB = (R_i, p_i)_{i=1, \dots, N})$ from \mathcal{E} , the database generates a key pair and encrypts the records by running $((pk_{\text{DB}}, EDB), sk_{\text{DB}}) \xleftarrow{\$} \text{DBSetup}(DB)$, and sends $(pk_{\text{DB}}, EDB, (p_i)_{i=1, \dots, N})$ to all customers C_1, \dots, C_M .

Upon receiving (create_wallet) from \mathcal{E} , customer C_j obtains an empty wallet by engaging in a CreateWallet protocol with the database provider on common input pk_{DB} . The provider uses his secret key sk_{DB} as a private input. At the end of the protocol, the customer obtains the empty wallet $W_0^{(j)}$ with zero balance or \perp indicating failure. In the former case C_j returns a bit 1 to \mathcal{E} , in the latter it outputs 0. DB does not return anything to the environment.

Upon receiving $(\text{recharge}, m)$ from \mathcal{E} customer C_j engages in a Recharge protocol with DB on common input pk_{DB}, m , using sk_{DB} and $W_i^{(j)}$ as private inputs to DB and C_j , respectively, where $W_i^{(j)}$ is C_j 's current wallet. At the end of the protocol, C_j either obtains the new wallet $W_{i+1}^{(j)}$ or \perp . In the former case, it returns a bit 1 to \mathcal{E} , in the latter it outputs 0. DB does not return anything to the environment.

Upon receiving $(\text{purchase}, \sigma)$ from \mathcal{E} , customer C_j engages in a Purchase protocol with DB on common input (pk_{DB}) , on C_j 's private input $\sigma, ER_\sigma, p_\sigma, W_i^{(j)}$, and on DB's private input sk_{DB} , until C_j obtains the record R_σ and a new wallet $W_{i+1}^{(j)}$. The customer returns two bits to the environment, the first indicating whether the record transfer succeeded (i.e., 0 if $R_\sigma = \perp$ and 1 otherwise), the second indicating whether the wallet update succeeded (i.e., 0 if $W_{i+1}^{(j)} = \perp$ and 1 otherwise). DB does not return anything to the environment.

THE IDEAL WORLD. In the ideal world all participants communicate through a trusted party T which implements the functionality of our protocol. We describe the behavior of T on the inputs of the ideal-world customers C'_1, \dots, C'_M and the ideal-world database DB'. The trusted party T maintains the database DB and an array $W[\cdot]$ to keep track of the balance in customer's wallets. Initially all entries are unspecified, i.e., $DB \leftarrow \varepsilon$ and

$W[j] \leftarrow \varepsilon$ for $j = 1, \dots, N$. The trusted party responds to queries from the different parties as follows.

Upon receiving $(\text{initdb}, (R_i, p_i)_{i=1, \dots, N})$ from DB' , T checks whether $0 \leq p_i \leq p_{\max}$ for $i = 1, \dots, N$. If so, it sets $DB \leftarrow (R_i, p_i)_{i=1, \dots, N}$ and sends a message $(\text{initdb}, \{p_i\}_{i=1, \dots, N})$ to all customers.

Upon receiving (create_wallet) from C'_j , T sends (create_wallet) to DB' who sends back a bit b . If $b = 1$ then T sets $W[j] \leftarrow 0$ and sends 1 to C'_j ; otherwise it simply sends 0 to C'_j .

Upon receiving $(\text{recharge}, m)$ from C'_j , T first checks that $W[j] \neq \varepsilon$ and $W[j] + m \leq b_{\max}$. If either of these checks fails, it sends back a bit 0 to C'_j , otherwise it proceeds as follows. T sends $(\text{recharge}, m)$ to DB' who sends back a bit b . If $b = 1$ then the T sets $W[j] \leftarrow W[j] + m$ and sends a bit 1 to C'_j ; otherwise it simply sends 0 to C'_j .

Upon receiving $(\text{purchase}, \sigma)$ from C'_j , T proceeds as follows. If $W[j] < p_\sigma$ then T simply returns a pair $(\perp, 1)$ to C'_j . Otherwise, it sends a message (purchase) to DB' , who sends back a pair of bits (b_1, b_2) indicating whether or not the record transfer and the wallet update succeeded. Party T sends a pair (R, b) back to C'_j that is composed as follows. If $b_1 = 1$ and $DB \neq \varepsilon$ then it sets $R \leftarrow R_\sigma$, otherwise it sets $R \leftarrow \perp$. If $b_2 = 1$ then T sets $W[j] \leftarrow W[j] - p_\sigma$ and $b \leftarrow 1$; else it sets $W[j] \leftarrow \varepsilon$ and $b \leftarrow 0$.

The honest ideal-world parties C'_1, \dots, C'_M, DB simply relay inputs and outputs between the environment \mathcal{E} and the trusted party T . Dishonest parties can deviate arbitrarily from this behavior.

Note that in the ideal world the database cannot tell which customer makes a purchase, which record she is querying, or what the price of this record is, therefore guaranteeing perfect customer privacy. At the same time, customers in the ideal world can only purchase records that they can afford, they can only obtain one record per purchase, and even colluding customers cannot obtain records that they wouldn't have been able to afford individually, thereby guaranteeing perfect database security.

3 Preliminaries

Let $\text{Pg}(1^\kappa)$ be a pairing group generator that on input 1^κ outputs descriptions of multiplicative groups \mathbb{G}, \mathbb{G}_T of prime order q where $q > 2^\kappa$. Let $\mathbb{G}^* = \mathbb{G} \setminus \{1\}$ and let $g \in \mathbb{G}^*$. The generated groups are such that there exists an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, meaning that (1) for all $a, b \in \mathbb{Z}_q$ it holds that $e(g^a, g^b) = e(g, g)^{ab}$; (2) $e(g, g) \neq 1$; and (3) the bilinear map is efficiently computable.

Definition 1 We say that the decision ℓ -bilinear Diffie-Hellman exponent (ℓ -BDHE) assumption holds in groups \mathbb{G}, \mathbb{G}_T of order $q > 2^\kappa$ if for all polynomial-time adversaries A the advantage $\mathbf{Adv}_{\mathbb{G}, \mathbb{G}_T}^{\text{BDHE}}(\kappa)$ of, given a tuple $(g, h, g^\alpha, \dots, g^{\alpha^{\ell-1}}, g^{\alpha^{\ell+1}}, \dots, g^{\alpha^{2\ell}}, S)$, to distinguish whether $S = e(g, h)^{\alpha^\ell}$ or $S \stackrel{\$}{\leftarrow} \mathbb{G}_T^*$, is a negligible function in κ for $g, h \stackrel{\$}{\leftarrow} \mathbb{G}^*$ and $\alpha \stackrel{\$}{\leftarrow} \mathbb{Z}_q$. ■

Definition 2 We say that the ℓ -strong Diffie-Hellman (ℓ -SDH) assumption [6] holds in group \mathbb{G} of order $q > 2^\kappa$ if for all polynomial-time adversaries A the advantage is a negligible function in κ , where $g \stackrel{\$}{\leftarrow} \mathbb{G}^*$ and $x \stackrel{\$}{\leftarrow} \mathbb{Z}_q$. ■

3.1 Modified Boneh-Boyen Signatures

We use the following modification of the weakly-secure signature scheme by Boneh and Boyen [6]. The scheme uses a pairing generator Pg as defined above.

The signer's secret key is $(x_m, x_p) \xleftarrow{\$} \mathbb{Z}_q$, the corresponding public key is $(g, y_m = g^{x_m}, y_p = g^{x_p})$ where g is a random generator of \mathbb{G} . The signature on the tuple of messages (m, p) is $s \leftarrow g^{1/(x_m+m+x_pp)}$; verification is done by checking whether $e(s, y_m \cdot g^m \cdot y_p^p) = e(g, g)$ is true.

This signature scheme is the special case for $\ell = 2$ of the modified Boneh-Boyen signature scheme used by Camenisch et al. [10], who also show it to be unforgeable under weak chosen-message attack if the $(N + 1)$ -SDH assumption holds, where N is the number of signing queries.

3.2 Zero-Knowledge Proofs and Σ -Protocols

When referring to the zero-knowledge proofs, we will follow the notation introduced by Camenisch and Stadler [17] and formally defined by Camenisch, Kiayias, and Yung [12]. For instance, $\text{PK}\{(a, b, c) : y = g^a h^b \wedge \tilde{y} = \tilde{g}^a \tilde{h}^c\}$ denotes a “zero-knowledge Proof of Knowledge of integers a, b, c such that $y = g^a h^b$ and $\tilde{y} = \tilde{g}^a \tilde{h}^c$ holds,” where $y, g, h, \tilde{y}, \tilde{g}$, and \tilde{h} are elements of some groups $G = (g) = (h)$ and $\tilde{G} = (\tilde{g}) = (\tilde{h})$.

Given a protocol in this notation, it is straightforward to derive the actual protocol implementing the proof. The computational complexities of the proof protocol are also easily derived from this notation: for each term $y = g^a h^b$, the prover and the verifier have to perform an equivalent computation, and to transmit one group element and one response value for each exponent. We refer to Camenisch et al. [12] for details.

3.3 Wallet Signature Scheme

We use the signature scheme proposed and proved secure by Au et al. [4], which is based on the schemes of Camenisch and Lysyanskaya [13] and of Boneh et al. [7].

The signer's secret key is a random element $x \xleftarrow{\$} \mathbb{Z}_q$. The public key contains a number of random bases $g_1, h_0, \dots, h_\ell, h_{\ell+1} \xleftarrow{\$} \mathbb{G}$, where $\ell \in \mathbb{N}$, and $y \leftarrow g_1^x$. A signature on messages $m_0, \dots, m_\ell \in \mathbb{Z}_q$ is a tuple (A, r, s) where $r, s \xleftarrow{\$} \mathbb{Z}_q$ are values chosen at random by the signer and $A = (g_1 h_0^{m_0} \cdots h_\ell^{m_\ell} h_{\ell+1}^r)^{1/(x+s)}$. Such a signature can be verified by checking whether $e(A, g_1^s y) = e(g_1 h_0^{m_0} \cdots h_\ell^{m_\ell} h_{\ell+1}^r, g_1)$.

Now assume that we are given a signature (A, r, s) on messages $m_0, \dots, m_\ell \in \mathbb{Z}_q$ and want to prove that we indeed possess such a signature. This can be done by augmenting the public key with values $u, v \in \mathbb{G}$ such that $\log_{g_1} u$ and $\log_{g_1} v$ are not known, choosing random values $t, t' \xleftarrow{\$} \mathbb{Z}_q$, computing $\tilde{A} = Au^t$, $B = v^t u^{t'}$, and executing the proof of knowledge

$$\begin{aligned} \text{PK}\{(\alpha, \beta, s, t, t', m_0, \dots, m_\ell, r) : B = v^t u^{t'} \wedge 1 = B^{-s} v^\alpha u^\beta \wedge \\ \frac{e(\tilde{A}, y)}{e(g_1, g_1)} = e(\tilde{A}^{-s} u^\alpha h_{\ell+1}^r \prod_{i=0}^\ell h_i^{m_i}, g_1) e(u, y)^t\} , \end{aligned}$$

where $\alpha = st$ and $\beta = st'$.

DBSetup($DB = (R_i, p_i)_{i=1,\dots,N}$) :

If $b_{\max} > 2^{\kappa-1}$ or $\exists i : p_i > p_{\max}$ then abort

$(\mathbb{G}, \mathbb{G}_T, q) \xleftarrow{\$} \text{Pg}(1^\kappa)$; $g_t, h_t \xleftarrow{\$} \mathbb{G}_T^*$; $g, h, g_1, h_1, h_2, h_3 \xleftarrow{\$} \mathbb{G}^*$; $x_R, x_p, x_b, x_w \xleftarrow{\$} \mathbb{Z}_q$
 $H \leftarrow e(g, h)$; $y_R \leftarrow g^{x_R}$; $y_p \leftarrow g^{x_p}$; $y_b \leftarrow g^{x_b}$; $y_w \leftarrow g_1^{x_w}$

For $i = 1, \dots, N$ do $E_i \leftarrow g^{\overline{x_R+i+x_p+p_i}}$; $F_i \leftarrow e(h, E_i) \cdot R_i$; $ER_i \leftarrow (E_i, F_i)$

For $i = 0, \dots, b_{\max}$ do $y_b^{(i)} \leftarrow g^{1/(x_b+i)}$

$sk_{\text{DB}} \leftarrow (h, x_R, x_p, x_b, x_w)$; $pk_{\text{DB}} \leftarrow (g, H, g_1, h_1, h_2, h_3, y_R, y_p, y_b, y_b^{(0)}, \dots, y_b^{(b_{\max})}, y_w)$
Return $((pk_{\text{DB}}, ER_1, \dots, ER_N), sk_{\text{DB}})$

Fig. 1. Database setup algorithm

It was proved in [4] that the above signature is unforgeable under adaptively chosen message attack if the Q -SDH assumption holds, where Q is the number of signature queries, and that the associated PoK is perfect honest-verifier zero-knowledge.

3.4 Set Membership Scheme

To prove that the customer's new balance after buying a record remains positive and is not more than the maximum balance we use a signature-based set membership protocol suggested by Camenisch, Chaabouni and shelat [9].

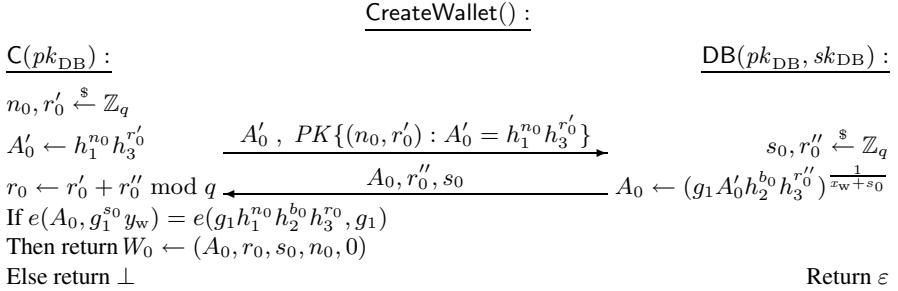
They consider a zero-knowledge protocol which allows a prover to convince a verifier that a digitally committed value is an element of a given public set. The verifier signs the individual elements and sends the signatures to the prover. The prover shows that he knows a valid signature (by the verifier) on the element that he holds. The scheme of [9] employs the weak signature scheme by Boneh and Boyen [6]. They prove that their protocol is a zero-knowledge argument of set membership for a set Φ , if the $|\Phi|$ -SDH assumption holds.

4 Our UP-OT Construction

We now describe our scheme in detail. To issue wallets and update customers' balances, we employ the signature scheme presented in Section 3.3. To implement the oblivious transfer with anonymous payments we extend the OT protocol by Camenisch et al. [15]. We will also use a number of zero-knowledge proofs about discrete logarithms as described in Section 3.2.

Initial Setup. In Figure 1 we describe the setup procedure of the database provider, who also issues wallets to customers. Customers do not have their own setup procedure.

The database provider runs the randomized DBSetup algorithm to initiate a database containing records R_1, \dots, R_N with corresponding prices p_1, \dots, p_N . It generates a pairing group of prime order q for security parameter κ , a number of random generators, and four secret keys x_R, x_p, x_b , and x_w with corresponding public keys y_R, y_p, y_b , and y_w . Intuitively, x_R is used as a randomness seed to encrypt the records, x_p securely links prices to records, x_b authenticates all possible balances, and x_w authenticates the balance in customers' wallets.

**Fig. 2.** Create wallet protocol

The database provider encrypts each record R_i with its own key to a ciphertext (E_i, F_i) . These keys are in fact signatures on the index i and the price p_i of the record under the database provider's secret keys x_R and x_p . The pairs (E_i, F_i) can be seen as an ElGamal encryption [23] in \mathbb{G}_T of the record R_i under the public key H . But instead of using random elements from \mathbb{G}_T as the first component, our protocol uses verifiably random [22] values $E_i = g^{\frac{1}{x_R+i+x_p \cdot p_i}}$. It is this verifiability that during the purchase protocol allows the database to check that the customer is indeed asking for the decryption key of a single record with a price that is within his current balance.

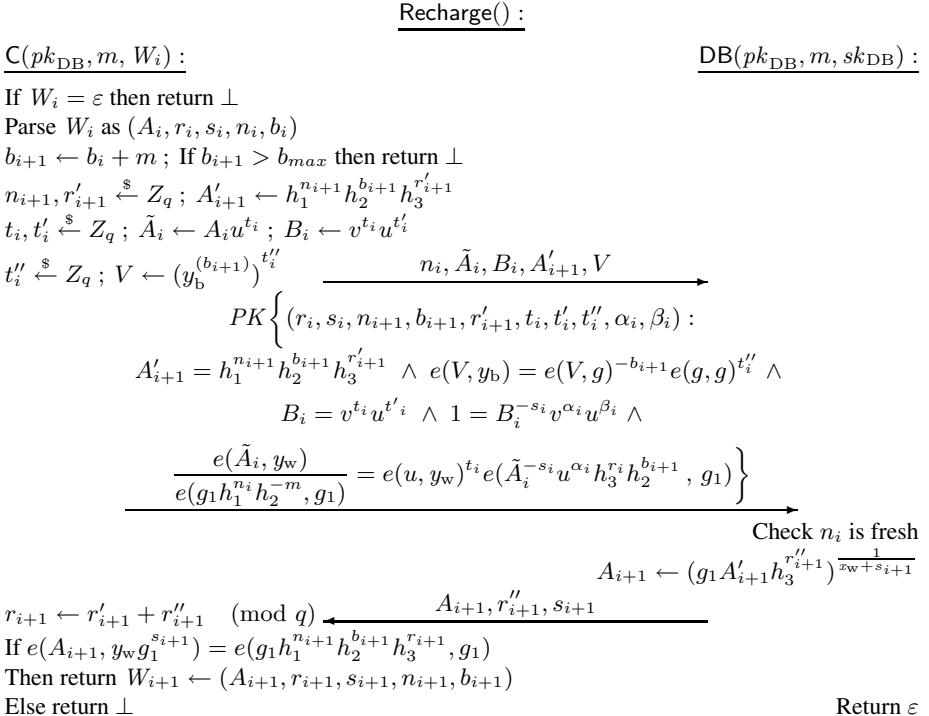
Let $p_{max} \leq b_{max} < 2^{\kappa-1} < q/2$ be the maximal balance that can be stored in a customer's wallet. To prove that the customer's new balance after buying a record remains positive and is not more than the maximum balance, we use a signature-based set membership protocol of Section 3.4. Here the set contains all possible balances from the customer's wallet $\{0, \dots, b_{max}\}$. So for each possible balance $0 \leq i \leq b_{max}$ the database provider uses x_b to compute a signature $\{y_b^{(i)}\}$. These values are included in the database's public key; they will be used by the customer to prove that her balance remains positive after subtracting the price of the purchased record.

The encrypted database consists of a public key pk_{DB} and the encrypted records ER_1, \dots, ER_N . It is made available to all customers, e.g., by publishing it on a website or by distributing it on DVDs. The server keeps the database secret key sk_{DB} to itself.

Obtaining wallets. Before purchasing any records, customers first need to create an empty wallet and then charge it with money. To create a wallet, the customer runs the CreateWallet protocol with the database provider depicted in Figure 2.

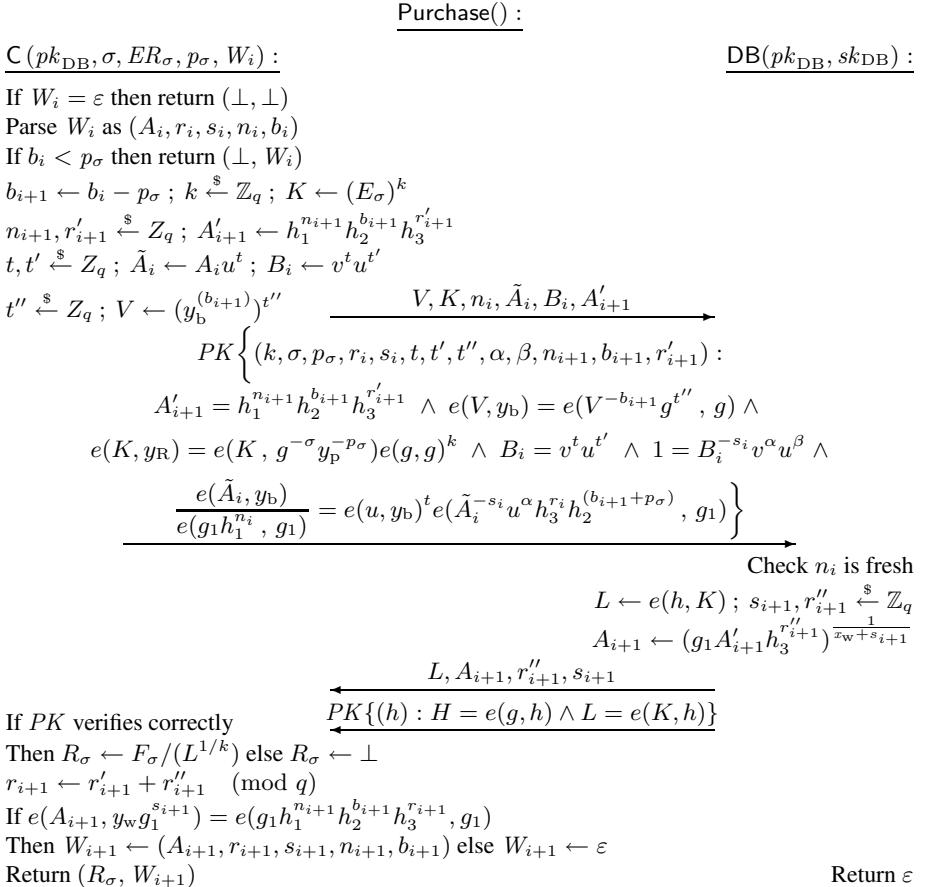
The database provider's public key pk_{DB} is a common input. The database provider has his secret key sk_{DB} as a private input. At the end of the protocol, the customer obtains a wallet $W_0 = (A_0, r_0, s_0, n_0, b_0 = 0)$ signed by the database provider. Here, (A_0, r_0, s_0) is essentially a signature as per the scheme of Section 3.3 of a serial number n_0 chosen by the customer and the initial balance of the wallet $b_0 = 0$. Next, the customer verifies the wallet's signature and outputs W_0 if the check is successful.

Recharge protocol. Customers can recharge the balance of their wallets by engaging in a Recharge protocol (Figure 3) with the database server. Doing so does not reveal

**Fig. 3.** Recharge protocol

the remaining balance in the wallet, nor whether this is a freshly created wallet or an updated wallet obtained after purchasing a record. The common inputs are the database provider's public key pk_{DB} and the amount of money m that the customer wants to add to her balance. The database's secret key sk_{DB} and the customer's current wallet W_i are private inputs to the database and the customer, respectively.

If the customer already obtained a wallet earlier (her state is not empty), she updates her balance to $b_{i+1} = b_i + m$ and generates a fresh serial number n_{i+1} and a randomizer r'_{i+1} for the new wallet. Then she chooses from the set of database signatures $y_b^{(0)}, \dots, y_b^{(b_{max})}$ of possible balances the signature corresponding to her new balance and blinds it as $V = (y_b^{(b_{i+1})})^{t''_i}$. This allows her to next prove that her new balance b_{i+1} is positive and less than b_{max} with the set membership scheme from [9]. The customer further proves that she correctly increased her balance by the amount m being deposited. The database provider checks whether the proof is valid and whether the serial number n_i is fresh, i.e., whether it previously saw the number n_i . If not, then the database decides that the customer is trying to overspend and aborts. Otherwise, if the database provider accepts the proof, it signs the customer's new wallet with updated balance and sends it to the customer. The customer checks the validity of the signature on her new wallet, and if it verifies correctly, outputs an updated state containing the new wallet W_{i+1} .

**Fig. 4.** Purchase protocol

Purchase protocol. When the customer wants to purchase a record from the database, she engages in a Purchase protocol (Figure 4) with the database server. The only common input is the database's public key pk_{DB} . The customer has as a private input her selection index $\sigma \in \{1, \dots, N\}$, the encrypted record ER_σ and its price p_σ , and her current wallet W_i . The database provider uses its secret key sk_{DB} as a private input.

The customer blinds the first part of the chosen encrypted record E_{σ_i} and sends this blinded version K to the database. Note that E_{σ_i} is derived from the database provider's secret key, the index and the price of the record. Next, the customer updates her balance to $b_{i+1} = b_i - p_{\sigma_i}$, generates a fresh serial number n_{i+1} and a randomizer for the new wallet. Then she chooses from the set of database signatures $y_b^{(0)}, \dots, y_b^{(b_{\max})}$ of possible balances the signature corresponding to her new balance and blinds it as $V = (y_b^{(b_{i+1})})^{t''}$. This allows her to prove that her new balance b_{i+1} is positive employing the set membership scheme from [9]. The customer further proves that K is correctly formed as a blinding of some E_{σ_i} and that she correctly reduced her balance by the

price of requested record. The database provider checks if the serial number n_i is fresh, i.e., whether it previously saw the number n_i . If not, then the database decides that the customer is trying to double-spend and aborts. Otherwise, if the database provider accepts the proof, it computes L from h and K , sends L to the customer, and executes a proof of knowledge of the database secret key h and that L was computed correctly. In our security proof, this zero-knowledge proof will enable us to extract h and decrypt the contents of the database. Also database provider signs the customer's new wallet with updated balance and sends it to the customer. The customer checks the validity of the zero-knowledge proof and of the signature on her new wallet. \perp as the record; if the wallet signature is invalid, then it returns ε as the new wallet; if all goes correctly, she outputs the record R_σ and the new wallet W_{i+1} . The protocol is easily seen to be correct by observing that $L = e(h, E_{\sigma_i})^k$, so therefore $F_\sigma / L^{1/k} = R_{\sigma_i}$.

Notice that $PK\{(h) : H = e(g, h) \wedge L = e(K, h)\}$ can be realized in the standard way as $e(g, \cdot)$ is a group (one-way) homomorphism that maps \mathbb{G} onto \mathbb{G}_T .

We finally remark that the database has to calculate a signature of every element in the set of all possible balances in the customer's wallet $\{0, \dots, b_{\max}\}$ and encrypt all records $(1, \dots, N)$ only once at the setup phase, and the customer has to download the entire encrypted database and balance signatures only once as well. So the communication and computation complexity of the protocol depends on a cardinality of a set of all possible balances in the customer's wallet and a number of the records in the database only at the setup phase. The other parts of the protocol (create wallet, recharge and purchase), however, require only a constant number of group elements to be sent and a constant number of exponentiations and pairings to be computed.

5 Security Analysis

The security of our protocol is analyzed by proving indistinguishability between adversary actions in the real protocol and in an ideal scenario that is secure by definition. Given a real-world adversary A , we construct an ideal-world adversary A' such that no environment \mathcal{E} can distinguish whether it is interacting with A or A' .

Theorem 3 *If the $(N + 2)$ -BDHE assumption holds in \mathbb{G}, \mathbb{G}_T and the $\max(q_W, b_{\max} + 1, N + 1)$ -SDH assumption holds in \mathbb{G} , then the UP-OT protocol depicted in Figures 1–4 securely implements the UP-OT functionality, where N is the number of database records, b_{\max} is the maximum possible balance in a customer's wallet, and q_W is the number of created wallets.* ■

We prove the theorem in two steps: first, we prove the case where the database is corrupted, and next, we prove the case where a subset of the customers are corrupted. We do not consider the cases where all parties are honest and where all parties are dishonest as these cases have no practical interest. By lack of space, we only provide sketches of the construction of the ideal-world adversary A' below. A detailed proof is available in the full version of this paper [11].

Corrupted database. We first prove that for all environments \mathcal{E} and all real-world adversaries A controlling the database there exists an ideal-world adversary A' such that \mathcal{E} can distinguish the real world from the ideal world with probability at most $2^{-\kappa}$.

Since the adversary can always simulate additional customers himself, we can simplify the setting to a single honest customer C . We construct an ideal-world adversary A' that plays the role of the database and that runs the real-world adversary A as a subroutine as follows.

A' simply relays all messages between the environment \mathcal{E} and A . It runs A to obtain the database's public key pk_{DB} and the encrypted database $EDB = (pk_{\text{DB}}, (E_1, F_1), \dots, (E_N, F_N))$.

Upon receiving a message (create_wallet) from T , it executes the customer's side of the CreateWallet protocol with A . If the obtained wallet is valid, A' returns $b = 1$ to T , otherwise it returns $b = 0$.

Upon receiving $(\text{recharge}, m)$ from T , A' executes the customer's side of the Recharge protocol for amount m with A , but replaces the value V with a random element from \mathbb{G} and simulates the PK protocol. If the protocol returns a valid wallet, then A' returns $b = 1$ to T ; if the protocol returns \perp then A' returns $b = 0$.

At the first purchase message from T , A' simulates an honest user querying for R_1 , but replaces V with a random value from \mathbb{G} and simulates the proof of knowledge. Then it extracts h from A in the last proof of knowledge, uses it to decrypt R_i as $F_i/e(h, E_i)$ for $i = 1, \dots, N$ and sends $(\text{initdb}, (R_i, p_i)_{i=1, \dots, N})$ to T . A' sends a pair of bits (b_1, b_2) back to T depending whether the obtained record is valid ($b_1 = 1$) or not ($b_1 = 0$) and whether the updated wallet is valid ($b_2 = 1$) or not ($b_2 = 0$).

Corrupted customers. Next, we prove that for all environments \mathcal{E} and all real-world adversaries A controlling some of the customers, there exists an ideal-world adversary A' such that \mathcal{E} can distinguish the real from the ideal world with probability at most

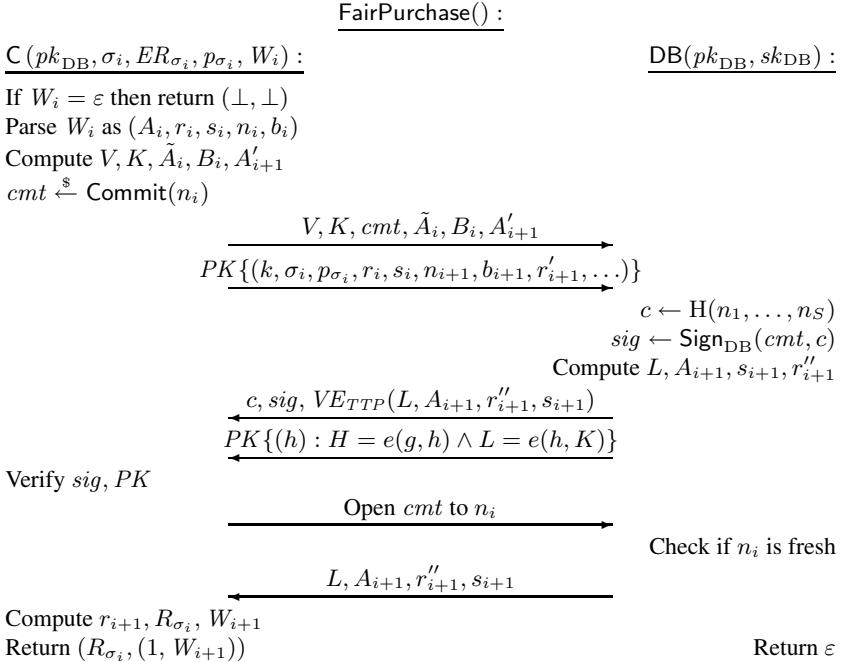
$$2^{-\kappa} \cdot Q + \mathbf{Adv}_{\mathbb{G}}^{q\text{-SDH}}(\kappa) + (N+1) \cdot \mathbf{Adv}_{\mathbb{G}, \mathbb{G}_T}^{(N+2)\text{-BDHE}}(\kappa),$$

where Q is the total number of create wallet, recharge, and purchase queries, $q = \max(q_{\text{W}}, b_{\text{max}} + 1, N + 1)$, q_{W} is the total number of create wallet queries, b_{max} is the maximum possible balance in a customer's wallet, and N is the number of records in the database.

Since the UP-OT functionality prevents colluding customers from pooling their wallets we have to consider multiple customers here, some of which are corrupted, and some of which are honest. Given a real-world adversary A controlling some of the customers, we construct an ideal-world adversary A' controlling the same customers as follows.

A' simply relays all communication between the environment \mathcal{E} and A . Upon receiving $(\text{initdb}, \{p_i\}_{i=1, \dots, N})$ from T , A' creates an encrypted database (EDB, sk_{DB}) encrypting N random records, meaning that F_1, \dots, F_N are random group elements of \mathbb{G}_T .

To simulate A 's CreateWallet, Recharge, and Purchase protocol queries, A' plays the role of a real-world database by extracting from A the wallet signatures (A_i, r_i, s_i) , record signatures E_{σ_i} and range proof signatures $y_b^{(i)}$ that it uses in the zero-knowledge proofs, and aborting if either a forged wallet signature or a forged range proof signature is detected. When a cheating customer C_j (controlled by A) makes a CreateWallet query, A' sends a message $(\text{create_wallet}, j)$ to T ; when C_j makes a Recharge query

**Fig. 5.** Fair purchase protocol

for amount m , A' sends a message $(\text{recharge}, j, m)$ to T . These never cause C'_j 's balance to exceed b_{\max} , because that would imply a forgery in the range proof signature, which we ruled out above.

When a cheating customer C_j makes a Purchase query, A' extracts the index of the record being purchased σ_i and the exponent k , and sends $(\text{purchase}, j, \sigma_i)$ to T to obtain the record R_{σ_i} . Note that C_j 's balance is always sufficient to purchase R_{σ_i} , because otherwise A would have forged one of the signature schemes, which we ruled out above. Next, A' computes $L \leftarrow (F_{\sigma_i}/R_{\sigma_i})^k$ and simulates the zero-knowledge proof as in the simulation above.

6 Fair Purchase and Recharge Protocols

In the recharge and purchase protocols of our basic scheme, the customer has to spend her current wallet (i.e., reveal the serial number n_i) before obtaining a new wallet and the decryption key to the purchased record. A malicious database could abuse this situation by aborting the transaction after the wallet was spent, thereby leaving the customer with a spent wallet and without a new wallet and the record that she wanted.

In this section we sketch how to strengthen our recharge and purchase protocols against this type of attacks by introducing a trusted third party (TTP) who's only involved in case of conflict. Essentially, we let the customer and the database engage in

an optimistic fair exchange [2,3] of the serial number n_i against the record decryption key L and updated wallet $(A_{i+1}, r''_{i+1}, s_{i+1})$ as shown on Figure 5.

Here the customer and the database proceed as in the simple purchase protocol, except that instead of sending n_i in the clear the customer sends a commitment $cmt = \text{Commit}(n_i)$ to the database, and performs the proof of knowledge $PK\{(n_{i+1}, s_i, \dots)\}$ based on this commitment. If the proof is accepted, the server sends back a verifiable encryption [16] of the decryption key and the new wallet $(L, A_{i+1}, r''_{i+1}, s_{i+1})$ under the trusted third party's public key, and performs the proof of knowledge $PK\{(h) : \dots\}$ based on this verifiable encryption. The database also computes the hash c of all previously revealed serial numbers and provides the customer with a signature on cmt, c . Only after receiving the signature and verifying the PK does the customer reveal the serial number n_i to the database. The database checks if it is fresh serial number, and if so opens $L, A_{i+1}, r''_{i+1}, s_{i+1}$ to the customer.

If the database tries to cheat by not sending the decryption key or the new wallet, then the customer can take n_i , the verifiable encryption and the database's signature to the TTP. The TTP will contact the database to ask for the list of serial numbers that were included in the computation of c and checks whether n_i appears in this list. If not, then the database decrypts $L, A_{i+1}, r''_{i+1}, s_{i+1}$ for the customer. If it does, the TTP decides that the customer's complaint was unjustified and does nothing.

If the customer tries to cheat by reusing an old wallet or not opening the commitment cmt , then the database simply doesn't reveal $L, A_{i+1}, r''_{i+1}, s_{i+1}$ to the customer.

Similarly to the FairPurchase protocol sketched above, one could also design a FairRecharge protocol to ensure that the customer obtains a recharged wallet after a payment is made. For this to work, the underlying payment system has to provide a mechanism by which the customer can prove to the TTP that the payment was made. Details are left to the full version [11].

Acknowledgements

This work was supported in part by the European Community through the Seventh Framework Programme (FP7/2007-2013) project PrimeLife (agreement no. 216483).

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 119. Springer, Heidelberg (2001)
2. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: ACM CCS 1997. ACM Press, New York (1997)
3. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. IEEE JSAC 18(4), 593–610 (2000)
4. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
5. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
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. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997)
9. 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)
10. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: ACM CCS 2009. ACM Press, New York (2009)
11. Camenisch, J., Dubovitskaya, M., Neven, G.: Unlinkable Priced Oblivious transfer with Rechargeable Wallets. In: Cryptology ePrint Archive (2010)
12. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized schnorr proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2010)
13. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
14. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number n is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 107. Springer, Heidelberg (1999)
15. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
16. 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)
17. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
18. Camenisch, J.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich, Diss. ETH No. 12520 (1998)
19. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001. IEEE Computer Society Press, Los Alamitos (2001)
20. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
21. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. In: Jareck, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 501–520. Springer, Heidelberg (2009)
22. 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)
23. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
24. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: ACM STOC. ACM Press, New York (1998)
25. Pfitzmann, B., Waidner, M.: Composition and integrity preservation of secure reactive systems. In: ACM CCS 2000. ACM Press, New York (2000)
26. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)
27. Tobias, C.: Practical oblivious transfer protocols. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 415–426. Springer, Heidelberg (2003)
28. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press, Los Alamitos (2001)
29. Rial, A., Kohlweiss, M., Preneel, B.: Universally composable adaptive priced oblivious transfer. In: Shacham, H. (ed.) Pairing 2009. LNCS, vol. 5671, pp. 231–247. Springer, Heidelberg (2009)

Multiple Denominations in E-cash with Compact Transaction Data^{*}

Sébastien Canard¹ and Aline Gouget²

¹ Orange Labs R&D, 42 rue des Coutures, F-14066 Caen, France

² Gemalto, 6, rue de la Verrerie, F-92190 Meudon, France

Abstract. We present a new construction of *divisible* e-cash that makes use of 1) a new generation method of the binary tree of keys; 2) a new way of using bounded accumulators. The transaction data sent to the merchant has a constant number of bits while spending a monetary value 2^ℓ . Moreover, the spending protocol does not require complex zero-knowledge proofs of knowledge such as proofs about double discrete logarithms. We then propose the first strongly anonymous scheme with standard unforgeability requirement and realistic generation parameters while improving the efficiency of the spending phase.

1 Introduction

In e-cash systems, users withdraw coins from a bank and use them to pay merchants (preferably without involving the bank during this protocol). Finally, merchants deposit coins to the bank. An e-cash system must prevent both a user from double-spending, and a merchant from depositing twice a coin. The anonymity of honest users should be protected whereas the identity of cheaters must be recovered preferably without using a trusted third party.

Divisible e-cash aims at improving the efficiency of both the withdrawal protocol and the spending of multiple denominations. The underlying idea is to efficiently withdraw a single divisible coin equivalent to 2^L unitary coins. The user can spend this coin by dividing its monetary value, e.g. by sub-coins of monetary value 2^ℓ , $0 \leq \ell \leq L$. In this paper, we revisit the divisible e-cash approach by targeting the most demanding security model while providing a realistic parameter generation algorithm and an efficient spending protocol.

1.1 Related Work

A generic construction of divisible e-cash schemes which fulfill the classical properties of anonymity and strongly unlinkability without using a trusted third party to revoke the identity of cheaters has been proposed in [9]. The wallet is represented by a binary tree such that each internal node corresponds to an amount, i.e. 2^{L-i} if the node's distance to the root is i , $0 \leq i \leq L$. Each node in the

^{*} This work has been financially supported by the French Agence Nationale de la Recherche and the TES Cluster under the PACE project.

tree is related to a *key* such that the key of a child can be computed from the key of one of its ascendants. The main efficiency bottleneck of the practical instantiation given in [9] is that the user has to prove during the spending phase the correctness from the tree root to the target node without revealing none of the $L - \ell$ intermediate values. As one node key is derived from its parents using a modular exponentiation, the user must prove, for each intermediate value and using proofs about double discrete logarithms, that they satisfy a certain relation. Each such proof is expensive and requires $\Omega(k)$ group elements to be communicated in order to guarantee $1/2^k$ soundness error. Moreover, the construction of the binary tree used in [9] (and previously used in [11]) is difficult to instantiate in practice [1]. Indeed, this construction necessitates to manage $L + 2$ groups G_0, G_1, \dots, G_{L+1} with prime order p_0, p_1, \dots, p_{L+1} respectively, such that for all $1 \leq i \leq L+1$, G_{i+1} is a subgroup of $\mathbb{Z}_{p_i}^*$. One possibility is to take $p_i = 2 \times p_{i-1} + 1$ for all $1 \leq i \leq L+1$. Using prime number theory, it is possible to show that the probability to generate such prime numbers is approximately 2^{-95} for 1024 bits prime numbers and $L = 10$, which is unpractical.

A very efficient variant of this scheme based on bounded accumulators has been proposed in [1]. Its main drawback is that it does not fulfill the classical security property of unforgeability. Indeed, it is possible for a malicious user to withdraw a divisible coin of monetary value $L2^L$ whereas the legitimate value is 2^L by cheating in the construction of the binary tree of keys during the withdrawal protocol. Next, the user can spend $L2^L$ coins without being detected and identified. The solution proposed by the authors is that the bank will use the cut-and-choose method during the withdrawal protocol by flipping a coin b and executing the withdrawal protocol correctly if $b = 1$ and asking the user to reveal her binary tree that is finally dropped if $b = 0$. If the revealed tree is correct, the user is honest and the withdrawal protocol is repeated again from the beginning. If the user is a cheater, a fine of value $2L2^L$ is deducted from the user's account. This drawback may be considered as unacceptable from the bank point of view even if the bank should not loose money "on average".

1.2 Our Contribution

We revisit the divisible e-cash approach by targeting both the most demanding security model and the effective possibility to instantiate an e-cash system from a theoretical method. We introduce a new construction based on algebraic objects to generate the binary tree without any previously mentioned problems (impracticability of the key generation [9] and unusual security model [1]). We introduce a new technique to prove the validity of the spending. We show that it is possible to prove that one node key is derived from its father, which is impossible in the proposal of [1]. This enables us to prove that one node key is derived from only its father and we do it only once instead of $(L - \ell)$ times in the scheme proposed in [9] for spending 2^ℓ coins from a divisible coin of 2^L coins. Next, we prove the remainder of the paths from the spent node to the leaves using a variant of the accumulator technique from [1]. In our construction, the spender only sends to the merchant a constant number of bits to spend 2^ℓ coins.

2 Preliminaries

2.1 Construction of the Binary Tree of Keys

In the following, any divisible coin of monetary value 2^L is assigned to a binary tree with $L + 2$ levels, as done in [9]. The value of the tree root (at level 0) is 2^L . Any other internal node in the tree has a value corresponding to half of the amount of its parent node. The leaves (at level $L + 1$) have no monetary value. For every level i , $0 \leq i \leq L + 1$, the 2^i nodes are assigned keys denoted by $k_{i,j}$ with $0 \leq j \leq 2^i - 1$. The following rule must be satisfied in order to protect over-spending (c.f. Definition 3): *when a node n is used, none of descendant and ancestor nodes of n can be used, and no node can be used more than once.*

We now describe a new way to generate the binary tree of keys based on algebraic objects which can be efficiently generated. Let q, p, P be three primes such that p is of size l_p , q is of size l_q and divides $p - 1$, and $P = 2p + 1$. We denote by \mathcal{G}_q (resp. \mathcal{G}_p) the subgroup of \mathbb{Z}_p^* (resp. \mathbb{Z}_P^*) of order q (resp. p) and g_0, g_1 are two generators of \mathcal{G}_q . The keys of the tree are computed from the root to the leaves as follows. Given a key $k_{i,j}$ with $0 \leq i \leq L$ and $0 \leq j \leq 2^i - 1$ of an internal node, the two keys related to its two direct descendants are computed as follows: $k_{i+1,2j} = g_0^{k_{i,j} \pmod q} \pmod p$ and $k_{i+1,2j+1} = g_1^{k_{i,j} \pmod q} \pmod p$.

2.2 Discrete Log Relation Sets

Roughly speaking, a Zero Knowledge Proof of Knowledge (ZPK) is an interactive protocol during which a prover proves to a verifier that he knows a set of secret values $\alpha_1, \dots, \alpha_q$ that verify a given relation R , without revealing the secret values; we denote it by $\text{POK}(\alpha_1, \dots, \alpha_q : R(\alpha_1, \dots, \alpha_q))$. In the following, the secret values are discrete logarithms in relations constructed over a group either of prime or unknown order. These constructions should verify the soundness and zero-knowledge properties [8,4]. The relation R can be a proof of knowledge of a discrete logarithm denoted by $\text{POK}(\alpha : y = g^\alpha)$, a proof of knowledge of a representation, denoted by $\text{POK}(\alpha_1, \dots, \alpha_q : y = g_1^{\alpha_1} \dots g_q^{\alpha_q})$, or a proof of equality of discrete logarithms, denoted by $\text{POK}(\alpha : y = g^\alpha \wedge z = h^\alpha)$. Note that, contrary to [9], we do not use the complex proof of knowledge of double-discrete logarithms. We apply the Fiat-Shamir heuristic [10] to turn it into a signature on some message m : $\text{SOK}(\alpha_1, \dots, \alpha_q : R(\alpha_1, \dots, \alpha_q))(m)$.

2.3 Signature Schemes with Additional Features

Camenisch and Lysyanskaya [7] have proposed various unforgeable signature schemes based on Pedersen's commitment scheme to which they add some specific protocols. There is first an efficient protocol between a user \mathcal{U} and a signer \mathcal{S} that permits \mathcal{U} to obtain from \mathcal{S} a signature σ of some commitment C on values (x_1, \dots, x_ℓ) unknown from \mathcal{S} . \mathcal{S} computes $\text{CLSIGN}(C)$ and \mathcal{U} obtains $\sigma = \text{SIGN}(x_1, \dots, x_\ell)$ and second, an efficient proof of knowledge of a signature of some committed values.

The Extended Special Signature (ESS+) scheme introduced in [2,1] is a variant of the Camenisch-Lysyanskaya (CL) signature scheme that allows to sign a block of messages, one of them being an element in a cyclic group. The user obtains a signature $\sigma = \text{SIGN}(X, x_1, \dots, x_\ell)$ where X is an element of the multiplicative group while the x_i 's are exponents. In our divisible e-cash system, we may use the signature scheme described in [1] together with the following zero-knowledge proof of knowledge: $\text{POK}(X, x_1, \dots, x_\ell, \sigma : \sigma = \text{SIGN}(X, x_1, \dots, x_\ell))$, which is unforgeable under the AWSM assumption [2]. Note that any signature scheme with the same features can also be used.

2.4 Bounded Accumulators

An accumulator scheme Acc is a method which permits to accumulate a large set of objects in a single short value. It next provides evidence that a given object is contained in the accumulator by producing a related witness. We denote by $x \in \text{Acc}$ or $(x, w) \in \text{Acc}$ that the value x is accumulated in Acc , possibly with the witness w . It is possible to prove (in a zero-knowledge manner) that one (secret) value is truly accumulated in a given accumulator such that the computation and the verification of the proof do not depend on the number of accumulated values. The main accumulator schemes are described in [6,12,5]. As noticed in [2], the proposal given in [12] (and this is also the case for [5]) is bounded in the sense that it should not be possible to accumulate more than a given number s of objects, this number being stated at the key generation process. In our scheme, the payer needs to prove that she knows a secret accumulator Acc certified by some authorities in which several revealed values x_1, \dots, x_ℓ are accumulated, that is $\text{POK}(w_1, \dots, w_\ell, \text{Acc}, \sigma : (x_1, w_1) \in \text{Acc} \wedge \dots \wedge (x_\ell, w_\ell) \in \text{Acc} \wedge \sigma = \text{SIGN}(\text{Acc}))$, also denoted $\text{POK}(\text{Acc}, \sigma : (x_1, \dots, x_\ell) \in \text{Acc} \wedge \sigma = \text{SIGN}(\text{Acc}))$. This new feature obviously not introduce any new flaw in the accumulator scheme. In Appendix A, we describe a construction such that this proof does not depend on the number ℓ of revealed values.

3 Model for Divisible E-cash

3.1 Procedures for Divisible E-cash

Three types of actors are involved in a divisible e-cash system: the bank \mathcal{B} , the user \mathcal{U} and the merchant \mathcal{M} . We denote by λ the security parameter. The monetary value of a divisible coin is fixed to 2^L . A divisible e-cash system \mathcal{S} can be defined by the following polynomial-time procedures:

- $\text{SETUP}(1^\lambda)$ is a probabilistic algorithm which outputs the parameters of the system param . In the following, param and 1^λ are implicitly in the input of all algorithms and protocols;
- $\text{BKEYGEN}()$ is a probabilistic algorithm which outputs (bsk, bpk) as the secret and public keys of the bank, respectively. A database cdb of all spent coins is initialized to the empty set ϵ ;

- $\text{UKEYGEN}()$ is a probabilistic algorithm which outputs (usk, upk) as the secret and public keys of the user, respectively. Note that the same algorithm is executed by the merchants to get (msk, mpk) ;
- $\text{WITHDRAW}[\mathcal{B}(\text{bsk}) \longleftrightarrow \mathcal{U}(\text{usk}, \text{bpk})]$ is a protocol which permits \mathcal{U} to withdraw a divisible coin co , while the bank outputs its view viewW ;
- $\text{SPEND}[\mathcal{U}(\text{usk}, \text{co}, \text{bpk}, \ell) \longleftrightarrow \mathcal{M}(\text{msk}, \text{bpk})]$ is a protocol which permits \mathcal{U} to spend a value 2^ℓ from the divisible coin co to the merchant \mathcal{M} . The user outputs a new state for co and \mathcal{M} outputs the received coin rco ;
- $\text{DEPOSIT}[\mathcal{M}(\text{msk}, \text{rco}, \text{bpk}) \longleftrightarrow \mathcal{B}(\text{bsk}, \text{mpk}, \text{cdb})]$ is a protocol which permits \mathcal{M} to deposit a coin rco to the bank. The bank outputs either 1 and the monetary value 2^ℓ , or executes the algorithm IDENTIFY if the database cdb already contains the serial number in rco . This procedure is sometimes written $\text{DEPOSIT}(\text{rco})$ for simplicity.
- $\text{IDENTIFY}(\text{rco}, \text{cdb})$ (or $\text{IDENTIFY}(\text{rco})$ for short) is an algorithm which outputs the public key upk of a fraudulent player (either a user or a merchant) together with a proof π_G ;
- $\text{VERIFYGUILT}(\text{rco}, \text{cdb}, \text{upk}, \pi_G)$ is an algorithm which outputs 1 if π_G is a valid proof that the player’s public key upk has made a fraud during the spending of the coin rco , and 0 otherwise.

In the following, it is assumed that if an honest user runs a WITHDRAW protocol with an honest bank, then neither will output an error message. If an honest user runs a SPEND protocol with an honest merchant, then the merchant always accept the coin.

3.2 Security Properties

The adversary \mathcal{A} interacts with a challenger \mathcal{C} in order to break a security property. The adversary \mathcal{A} has access to the procedures of the system and to the parameters param . In addition, two oracles are defined in order to add and corrupt users: $\text{ADDU}()$ and $\text{CORRUPTU}(j)$, where j is related to the user public key usk_j . In the following, the execution of \mathcal{A} with access to the oracle XXXX and with input e is denoted by $\mathcal{A}^{\text{XXXX}}(e)$.

Unforgeability. It guarantees that no coalition of players can deposit more coins than they have withdrawn from the bank.

Experiment $\text{Exp}_{\mathcal{S}, \mathcal{A}}^{\text{unforge}}(\lambda)$: <ul style="list-style-type: none"> – $(\text{param}) \leftarrow \text{SETUP}(\lambda)$, – $(\text{bsk}, \text{bpk}) \leftarrow \text{BKEYGEN}()$, $(\text{mpk}) \leftarrow \mathcal{A}()$ – $\text{dc} \leftarrow 0$, $\text{sp} \leftarrow 0$, $\text{cont} \leftarrow \text{true}$, – while ($\text{cont} == \text{true}$), <ul style="list-style-type: none"> $b \leftarrow \text{WITHDRAW}[\mathcal{C}(\text{bsk}) \longleftrightarrow \mathcal{A}(\text{bpk})]$ if ($b == 1$), then $\text{dc} \leftarrow \text{dc} + 1$ $(\text{rco}, \ell) \leftarrow \mathcal{A}(\text{bpk})$ if ($\text{DEPOSIT}(\text{rco}) == 1$), then $\text{sp} \leftarrow \text{sp} + 2^\ell$ $\text{cont} \leftarrow \mathcal{A}()$ – if $2^L \cdot \text{dc} < \text{sp}$ return 1 – return 0

The success probability of \mathcal{A} is defined by $Succ_{\mathcal{S}, \mathcal{A}}^{\text{unforge}}(\lambda) = \Pr [Exp_{\mathcal{S}, \mathcal{A}}^{\text{unforge}}(\lambda) = 1]$.

Definition 1 (Unforgeability). A system \mathcal{S} is unforgeable if for any polynomial-time adversary \mathcal{A} , the success probability $Succ_{\mathcal{S}, \mathcal{A}}^{\text{unforge}}(\cdot)$ is negligible.

Anonymity. It guarantees that the bank, even helped by malicious users, cannot learn anything about a spending other than what is available from side information from the environment. In the following experiment, b is a bit.

Experiment $Exp_{\mathcal{S}, \mathcal{A}}^{\text{anon}-b}(\lambda)$:

- $(\text{param}) \leftarrow \text{SETUP}(\lambda), (\text{bpk}) \leftarrow \mathcal{A}()$
- $(\text{upk}_0, \text{upk}_1) \leftarrow \mathcal{A}^{\text{WITHDRAW}, \text{SPEND}, \text{ADDU}, \text{CORRUPTU}}()$
- $\text{rco} \leftarrow \text{SPEND}[\mathcal{C}(\text{usk}_b) \leftarrow \mathcal{A}()]$
- return $b' \leftarrow \mathcal{A}^{\text{WITHDRAW}, \text{SPEND}, \text{ADDU}, \text{CORRUPTU}}(\text{rco})$

The advantage of \mathcal{A} for the anonymity experiment is defined by:

$$Adv_{\mathcal{S}, \mathcal{A}}^{\text{anon}}(\lambda) = \Pr [Exp_{\mathcal{S}, \mathcal{A}}^{\text{anon}-1}(\lambda) = 1] - \Pr [Exp_{\mathcal{S}, \mathcal{A}}^{\text{anon}-0}(\lambda) = 1].$$

Definition 2 (Anonymity). A system \mathcal{S} is anonymous if for any polynomial-time adversary \mathcal{A} , the adversary advantage $Adv_{\mathcal{S}, \mathcal{A}}^{\text{anon}}(\cdot)$ is negligible.

Identification of double-spenders. From the bank's point of view, no collection of users should be able to double-spend a coin without revealing one of their identities.

Experiment $Exp_{\mathcal{S}, \mathcal{A}}^{\text{idds}}(\lambda)$:

- $(\text{param}) \leftarrow \text{SETUP}(\lambda), (\text{bsk}, \text{bpk}) \leftarrow \text{BKEYGEN}()$
- $\text{rco} \leftarrow \mathcal{A}^{\text{WITHDRAW}, \text{SPEND}, \text{ADDU}, \text{CORRUPTU}}(\text{bpk})$
- if $(\text{DEPOSIT}(\text{rco}) == 0 \wedge \text{VERIFYGUILT}(\text{rco}, \text{IDENTIFY}(\text{rco})) == 0)$ return 1
- return 0

The success probability of \mathcal{A} is defined by $Succ_{\mathcal{S}, \mathcal{A}}^{\text{idds}}(\lambda) = \Pr [Exp_{\mathcal{S}, \mathcal{A}}^{\text{idds}}(\lambda) = 1]$.

Definition 3 (Identification of double spender). A system \mathcal{S} identifies double-spender if for any polynomial-time adversary \mathcal{A} , the success probability $Succ_{\mathcal{S}, \mathcal{A}}^{\text{idds}}(\cdot)$ is negligible.

Exculpability. It guarantees that the bank, even cooperating with malicious users, cannot falsely accuse honest users from having double-spent a coin. In the experiment, \mathcal{CU} is the set of corrupted users.

Experiment $Exp_{\mathcal{S}, \mathcal{A}}^{\text{exculp}}(\lambda)$:

- $(\text{param}) \leftarrow \text{SETUP}(\lambda), (\text{bpk}) \leftarrow \mathcal{A}()$
- $\text{cont} \leftarrow \text{true}, \mathcal{CU} \leftarrow \emptyset$
- while ($\text{cont} == \text{true}$),
 - $(j, \text{cont}) \leftarrow \mathcal{A}^{\text{WITHDRAW}, \text{SPEND}, \text{ADDU}, \text{CORRUPTU}}(\mathcal{CU}), \mathcal{CU} \leftarrow \mathcal{CU} \cup \{\text{upk}_j\}$
- $\text{rco} \leftarrow \mathcal{A}^{\text{WITHDRAW}, \text{SPEND}, \text{ADDU}, \text{CORRUPTU}}()$
- if $(\text{IDENTIFY}(\text{rco}) = (\text{upk}, \pi_G) \wedge \text{VERIFYGUILT}(\text{rco}, \text{upk}, \pi_G) = 1 \wedge \text{upk} \notin \mathcal{CU})$ return 1
- return 0

The success probability of \mathcal{A} is defined by $Succ_{\mathcal{S}, \mathcal{A}}^{\text{exculp}}(\lambda) = \Pr [Exp_{\mathcal{S}, \mathcal{A}}^{\text{exculp}}(\lambda) = 1]$.

Definition 4 (Exculpability). A system \mathcal{S} is exculpable if for any polynomial-time adversary \mathcal{A} , the success probability $Succ_{\mathcal{S}, \mathcal{A}}^{\text{exculp}}(\cdot)$ is negligible.

4 Description of Our Divisible E-cash Construction

4.1 Setup and Key Generation Procedures

A divisible coin has a value set to 2^L , where L is a positive integer. As in [9], a divisible coin in our system is represented by a binary tree of $L + 2$ levels and the leaves have no value.

Let λ be the security parameter. Let q, p, P be three primes such that q divides $p - 1$ and $P = 2p + 1$. The size of p (resp. q) is denoted by l_p (resp. l_q). We denote by \mathcal{G}_q (resp. \mathcal{G}_p) the subgroup of \mathbb{Z}_p^* (resp. \mathbb{Z}_P^*) of order q (resp. p); g_0 and g_1 (resp. G and H) are two generators of \mathcal{G}_q (resp. \mathcal{G}_p). Finally, let $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$ be a collision-resistant hash function.

The parameters of $L + 2$ bounded accumulators $\text{Acc}, \text{Acc}_1, \dots, \text{Acc}_{L+1}$ on the cyclic group \mathcal{G}_p are generated during the SETUP procedure (see Appendix A). The accumulator Acc is bounded to $2^{L+2} - 2$ in order to accumulate all the keys of nodes in the tree from level 1 to level $L + 1$ (and thus the key of the root is not accumulated in Acc). The accumulator Acc_i is bounded to 2^i in order to accumulate all the keys of nodes at level i , with $i \in [1, L + 1]$. Then, it is not possible to accumulate more than 2^i keys of value 2^{L-i} (see Figure 1).

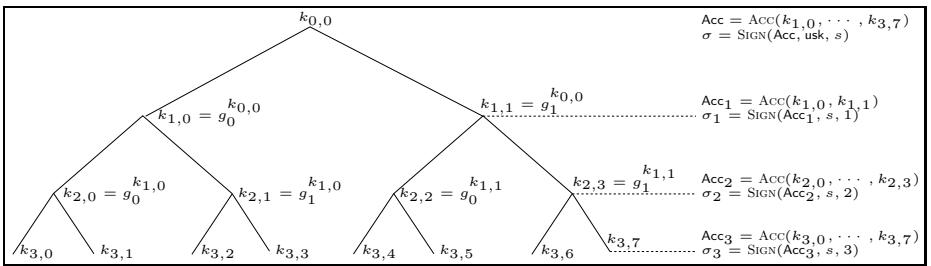


Fig. 1. Our new binary tree for a coin of monetary value 2^2 with $L = 2$

The algorithm BKEYGEN is performed by the bank in order to generate a key pair (bsk, bpk) for the signature scheme ESS+ [1] on the group \mathcal{G}_p . The algorithm UKEYGEN is executed by any user and merchant of the system. It consists in randomly choosing a secret $\text{usk} \in \mathbb{Z}_p^*$ (resp. $\text{msk} \in \mathbb{Z}_p^*$) and computing $\text{upk} = G^{\text{usk}}$ (resp. $\text{mpk} = G^{\text{msk}}$). Moreover, any user public key is assumed to be certified by an authority and the bank can be convinced that it belongs to a known identified user.

4.2 The Withdrawal Protocol

The withdrawal phase is a protocol between the user \mathcal{U} (on input usk and bpk) and the bank \mathcal{B} (on input bsk), which permits \mathcal{U} to withdraw a coin of value 2^L . In a nutshell, \mathcal{U} computes the keys $k_{0,0}, \dots, k_{L+1,2^{L+1}-1}$ of the binary tree

and next the accumulators $\text{Acc}, \text{Acc}_1, \dots, \text{Acc}_{L+1}$. Next, \mathcal{B} produces $L+2$ ESS+ signatures on the messages $(\text{Acc}, \text{usk}, s), (\text{Acc}_1, s, 1), \dots, (\text{Acc}_{L+1}, s, L+1)$. These signatures give a proof of the interaction between the user knowing usk and the bank. The secret value s is used to link together the $L+2$ signatures of a given withdrawal protocol knowing that the user and bank contribute randomness to the value s .

An important remark is that it is not necessary for the bank to check if the tree of keys is well-formed or if the accumulators are well-formed since they are bounded using appropriate values. As explained in the following, if the user cheats in the construction of the tree of keys or in the construction of the accumulated values, he won't be able to correctly execute the spending protocol, as the merchant will be able to make all validity checks. More formally, the protocol works as follows:

- \mathcal{U} chooses at random the key root $k_{0,0} \in \mathbb{Z}_p^*$ and computes the keys of the full tree: given a key node $k_{i,j}$ with $0 \leq i \leq L$ and $0 \leq j \leq 2^i - 1$, the keys related to its two direct descendants are $k_{i+1,2j} = g_0^{k_{i,j} \pmod q} \pmod p$ and $k_{i+1,2j+1} = g_1^{k_{i,j} \pmod q} \pmod p$. The keys are stored in a table tr ;
- \mathcal{U} accumulates the keys $k_{i,j}$, for all $1 \leq i \leq L+1$ and $0 \leq j \leq 2^i - 1$, in Acc and sends it to \mathcal{B} . Next \mathcal{U} and \mathcal{B} interact using the ESS+ interactive protocol in order to get a signature σ on $(\text{Acc}, \text{usk}, s)$, where $s \in \mathbb{Z}_p^*$ is a secret value only known by the user and jointly generated by both \mathcal{U} and \mathcal{B} . For this purpose, the user commits to the values usk and s' and the bank modifies s' to $s = s' + s''$ (without learning any information about s'), produces the commitment to Acc , usk and s and signs this commitment. Note that \mathcal{B} can verify that usk is related to a known public key upk (*i.e.* by making \mathcal{U} produces a proof of knowledge of usk such that $\text{upk} = G^{\text{usk}}$).
- for every i such that $1 \leq i \leq L+1$, \mathcal{U} accumulates in Acc_i the keys $k_{i,j}$, with $j \in [0; 2^i - 1]$. Next, \mathcal{U} sends $\text{Acc}_1, \dots, \text{Acc}_{L+1}$ to \mathcal{B} and interacts with \mathcal{B} using the ESS+ interactive protocol in order to get the signatures $\sigma, \sigma_1, \dots, \sigma_{L+1}$ on $(\text{Acc}_1, s, 1), \dots, (\text{Acc}_{L+1}, s, L+1)$, respectively. For this purpose, the user commits s' and proves that this is the same as in the previous step. The bank verifies the proof, again modifies s' to $s = s' + s''$, produces the commitment on Acc_i , s and i and signs it. Note that one single commitment to s' by the user is enough to obtain all the signatures.

At the end of this protocol, \mathcal{U} outputs a coin $\text{co} = \{\text{tr}, \text{Acc}, \text{Acc}_1, \dots, \text{Acc}_{L+1}, \sigma, \sigma_1, \dots, \sigma_{L+1}, \text{spc}\}$, where $\text{spc} \leftarrow \epsilon$ will contain information on spent nodes. Note that tr can be either erased or kept to avoid the re-computation of the key nodes during the spending phase.

4.3 The Spending Phase

We suppose that a user \mathcal{U} , with keys (usk, upk) and with a coin $\text{co} = \{\text{tr}, \text{Acc}, \text{Acc}_1, \dots, \text{Acc}_{L+1}, \sigma, \sigma_1, \dots, \sigma_{L+1}, \text{spc}\}$ wants to spend a value $2^\ell \leq 2^L$ to a merchant \mathcal{M} (with input msk and bpk). Informally, \mathcal{U} chooses an unspent

node j_0 in the tree at level $L - \ell$ and computes 1) a serial number S as the concatenation of the keys $k_{L-\ell+1,2j_0}$ and $k_{L-\ell+1,2j_0+1}$ of the two descendant nodes and 2) the security tag using the key $k_{L-\ell,j_0}$. In addition, \mathcal{U} produces a proof of validity of the accumulators and the ESS+ signatures coming from the withdrawal protocol. More formally, the protocol works as follow:

- \mathcal{U} receives \mathcal{M} 's public key mpk and the proof π that \mathcal{M} knows msk . Next, \mathcal{U} and \mathcal{M} can compute $R = \mathcal{H}(\text{mpk} \parallel \text{info})$ where info is a pre-determined public information including the monetary value 2^ℓ and e.g. current time;
- \mathcal{U} chooses in the binary tree an unspent node j_0 at level $L - \ell$ and finds in tr (or recompute) the corresponding key $k_{L-\ell,j_0}$ and its two direct descendants $k_{L-\ell+1,2j_0}$ and $k_{L-\ell+1,2j_0+1}$;
- the serial number is then formed as $S = k_{L-\ell+1,2j_0} \| k_{L-\ell+1,2j_0+1}$ and the security tag of this spending is $T = \text{upk} \cdot H^{R \cdot k_{L-\ell,j_0}}$;
- finally \mathcal{U} produces the signature of knowledge:

$$\begin{aligned} \Pi &= \text{SOK}(\text{usk}, k_{L-\ell,j_0}, s, \text{Acc}, \sigma, \text{Acc}_{L-\ell+1}, \sigma_{L-\ell+1} : \\ T &= G^{\text{usk}} \cdot (H^R)^{k_{L-\ell,j_0}} \wedge k_{L-\ell+1,2j_0} = g_0^{k_{L-\ell,j_0}} \wedge k_{L-\ell+1,2j_0+1} = g_1^{k_{L-\ell,j_0}} \wedge \\ &(k_{L-\ell+1,2j_0}, k_{L-\ell+1,2j_0+1}, \dots, k_{L+1,2^{\ell+1}j_0}, \dots, k_{L+1,2^{\ell+1}(j_0+1)-1}) \in \text{Acc} \wedge \\ &(k_{L-\ell+1,2j_0}, k_{L-\ell+1,2j_0+1}) \in \text{Acc}_{L-\ell+1} \wedge \sigma = \text{SIGN}(\text{Acc}, u, s) \\ \sigma_{L-\ell+1} &= \text{SIGN}(\text{Acc}_{L-\ell+1}, s, L - \ell + 1))(\text{mpk} \parallel \text{info} \parallel R \parallel S \parallel T) \end{aligned}$$

The spent coin is $\text{rco} = \{\ell, S, T, \Pi, R\}$. Its validity is checked by \mathcal{M} by computing all the descendant keys of S before performing the verification of Π (see below). Moreover, the merchant, using the parameters used in the proof of knowledge Π , can check that the accumulator $\text{Acc}_{L-\ell+1}$ has been signed with the value $L - \ell + 1$, that the used parameters correspond to the ones of the right bounded accumulator. The divisible coin of \mathcal{U} is updated as $\text{co} = \{\text{tr}, \text{Acc}, \text{Acc}_1, \dots, \text{Acc}_{L+1}, \sigma, \sigma_1, \dots, \sigma_{L+1}, \text{spc} = \text{spc} \cup \{(L - \ell, j_0)\}\}$.

The proof Π is done non-interactively by using usual zero-knowledge proofs of knowledge (see Appendix B) and the Fiat-Shamir heuristic [10], in the random oracle model, using $m = \text{mpk} \parallel \text{info} \parallel R \parallel S \parallel T$ as a message. It proves that

- the security tag T is correctly computed from usk , R and $k_{L-\ell,j_0}$;
- the serial number S is correctly computed from $k_{L-\ell,j_0}$;
- all the descendants of the node $k_{L-\ell,j_0}$ are accumulated in Acc ;
- $k_{L-\ell+1,2j_0}$ and $k_{L-\ell+1,2j_0+1}$ are accumulated in $\text{Acc}_{L-\ell+1}$;
- the values Acc , usk and s (resp. $\text{Acc}_{L-\ell+1}$, s and $L - \ell + 1$) are signed by the bank in σ (resp. $\sigma_{L-\ell+1}$).

Note that the spender only needs to prove that $k_{L-\ell+1,2j_0}$ and $k_{L-\ell+1,2j_0+1}$ are correctly derived from $k_{L-\ell,j_0}$. This is not necessary for the other descendant nodes. In fact, the receiver can easily compute all the descendant of $k_{L-\ell+1,2j_0}$ and $k_{L-\ell+1,2j_0+1}$. As they are all accumulated into Acc and both $k_{L-\ell+1,2j_0}$ and $k_{L-\ell+1,2j_0+1}$ are accumulated in $\text{Acc}_{L-\ell+1}$, it is enough to prove that the spent coin is correct.

As shown in Appendix B, this proof is done in constant time. It does not depend on the monetary value 2^ℓ which is spent, except when the spender needs to develop a polynomial of degree 2^ℓ , which is quite immediate in practice. Moreover, as the merchant can compute all the keys from the ones used in the serial number to the ones of the leaves, the user does not need to send them to the merchant. Thus, the transaction data sent to the merchant has a constant number of bits while spending a monetary value of 2^ℓ .

4.4 Deposit and Detection of Frauds

The deposit of a coin $\text{rco} = \{\ell, S, T, \Pi, R\}$ with value ℓ is done by \mathcal{M} which sends it to \mathcal{B} with a signature of the deposit request. First \mathcal{B} verifies the correctness of Π . If it is correct, \mathcal{B} computes the keys related to the $2^{\ell+1}$ leaves of $S = k_{L-\ell+1,2j_0} \| k_{L-\ell+1,2j_0+1}$ at level L in the tree. If at least one of these keys is already in its database cdb , \mathcal{B} executes the procedure of double-spender identification. Else, \mathcal{B} adds the $2^{\ell+1}$ leaf keys in cdb and outputs 1. \mathcal{B} could store only 2^ℓ keys, i.e. it will always store leaves that are “right” (or left) child.

In case of a double-spending detection, the bank \mathcal{B} , given two coins $\text{rco}_1 = \{\ell_1, S_1, T_1, \Pi_1, R_1\}$ and $\text{rco}_2 = \{\ell_2, S_2, T_2, \Pi_2, R_2\}$, tests if $R_1 = R_2$ which means that \mathcal{M} is a cheater since the hash function \mathcal{H} is collision-resistant. The proof of the cheat consists in publishing both deposits (including two signatures of \mathcal{M} on the deposit request for the same coin). Else \mathcal{B} will identify a user public key using the same technique as described in [9]. We distinguish two cases:

1. if $\ell_1 = \ell_2 = \ell$, then the same node key $k_{L-\ell,j_0}$ has been used in both T_1 and T_2 . Thus, \mathcal{B} computes $\text{upk} = (T_1^{R_2}/T_2^{R_1})^{\frac{1}{R_2-R_1}}$;
2. if $\ell_1 \neq \ell_2$ (e.g. $\ell_1 < \ell_2$), then from S_2 , \mathcal{B} can compute $k_{L-\ell_1,j_0}$ such that $T_1 = \text{upk} \cdot H^{R_1 \cdot k_{L-\ell_1,j_0}}$ and thus retrieve $\text{upk} = T_1/H^{R_1 \cdot k_{L-\ell_1,j_0}}$;

Finally, \mathcal{B} outputs the proof π_G based on the two entries in the database cdb .

4.5 Efficiency Considerations

We compare the efficiency of the strongly anonymous divisible e-cash schemes of the state-of-the-art [9,1] with our new proposal. We give in Table 1 the computation cost of the binary tree, the time complexity of the withdrawal and spending phases and the size of the divisible coin, where EXP is a modular exponentiation, and $\text{DEV}(i)$ is the time needed to develop a polynomial of degree i . We differentiate in our comparison both types of secure divisible e-cash systems depending on the security model, i.e. classical model with truly unforgeability [9] and unusual model with a statistical balance assumption [1]. We do not include the complexity of the deposit phase and the size of the database which are similar in the three schemes.

Based on Table 1, we can conclude that our new proposal is significantly more efficient than the one of Canard-Gouget [9], regarding the spending phase, with the same security level. Our new proposal is little bit less efficient than the Au *et al.* one [1] but with a better security result. We consequently obtain the best trade off between previous approaches, considering efficiency and security.

Table 1. Efficiency comparison between related work and our proposal

Divisible e-cash scheme	Au <i>et al.</i> [1]	Canard-Gouget [9]	this paper
Model choice	Statistical balance	Unforgeability	
Binary tree computation	$(2^{L+1} - 2)\text{EXP}$	$(2^{L+2} - 2)\text{EXP}$ (not necessarily computed)	$(2^{L+2} - 2)\text{EXP}$
Divisible coin storage size	$(2^{L+1} - 2) p + (L + 1)\text{SIGN}$ +($L + 1\text{ACC}$)	$2 p + (1)\text{SIGN}$	$(2^{L+2} - 2) p + (L + 2)\text{SIGN}$ +($L + 2\text{ACC}$)
Computational complexity of withdraw	$(L + 1)\text{SIGN}$ +($L + 1\text{ACC}$)	$(1)\text{SIGN}$	$(L + 2)\text{SIGN}$ +($L + 2\text{ACC}$)
Computational complexity of spending 2^ℓ	$\text{EXP} + \text{SOK}(\text{SIGN}$ + $\text{ACC} + 2\text{EXP})$	$(i + 3)\text{EXP} + \text{SOK}(\text{SIGN}$ + $\mathcal{O}((L - \ell)t)\text{EXP})$	$\text{EXP} + \text{DEV}(2^\ell) +$ $\text{SOK}(2\text{SIGN} + 2\text{ACC} + 3\text{EXP})$
Spending transfer size	$2 p + \text{SOK}(\text{SIGN}$ + $\text{ACC} + 2\text{EXP})$	$3 p + \text{SOK}(\text{SIGN}$ + $\mathcal{O}((L - \ell)t)\text{EXP})$	$2 p + P + \text{SOK}(2\text{SIGN}$ + $2\text{ACC} + 3\text{EXP})$

4.6 Security Theorem

We give the statement of security for our new proposed scheme.

Theorem 1. *In the random oracle model, our divisible e-cash scheme fulfills (i) the unforgeability under the assumptions that ESS is unforgeable and the bounded accumulator scheme fulfills the bound property; (ii) the anonymity under the zero-knowledge property of ZKPK and the DDH assumption; (iii) the identification of double-spender under the unforgeability of ESS and (iv) the exculpability under the one-more discrete logarithm assumption.*

Proof. We consider the four properties.

– **Anonymity:** we use a reduced “game proof technique” from Shoup. We denote by ϵ the probability that \mathcal{A} succeeds in linking the spending protocol V_1 to a spending or withdrawal protocol.

During V_1 , \mathcal{A} gets a serial number S , a security tag T , a zero-knowledge proof of knowledge Π and a value R . It is obvious that R does not reveal any Shannon information on the user identity. Under the zero-knowledge property of the proof of knowledge Π , in the random oracle model, the value Π does not help \mathcal{A} in winning the anonymity experiment. Thus, V_1 is replaced by V_2 in which \mathcal{A} gets only S and T . The difference of probability between V_1 and V_2 is the probability of success of \mathcal{A} in breaking the zero-knowledge property of Π , namely $\text{Succ}_{\Pi, \mathcal{A}}^{\text{zk}}(\lambda)$.

In V_2 , we focus on T . The secret key $k_{L-\ell, j_0}$ is used only in the computation of T and Π . Indeed, knowing $T = \text{upk} \cdot H^{Rk}$ and $T' = \text{upk}' \cdot H^{R'k'}$, \mathcal{A} has to decide if the same upk is embedded in both T and T' . This is assumed to be infeasible under the discrete logarithm (DL) assumption. Thus, V_2 is replaced by V_3 in which \mathcal{A} gets only S . The difference of probability between V_2 and V_3 is the probability of success of \mathcal{A} in breaking the DL problem, namely $\text{Succ}_{\mathcal{A}}^{\text{dl}}(\lambda)$.

Finally, from S , \mathcal{A} needs to decide whether or not two different node keys k and k' are related to the same root key k_0 . This is assumed to be infeasible under a stronger variant of the Decisional Diffie-Hellman assumption (see [9] for details). We denote by $Adv_{\mathcal{A}}^{\text{ddh}}(\lambda)$ the corresponding success probability.

We conclude that $Adv_{DCS,\mathcal{A}}^{\text{anon}}(\lambda) \leq Succ_{H,\mathcal{A}}^{\text{zk}}(\lambda) + Adv_{\mathcal{A}}^{\text{ddh}}(\lambda)$.

- **Unforgeability:** we use a reduction to either the unforgeability of the signature scheme or to the bound property of the accumulator scheme. Let \mathcal{A} be an adversary that breaks the unforgeability property in polynomial time τ with a probability of success equal to ϵ . We interact with the black box adversary \mathcal{A} by flipping at random a coin and playing one among two games.

Game 1. We construct a machine \mathcal{M}_1 that breaks the existential unforgeability of ESS+ with access to a signing oracle SIGN:

- In the SETUP procedure, \mathcal{M}_1 sets the group defined for the signature scheme as \mathcal{G}_p and \mathcal{A} is given the public key $\mathsf{bpk} = \mathsf{spk}$ of the signature scheme.
- when \mathcal{A} ask for a withdrawal, dc is incremented by 1 and \mathcal{M}_1 , playing the role of the bank, interacts with \mathcal{A} by interacting with the oracle SIGN to obtain ESS+ signatures. Each time, \mathcal{M}_1 stores in askdb the signature and the corresponding messages;
- when \mathcal{A} asks for a spending protocol of 2^ℓ coins, sp is incremented by 2^ℓ and \mathcal{M}_1 , playing the role of the merchant, rewinds \mathcal{A} during its computation of the zero-knowledge proof of knowledge by using standard techniques (and in the random oracle model) in order to extract and store in recdb the signatures and the corresponding messages used by \mathcal{A} .
- at any time of the unforgeability experiment, \mathcal{A} sets $\mathsf{cont} = \text{false}$ and with probability ϵ , we have $2^L \cdot \mathsf{dc} < \mathsf{sp}$. In case there is one entry in $\mathsf{recdb} \setminus \mathsf{askdb}$. This entity is obviously a forgery.

Game 2. We construct a machine \mathcal{M}_2 that breaks the bound property of the accumulator scheme:

- In the SETUP procedure, \mathcal{M}_2 sets the group defined for the signature scheme as \mathcal{G}_p and \mathcal{A} is given the public key $\mathsf{bpk} = \mathsf{spk}$ of the signature scheme. Next, \mathcal{M}_2 generates the $L+2$ accumulators. \mathcal{M}_2 maintains a database accdb containing all accumulators generated by \mathcal{A} together with the bound of the corresponding bounded accumulator and the obtained accumulated values;
- when \mathcal{A} ask for a withdrawal, \mathcal{M}_2 plays the role of the bank;
- when \mathcal{A} asks for a spending protocol of 2^ℓ coins, sp is incremented by 2^ℓ and \mathcal{M}_2 , playing the role of the merchant, rewinds \mathcal{A} during its computation of the zero-knowledge proof of knowledge, using standard technique (and in the random oracle model) in order to extract the accumulators. Next, \mathcal{M}_2 stores in accdb the accumulator (if not already present in the database) and the corresponding accumulated values;

- at any time of the unforgeability experiment, \mathcal{A} sets `cont` = false and with probability ϵ , we have $2^L \cdot \text{dc} < \text{sp}$. In case there is one entry in `accdb` such that there are more accumulated values than the bound for this accumulator, \mathcal{M}_2 breaks the bound property of the accumulator scheme.

As a consequence, $\text{Succ}_{\text{DCS}, \mathcal{A}}^{\text{unforge}}(\lambda) = \frac{1}{2} (\text{Succ}_{\text{Sign}, \mathcal{A}}^{\text{unforge}}(\lambda) + \text{Succ}_{\text{Acc}, \mathcal{A}}^{\text{bound}}(\lambda))$.

- **Identification of Double Spender:** we use a reduction to the unforgeability of the signature scheme. Let \mathcal{A} be an adversary breaking the identification of double spender in polynomial time τ with a probability of success ϵ . We consider a black box adversary and construct a machine \mathcal{M} which breaks the unforgeability of the signature scheme. We can access to the group \mathcal{G}_p , the public key `spk` and interact with a signing oracle:

- In the `SETUP` procedure, \mathcal{M} sets the group defined for the signature scheme as \mathcal{G}_p and \mathcal{A} is given the public key `bpk = spk` of the signature scheme.
- when \mathcal{A} ask for a withdrawal, `dc` is incremented by 1 and \mathcal{M}_1 , playing the role of the bank, interacts with \mathcal{A} by interacting with the `SIGN` oracle to obtain ESS+ signatures. Each time, \mathcal{M} stores in `askdb` the signature and the corresponding messages;
- when \mathcal{A} asks for a spending protocol, \mathcal{M} , playing the role of the merchant, rewinds the adversary \mathcal{A} during its computation of the zero-knowledge proof of knowledge to extract and store in `recdb` the signatures and the corresponding messages used by \mathcal{A} ;
- at any time of the experiment, \mathcal{A} outputs one spent coin `rco` and, with probability ϵ , the `DEPOSIT` and the `VERIFYGUILT` procedures output 0. Thus, \mathcal{M} takes on input `rco` and the other coin with the same serial number and extracts, in $\Pi \in \text{rco}$ and using standard techniques, both signatures and the corresponding messages. Necessarily, one of the two signatures is not an output of the signing oracle since the signed `upk` is not detected by the `IDENTIFY` algorithm. Thus, \mathcal{M} has produced a forge on the signature scheme.

We have $\text{Succ}_{\text{DCS}, \mathcal{A}}^{\text{idds}}(\lambda) = \text{Succ}_{\text{Sign}, \mathcal{A}}^{\text{unforge}}(\lambda)$.

- **Exculpability:** suppose that an adversary \mathcal{A} succeeded in breaking the exculpability property. That means that there are two valid spends with the same serial number S (or, either S can be computed from S' , or S' can be computed from S) and two different proofs Π and Π' and two different correct randoms R and R' . As spendings are correct, the proofs include that both T and T' are well formed. Thus, since the user is honest, \mathcal{A} has faked T or T' .

We now use \mathcal{A} to break the one-more discrete logarithm problem [3]. Given $l + 1$ values, we have to find the discrete logarithm of all these values, and we can ask a discrete logarithm oracle at most l times. We first associate each value to the public key of one user (assuming there are at most l users) and we ask the oracle each time \mathcal{A} corrupt a user. It is possible to simulate all withdrawals and spends using standard techniques (in the random oracle model). \mathcal{A} finally outputs two correctly formed T and T' and the associated proofs of validity. Thus, T and T' are both formed from the same public key of a honest user.

From the two proofs of validity, we can extract the user secret key and thus break the one-more discrete logarithm. Indeed, since the user is honest, this discrete logarithm has not been requested to the oracle. We consequently have $\text{Succ}_{\mathcal{DCS}, \mathcal{A}}^{\text{exculp}}(\lambda) = \text{Succ}_{\mathcal{A}}^{\text{omdl}}(\lambda)$, which concludes the proof. \square

5 Conclusion

We have presented the first strongly anonymous and unforgeable e-cash scheme that can be instantiated in practice since it is possible to efficiently generate the parameters of the system. This new construction makes use, first, of a new generation method of the binary tree of keys based on algebraic objects which can be easily used in practice, and second, of a new technique to use bounded accumulators in divisible e-cash. The time complexity of the main critical protocol, which is the spending protocol, is relatively small in time and space for spending a monetary value 2^ℓ .

One may now think of designing a new system with the same features but additionally with a more efficient withdrawal procedure. It may also be possible to design a divisible e-cash system in the standard model, using the so-called Groth-Sahai proofs instead of the Fiat-Shamir heuristic.

Acknowledgements

We are grateful to Déborah Jourdes for her suggestions of improvement, and to anonymous referees for their valuable comments.

References

1. Au, M.H., Susilo, W., Mu, Y.: Practical anonymous divisible e-cash from bounded accumulators. In: Financial Cryptography 2008, LNCS. Springer, Heidelberg (2008) (to appear)
2. Au, M.H., Wu, Q., Susilo, W., Mu, Y.: Compact e-cash from bounded accumulator. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 178–195. Springer, Heidelberg (2007)
3. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The One-More-RSA-Inversion Problems and the Security of Chaum’s Blind Signature Scheme. *J. Cryptology* 16(3), 185–215 (2003)
4. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized schnorr proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2010)
5. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credential. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. 5443, vol. LNCS, pp. 481–500. Springer, Heidelberg (2009) (to appear)
6. 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)

7. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
8. Canard, S., Coisel, I., Traoré, J.: Complex zero-knowledge proofs of knowledge are easy to use. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 122–137. Springer, Heidelberg (2007)
9. Canard, S., Gouget, A.: Divisible e-cash systems can be truly anonymous. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 482–497. Springer, Heidelberg (2007)
10. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
11. Nakanishi, T., Sugiyama, Y.: Unlinkable divisible electronic cash. In: Okamoto, E., Pieprzyk, J.P., Seberry, J. (eds.) ISW 2000. LNCS, vol. 1975, pp. 121–134. Springer, Heidelberg (2000)
12. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)

A Bounded Accumulator with Additional Procedure

We give a concrete example of accumulator based on [12]. Let \mathcal{G} and $\tilde{\mathcal{G}}$ be two cyclic groups of prime order p , G (resp. \tilde{G}) is a generator of \mathcal{G} (resp. $\tilde{\mathcal{G}}$). Let \mathcal{G}_T be a multiplicative group of order p . We refer a bilinear structure for the groups $(\mathcal{G}, \tilde{\mathcal{G}}, \mathcal{G}_T)$. Let e be a bilinear map $e : \mathcal{G} \times \tilde{\mathcal{G}} \longrightarrow \mathcal{G}_T$.

Using [12], the number of values accumulated in a single accumulator is limited to a fixed number which is here denoted by s . In our construction, we use a new proof of knowledge, denoted $\text{POK}(\text{Acc} : (k_1, \dots, k_\ell) \in \text{Acc})$, to prove that several values known by the verifier are accumulated in a secret accumulator.

Let $U_0 \in \mathcal{G}$, $\tilde{V}_0 \in \tilde{\mathcal{G}}$, $\alpha \in \mathbb{Z}_p^*$, $U_i = U_0^{\alpha^i}$ for all $i \in \{1, \dots, s\}$ and $\tilde{V}_1 = \tilde{V}_0^\alpha$. The values (k_1, \dots, k_s) are accumulated in Acc as $\text{Acc} = U_0^{\prod_{i=1}^s (k_i + \alpha)}$. We denote by W the value $U_0^{\prod_{j \notin \{1, \ell\}} (k_j + \alpha)}$. Thus, we have $\text{Acc} = W^{\prod_{j \in \{1, \ell\}} (k_j + \alpha)}$. Let $P(\alpha) = \prod_{j \in \{1, \ell\}} (k_j + \alpha) = \sum_{j=1}^\ell p_j \alpha^j$ where the p_j 's are product of the k_i 's and the degree of P is ℓ . More precisely, we assume that there exists a public function $F(\{k_1, \dots, k_\ell\}) = \{p_1, \dots, p_\ell\}$. We need to introduce some additional public parameters: $\tilde{V}_i = \tilde{V}_0^{\alpha^i}$ for all $i \in \{2, \dots, s\}$. Then, we have $e(\text{Acc}, \tilde{V}_0) = e(W^{\prod_{j \in \{1, \ell\}} (k_j + \alpha)}, \tilde{V}_0) = e(W, \tilde{V}_0^{\prod_{j \in \{1, \ell\}} (k_j + \alpha)}) = e(W, \tilde{V}_0 \prod_{j=1}^\ell \tilde{V}_j^{p_j})$. In our setting, the values k_i 's are public, and thus it is also the case for the values p_j 's. Our ZPK is then: $\text{POK}(\text{Acc}, W : e(\text{Acc}, v_0) = e(W, \tilde{V}_0 \prod_{j=1}^\ell \tilde{V}_j^{p_j}))$.

B Proof of Validity of a Spending

The first part of the proof, consisting in proving that one knows usk and $k_{L-\ell, j_0}$ such that $T = G^{\text{usk}} \cdot (H^R)^{k_{L-\ell, j_0}}$, $k_{L-\ell+1, 2j_0} = g_0^{k_{L-\ell, j_0}}$ and $k_{L-\ell+1, 2j_0+1} = g_1^{k_{L-\ell, j_0}}$ is simply done by using standard discrete logarithm based proof of

knowledge. The next part of the proof consists in proving that several values known by the verifier are accumulated in a secret accumulator which is signed by the bank, using the ESS+ scheme. We describe the case of $\text{Acc}_{L-\ell+1}$; the case of Acc is similar.

We first need a proof that the tuple $(k_{L-\ell+1,2j_0}, k_{L-\ell+1,2j_0+1}) \in \text{Acc}_{L-\ell+1}$. Everyone can compute $\{\mathbf{p}_1, \mathbf{p}_2\} = \mathsf{F}(k_{L-\ell+1,2j_0}, k_{L-\ell+1,2j_0+1})$. Since there exists the public relation $e(\text{Acc}_{L-\ell+1}, \tilde{V}_0) = e(W, \tilde{P})$ with $\tilde{P} = \tilde{V}_0 \tilde{V}_1^{\mathbf{p}_1} \tilde{V}_2^{\mathbf{p}_2}$, the second part of the proof of knowledge is then $\text{POK}(s, \text{Acc}_{L-\ell+1}, \sigma_{L-\ell+1} : \sigma_{L-\ell+1} = \text{SIGN}(\text{Acc}_{L-\ell+1}, u) \wedge e(\text{Acc}_{L-\ell+1}, \tilde{V}_0) = e(W, \tilde{P}))$.

The signature $\sigma_{L-\ell+1}$ is an ESS+ signature on the message $M = (\text{Acc}_{L-\ell+1}, s, L-\ell+1)$. Thus, $\sigma_{L-\ell+1}$ is composed by the elements $\Sigma_1 = X(\text{Acc}_{L-\ell+1} G_0^a)^c$, $\Sigma_2 = (G_1 G_2^a G_3^b H_1^s H_2^{L-\ell+1})^{\frac{1}{x+c}}$ and $\tilde{\Sigma}_3 = \tilde{H}^c$ where $a, b, c \in_R \mathbb{Z}_p^*$ and $X, G_0, G_1, G_2, G_3, H_1, H_2, \tilde{H}$ are public. These values verify the following relations: $e(\Sigma_2, \tilde{\Sigma}_3 \tilde{Z}) = e(G_1, \tilde{H}) e(G_2, \tilde{H})^a e(G_3, \tilde{H})^b e(H_1, \tilde{H})^s e(H_2, \tilde{H})^{L-\ell+1}$ and $e(\Sigma_1, \tilde{H}) = Y e(\text{Acc}_{L-\ell+1} G_0^a, \tilde{\Sigma}_3)$. The second part of the proof is finally

$$\begin{aligned} & \text{POK}(s, \text{Acc}_{L-\ell+1}, \Sigma_1, \Sigma_2, \tilde{\Sigma}_3 : \\ & e(\text{Acc}_{L-\ell+1}, \tilde{V}_0) = e(W, \tilde{P}) \wedge e(\Sigma_1, \tilde{H}) = Y e(\text{Acc}_{L-\ell+1} G_0^a, \tilde{\Sigma}_3) \wedge \\ & e(\Sigma_2, \tilde{\Sigma}_3 \tilde{Z}) = e(G_1, \tilde{H}) e(G_2, \tilde{H})^a e(G_3, \tilde{H})^b e(H_1, \tilde{H})^s e(H_2, \tilde{H})^{L-\ell+1}). \end{aligned}$$

This proof is generated using standard techniques and the results in [2,1].

What's in a Name?

Evaluating Statistical Attacks on Personal Knowledge Questions

Joseph Bonneau¹, Mike Just², and Greg Matthews²

¹ University of Cambridge

² University of Edinburgh

Abstract. We study the efficiency of statistical attacks on human authentication systems relying on personal knowledge questions. We adapt techniques from guessing theory to measure security against a trawling attacker attempting to compromise a large number of strangers' accounts. We then examine a diverse corpus of real-world statistical distributions for likely answer categories such as the names of people, pets, and places and find that personal knowledge questions are significantly less secure than graphical or textual passwords. We also demonstrate that statistics can be used to increase security by proactively shaping the answer distribution to lower the prevalence of common responses.

1 Introduction

Secret knowledge stored in human memory remains the most widely deployed means of human-computer authentication. It is often referred to as *something you know* in contrast to biometrics (*something you are*) or hardware tokens (*something you have*). While human memory is limited, the high deployment costs of alternatives mean we will continue to rely on it for the foreseeable future.

The most common human-memory systems require recalling data specifically remembered for authentication. Passwords and PINs are the most well-known, but there exist a variety of graphical and textual schemes to aid in recalling secret data [31,29,22,6]. Among other problems, passwords are forgotten frequently enough [31] that many deployed systems also use personal knowledge for backup authentication. In contrast to passwords, personal knowledge questions such as “who was my first-grade teacher?” query facts remembered independently of the system so they are hoped to be recalled successfully when passwords fail.

In the majority of online banking, e-commerce, webmail and social networking websites, users register a question-answer pair on enrolment which can later be used to authorise a password reset. These systems can be no more secure than the difficulty of guessing the answers to these questions. This risk was highlighted in the past year as hackers exploited personal knowledge questions to compromise accounts of politician Sarah Palin and top executives at Twitter.

Despite their ubiquity, personal knowledge questions have received relatively little attention from the security community until recently. User studies have

demonstrated the ability of friends, family, and acquaintances to guess answers correctly [26,13,24], while other research has found some questions used in practice have a tiny set of possible answers [15,25]. Many common questions have also been shown to have answers readily available in public databases or online social networks [18]. For example, at least 30% of Texas residents' mothers' maiden names can be deduced from birth and marriage records [12].

Designers may be able to avoid easily looked-up questions, but it remains an open question as to how secure typical questions are against a *statistical attacker* that attempts to break into a small fraction of anonymous accounts by guessing the most likely answers. While this threat has been briefly touched on in previous research [15,26], we contribute a formal security model based on the information-theoretic model of guessing developed over the past decade. We then examine a range of public statistics that we collected to bound the efficiency of statistical attacks. Our results show most questions to be highly insecure, calling into serious question the continued use of personal knowledge questions.

2 Security Model

2.1 Authentication Protocol

Most deployed systems use personal knowledge questions in a simple challenge-response protocol. The party seeking access, called the *prover* or *claimant*, first sends its identity i to the *Verifier*. The verifier then responds with a challenge question q , to which the prover sends back an answer x . Unlike most challenge-response protocols, the prover's secret knowledge x is usually revealed to the verifier. Replay attacks can be partially addressed by having the verifier include a nonce r along with q , and having the prover respond with $H(x, q, r)$ for some one-way function H . However, an eavesdropper still gains the ability to perform offline search for likely values of x using H as an oracle (and as we shall see, few personal-knowledge questions are resistant to offline search).

Additionally, while the challenge from a verifier is typically a fresh random nonce for cryptographic challenge-response, the set Q of personal knowledge questions registered with the verifier is often very small or even a single question. Some non-traditional question types may increase $|Q|$, such as “preference-based authentication” [14], but the upper limit appears low due to the fundamental requirement of human effort to select and answer questions on enrolment.

Finally, unlike many challenge-response protocols, the verifier must maintain a counter t_i of failed authentication attempts from each prover i to limit the number of guesses an attacker can make. Such a protocol is said to be *online*, in contrast to stateless protocols in which the attacker can make as many guesses as bandwidth allows. Offline protocols rarely use personal knowledge questions due to the difficulty of preventing brute-force attacks, though systems have been proposed for personal password-backup which require simultaneously answering many questions [8,11].

2.2 Threat Model

Our attacker’s goal is to impersonate some legitimate prover i and successfully complete the protocol. The attacker may only desire to gain access on behalf of one specific user in a *targeted* attack, or may be content to gain access on behalf of any user in a *trawling* attack. In the former case, the attacker knows the account i represents some real-world person Peggy, enabling *research* attacks using search engines, online social networks, or public records. An active attacker could conduct more advanced research by dumpster diving, burgling Peggy’s home, or social engineering to trick Peggy into revealing her answer. Targeted attacks may also be performed by somebody who knows Peggy personally. Schechter et al. explored this attack in a laboratory setting and found a high rate of success by acquaintances at guessing personal knowledge questions [26].

Targeted attacks are powerful but do not scale. Trawling attacks, in contrast, require little per-user work and can be used to simultaneously attack many accounts. We assume that a trawling attacker, although they must provide a value for i when initiating the protocol, has no information about the real-world person behind i and must guess answers based on population-wide statistics.

A *blind attacker* guesses without even understanding the question q [15]. This scenario arises if the question is either not transmitted in the clear [21], is transmitted in a CAPTCHA-ised form, or is user-generated and difficult to automatically process.¹ We argue that a more successful attack strategy is to use a weighted combination of answers to likely questions.

An attacker who is able to correctly understand q but not i is a *statistical* attacker (called a *focused* attacker in [15]), whose strategy is to guess the most likely answers to q . Our main goal is to evaluate the security of common questions against statistical attack. While some questions (e.g., “What is my favourite colour?”) obviously have too few plausible answers to be secure, the most common classes of answer found repeatedly in practice are the “proper names” of people, pets, and places, whose security against guessing is not obvious.

The attackers we have identified are not exclusive. While there is a general hierarchy between blind, statistical, and research attacks, an attacker may combine statistics and targeted research. For example, partial knowledge of an account-holder’s identity may enable an attacker to refine her statistical tables (see Section 5).

3 Quantifying Resistance to Guessing

3.1 Mathematical Formulation of Guessing

We now turn to the mathematical problem of quantifying how secure a personal knowledge question q is against guessing. This problem has been previously

¹ Some users may even purposefully obfuscate their questions, such as “What do I want to do?” [15].

considered abstractly [4,23,3,20] and in the case of PINs [2], graphical passwords [6,29,22], and biometrics [1]; we synthesise previous analysis and define new metrics most applicable to trawling attackers.

Because a statistical attacker will respond equally to “what is my boss’ last name?” or “who was my kindergarten teacher?” by guessing common surnames, we seek to measure security of the underlying answer space. We consider the correct answer to be a random variable X drawn from a finite distribution \mathcal{X} which is known to the attacker, with $|\mathcal{X}| = N$ and probability $p_i = P(X = x_i)$ for each possible answer x_i , for $i \in [1, N]$. We assume that \mathcal{X} is arranged as a monotonically decreasing distribution with $p_1 \geq p_2 \geq \dots \geq p_N$. Our attacker’s goal is to guess X using as few queries of the form “is $X = x_i$?” as possible.

Intuitively, we may first think of the *Shannon entropy*

$$H_1(\mathcal{X}) = - \sum_{i=1}^N p_i \lg p_i \quad (1)$$

as a measure of the “uncertainty” of X . Introduced by Claude Shannon in 1948, entropy has entered common cryptographic parlance as a measure of security, with “high-entropy” secrets being considered advantageous [8,11]. As has been argued previously [4,23,3,20,2,6,1], H_1 is a poor estimator of guessing difficulty for security purposes, as it quantifies the average number of subset membership queries of the form “Is $X \in \mathcal{S}$?” for arbitrary subsets $\mathcal{S} \subseteq \mathcal{X}$.²

Because cryptographic protocols are specifically designed to require sequential guessing, a better metric is the expected number of attempts required to correctly guess X if the attacker takes up the obvious strategy of guessing each possible event in order of its likeliness, known as the *guessing entropy*:

$$G(\mathcal{X}) = E \left[\#_{\text{guesses}}(X \xleftarrow{\text{R}} \mathcal{X}) \right] = \sum_{i=1}^N p_i \cdot i \quad (2)$$

This measure was introduced by Massey [20] and later named by Cachin [4].

3.2 Marginal Guessing

Guessing entropy models an attacker who will never give up in her search, and thus it can be skewed by exceedingly unlikely events. A simple thought experiment demonstrates why this is inadequate for our purposes. Suppose Eve must sequentially guess k challenge questions with answers drawn from \mathcal{X} . Some questions will have uncommon answers, and Eve must make $\sim k \cdot G(\mathcal{X})$ guesses.

Now consider a second adversary Mallory whose goal is to guess the answers to k questions from a set of $m > k$ total questions. Her optimal strategy is to first guess the most likely value for each question in sequence, then the second-most

² The proof of this is a straightforward consequence of Shannon’s source coding theorem. Symbols $X \xleftarrow{\text{R}} \mathcal{X}$ can be encoded using a Huffman code with average bit length $\leq H_1(\mathcal{X}) + 1$, and the adversary can learn one bit at a time with set queries.

likely value for each question, and so on. Mallory’s efficiency will greatly increase as m increases, as she may never need to guess uncommon answers. Guessing entropy is inadequate as it doesn’t account for Mallory’s willingness to give up on the questions which have less probable answers.

To bound an attacker who only requires some probability α of guessing correctly, we define the *marginal guesswork* μ_α :

$$\mu_\alpha(\mathcal{X}) = \min \left\{ j \in [1, N] \mid \sum_{i=1}^j p_i \geq \alpha \right\} \quad (3)$$

This function, introduced by Pliam [23], is also referred to as the α -*work-factor*. We define a similar metric λ_β , the *marginal success rate*, slightly adapted from Boztaş [3], as the probability of success after β guesses have been made:

$$\lambda_\beta(\mathcal{X}) = \sum_{i=1}^{\beta} p_i \quad (4)$$

3.3 Effective Key Length Metrics

While it is important to remember that μ_α and λ_β are not measures of entropy, we nonetheless find it convenient to convert them into units of bits. This makes all the metrics H_1 , G , μ_α and λ_β directly comparable and has an intuitive interpretation as (logarithmically-scaled) attacker workload. We convert each metric by calculating the logarithmic size of a discrete uniform distribution \mathcal{U}_N of size $|\mathcal{U}_N| = N$ with $p_i = \frac{1}{N}$ for all $1 \leq i \leq N$, which has the same value of the guessing metric. This can be thought of as the “effective key length” as it represents the size of a randomly-chosen cryptographic key which would give equivalent security. The guessing entropy of \mathcal{U}_N is:

$$G(\mathcal{U}_N) = \sum_{i=1}^N p_i \cdot i = \frac{1}{N} \sum_{i=1}^N i = \frac{1}{N} \cdot \frac{N(N+1)}{2} = \frac{N+1}{2}$$

The entropy of this distribution is $\lg N$, so given the guessing entropy of an arbitrary distribution $G(\mathcal{X})$ we can find the logarithmic size of a uniform distribution with equivalent guessing entropy as:

$$\tilde{G}(\mathcal{X}) = \lg[2 \cdot G(\mathcal{X}) - 1] \quad (5)$$

The quantity $\tilde{G}(\mathcal{X})$ can then be interpreted as the effective key length of \mathcal{X} with respect to guessing entropy. We can similarly derive formulas for effective key length with respect to marginal guesswork and marginal success rate:

$$\tilde{\mu}_\alpha(\mathcal{X}) = \lg \left(\frac{\mu_\alpha(\mathcal{X})}{\alpha} \right) \quad \tilde{\lambda}_\beta(\mathcal{X}) = \lg \left(\frac{\beta}{\lambda_\beta(\mathcal{X})} \right) \quad (6)$$

Example Calculation. Consider a distribution \mathcal{Z} with $P_{\mathcal{Z}} = \{\frac{1}{3}, \frac{1}{18}, \frac{1}{18}, \frac{1}{18}, \dots\}$. Regardless of the tail probabilities, an attacker will have a 50% chance of successfully guessing a random variable drawn from \mathcal{Z} after 4 attempts, so $\lambda_4(\mathcal{Z}) = \frac{1}{2}$. The distribution \mathcal{U}_8 with eight equally likely events would also have $\lambda_4(\mathcal{U}_8) = \frac{1}{2}$, so these two distributions are equivalent with respect to λ_4 . Since $\lg |\mathcal{U}_8| = \lg 8 = 3$, we expect $\tilde{\lambda}_4(\mathcal{Z}) = 3$, and we can verify that by our formula:

$$\tilde{\lambda}_4(\mathcal{Z}) = \lg \left(\frac{4}{\lambda_4(\mathcal{Z})} \right) = \lg \left(\frac{4}{\frac{1}{2}} \right) = \lg 8 = 3$$

3.4 Relationship between Metrics

A natural question is whether $\tilde{\mu}_\alpha$ and $\tilde{\lambda}_\beta$ are bounded by H_1 or \tilde{G} ; unfortunately this is not the case. The following theorems demonstrate the fundamental incomparability of entropy, guessing entropy, and marginal guesswork:

Theorem 1. (Pliam) Given any $m > 0$, $\beta > 0$ and $0 < \alpha < 1$, there exists a distribution \mathcal{X} such that $\tilde{\mu}_\alpha(\mathcal{X}) < H_1(\mathcal{X}) - m$ and $\tilde{\lambda}_\beta(\mathcal{X}) < H_1(\mathcal{X}) - m$.

Theorem 2. (Boztaş) Given any $m > 0$, $\beta > 0$ and $0 < \alpha < 1$, there exists a distribution \mathcal{X} such that $\tilde{\mu}_\alpha(\mathcal{X}) < \tilde{G}(\mathcal{X}) - m$ and $\tilde{\lambda}_\beta(\mathcal{X}) < \tilde{G}(\mathcal{X}) - m$.

Theorem 3. (new) Given any $m > 0$, $\alpha_1 > 0$, and $\alpha_2 > 0$ with $0 < \alpha_1 < \alpha_2 < 1$, there exists a distribution \mathcal{X} such that $\tilde{\mu}_{\alpha_1}(\mathcal{X}) < \tilde{\mu}_{\alpha_1}(\mathcal{X}) - m$.

The first two results were demonstrated previously [23,3] but we combine the proof techniques here to prove both at once. We construct a pathological distribution \mathcal{X} with one likely event and many very-unlikely events. We set $p_1 = \frac{1}{2}$ and $p_i = \frac{1}{2^{2m+4}}$ for the remaining symbols ($|\mathcal{X}| = 2^{2m+3} + 1$). This gives $H_1(\mathcal{X}) > m + 3$ and $\tilde{G}(\mathcal{X}) > m + 1$, following from Massey's proof that \tilde{G} is bounded from below by $(H_1 - 2)$ [20]. But $\tilde{\mu}_{\frac{1}{2}}(\mathcal{X}) = \tilde{\lambda}_1(\mathcal{X}) = 1$, proving the theorem. Note that this construction requires $|\mathcal{X}| \in \Theta(4^m)$, the result does not hold if we impose limits on $|\mathcal{X}|$.

The third theorem, a new result, is proved similarly by setting $p_1 = \alpha_1$ and $p_n = \frac{1}{(\alpha_2 - \alpha_1) \cdot 2^m}$ for all $n > 1$. This gives $\tilde{\mu}_{\alpha_1} = \lg \left(\frac{1}{\alpha_1} \right) = -\lg \alpha_1$, but $\tilde{\mu}_{\alpha_2} = \lg \left(\frac{2^m + 1}{\alpha_2} \right) > \lg \left(\frac{2^m}{\alpha_2} \right) = m - \lg \alpha_1$, giving the desired gap m with $|\mathcal{X}| \in \Theta(2^m)$.

These results demonstrate that no measure is adequate for all security purposes, but that context-specific $\tilde{\mu}_\alpha$ and $\tilde{\lambda}_\beta$ must be used which reflect only the values in the distribution likely to be guessed. A highly-skewed distribution like human names might have high H and \tilde{G} can be very easy to guess despite having many unlikely events which inflate its apparent security.

3.5 Applicability to Personal Knowledge Questions

Assuming that a targeted attacker is likely to use victim-specific research, we are most concerned with a trawling attacker who will never guess uncommon

answers, simply trying a new target if common answers fail. The most useful metric we have is the marginal success rate λ_β . Assuming the system imposes a limit of t_{\max} incorrect guesses for each account, the critical value is the fraction of accounts the attacker can expect to compromise, which is $\lambda_{t_{\max}}$. In the limit of an attacker trying only the single most likely answer for multiple accounts, our security is $\tilde{\lambda}_1(\mathcal{X}) = -\lg(p_1)$, which is also called the min-entropy $H_\infty(\mathcal{X})$.

For offline attacks, λ_β is less meaningful because an attacker won't limit their guessing nearly as much. In this case, $\tilde{\mu}_{\frac{1}{2}}$ is a reasonable metric in that it avoids \tilde{G} 's dependence on very unlikely events, while still measuring the cost for an attacker to compromise a majority of available accounts.

3.6 Estimation from Statistics

A final subtlety is estimating our metrics from publicly available statistics based on random sampling from \mathcal{X} and not on complete knowledge of the distribution. This, too, strongly favours the use of μ_α and λ_β because they only reflect the most likely events and are not affected by large uncertainty on the tail probabilities of \mathcal{X} . Estimating μ_α and λ_β from a statistical sample is straightforward: we simply take the most likely events from the sample and use them to compute our metric. It is possible to compute a p -confidence interval for μ_α or λ_β by computing p -confidence intervals for each individual event probability, and using all of the minimum (eq. maximum) estimates to compute minimum estimates μ_α^- and λ_β^- (eq. μ_α^+ and λ_β^+). This technique strictly overestimates uncertainty, but in practice we've found most of the statistics which influence μ_α or λ_β have strong enough statistical support that the confidence interval is quite tight.³

In contrast, since H_1 and \tilde{G} depend on the entire distribution, they are much more difficult to reliably estimate from statistics. If we don't a priori know $|\mathcal{X}|$, it is impossible to provide any upper bound because we cannot know the number of events which haven't been observed by sampling. As a lower bound for security purposes, we simply assume no unobserved events exist.

A second problem is that unlikely events are often suppressed for privacy or brevity in published census data. Again in the name of a lower bound, we simply take the least-likely observed event and insert copies of it until the probability space is filled. In the case of surname data, for instance, which is given exactly for names shared by at least k people but suppressed for less common names, we repeatedly insert fictitious names shared by k people until the data set contains as many people as the target population. This crude approximation lowers our estimates of H_1 and \tilde{G} , but doesn't influence μ_α or λ_β .

³ Indeed, for $\alpha \leq \frac{1}{2}$ and $\beta < \frac{N}{2}$ we are always able to calculate μ_α and $\tilde{\lambda}_\beta$ to within 0.1 bit with $p > 99\%$. We expect errors from divergence between the population distribution and answers which humans actually choose to use to be so much greater than sampling error that we ignore it in the remainder of this paper.

4 Information Sources

4.1 Question Types and Their Use

Based upon recent research into deployed personal knowledge authentication systems, we focus our analysis on questions that ask for proper names, as summarised in Table 1. Rabkin collected 216 questions used by 11 financial institutions [25], and Schechter et al. collected 29 questions used for webmail services provided by AOL, Google, Yahoo!, and Microsoft [26]. These provide some hints at the type of questions used—Rabkin found approximately $\frac{1}{3}$ soliciting a person’s name and $\frac{1}{5}$ asking for place names, while Schechter et al. found $\frac{1}{4}$ soliciting a person’s name and $\frac{1}{6}$ asked for a place name. Unfortunately, this research provides no insight as to which questions users actually select. For example, relatively few questions asked for pet names, though this may be because there is only one way to phrase this question and not because it is unpopular.

Just and Aspinall collected approximately 500 user-generated challenge questions and categorised these questions into a small number of types [15], which we consider to be a more insightful data set. Most notably, they found that 34% of user questions asked for a human name, 15% asked for a pet name and 20% asked for a place name. Of the remainder, 22% asked for a user’s favourite item amongst films, singers, car brands, etc., 5% asked for a time, date, or number, and the remainder were ambiguous. Thus, we estimate that a few simple categories of proper names cover roughly 70% of real-world questions. The remainder, many of which ask for the user’s “favorites,” appear trivially vulnerable to guessing attacks and we ignore them in our study.

One subtlety with name data is that it is not always clear if users will respond with a forename (also called a ‘first name’ or ‘given name’), surname (also ‘last name’), or both. In such cases, a statistical attacker can simply estimate what probability of users will respond with which, and then combine the two probability distributions, scaling each by its sampling frequency. This should slow down attacks by no more than a factor of two. We also assume that middle names (though less commonly asked for) are reasonably approximated by forenames. In reality, middle names probably have slightly higher diversity, but the most common names are likely the same and an attacker can use a forename table in an attack without much slowdown.

4.2 Data Collection

To our knowledge, this is the first time a breadth of data has been collected for analysing personal knowledge questions. We collected data from government sources where possible, as many developed nations keep near-complete records of citizens’ names. In some cases the data is not made publicly available but is acquired and published by media organisations, as in the case of pet registration lists which are compiled by smaller local government bodies. We were also able to gather school and city data from official sources.

Table 1. Common answer categories

Category	Example Questions
<i>Forename</i>	What is your grandfather’s first name?
	What is your father’s middle name?
<i>Surname</i>	What is your mother’s maiden name?
	What was the last name of your favourite school teacher?
<i>General Name</i>	Who was your childhood best friend?
	What was your first pet’s name?
<i>Place</i>	In what city were you born?
	Where did you go for your honeymoon?
<i>Other</i>	What is the name of your high school?
	What was your grandfather’s occupation?
	What is your favourite movie?

Official sources often omit items occurring less than some minimum threshold. As mentioned in Section 3.6, we used estimates of the total population to overcome the missing data. A complete list of our data sources, as well as scripts used for calculations on the data, is made available on our project website.⁴ We also provide a summarised list of official sources used in Appendix A.

We found no official sources which provide lists of full names, so we collected names from 269 million randomly-crawled public profiles on the popular online social network Facebook. The demographic for this data is less clearly delineated, but can be used to roughly approximate the global Internet user population.

5 Results and Discussion

Our calculations of the metrics defined in Section 3 are displayed in Table 2.⁵ For online attacks, the marginal success rate $\tilde{\lambda}_3$ models an attacker limited to 3 guesses on each available account. For almost all of our data (exclusive of full names and primary schools), we have $\tilde{\lambda}_3 \lesssim 8$, indicating that the majority of deployed challenge questions systems are insecure against trawling attackers. If 3 guesses are allowed, an attacker can compromise roughly 1 in 80 accounts. This may even be an overestimate of security: the most extreme trawling attacker will make only 1 guess per account, represented by $\tilde{\lambda}_1 = H_\infty$.

For offline attacks, we mostly find $\tilde{\mu}_{\frac{1}{2}} \lesssim 12$, meaning an attacker can compromise the majority of accounts with only a few thousand guesses per account.

Our analysis demonstrates *weak subspaces* in the answer distribution for most personal knowledge questions which can be directly compared to weak answer spaces found in other authentication systems. In Figure 1 we plot the Facebook name distributions against textual passwords [16,28,27], mnemonic

⁴ <http://groups.inf.ed.ac.uk/security/KBA/>

⁵ We also include the Rényi entropy $H_\alpha(\mathcal{X}) = \frac{1}{1-\alpha} \lg \left(\sum_{i=1}^N p_i^\alpha \right)$ for $\alpha \in \{0, 2, \infty\}$.

As predicted by Boztaş [3], H_2 seems to provide a good estimate for $\tilde{\mu}_{\frac{1}{2}}$.

Table 2. Summary of statistics on real data

Source	H_0	H_1	\tilde{G}	H_2	$\tilde{\mu}_{\frac{1}{2}}$	$\tilde{\lambda}_3$	H_∞	x_1
Full Names								
Facebook	26.9	25.2	26.0	21.1	24.3	14.0	13.9	John Smith
Surnames								
South Korea	7.5	4.6	4.5	3.5	3.3	2.7	2.2	Kim
Chile	6.8	6.6	6.3	6.3	6.0	4.9	4.5	González
Spain	9.6	8.9	9.1	7.6	8.8	5.4	5.0	Garcia
Japan	14.5	11.3	12.0	9.0	9.2	6.2	6.0	Satō
Finland	13.8	12.2	12.3	10.5	10.5	7.9	7.8	Virtanen
England	17.4	13.3	14.6	10.2	11.0	6.7	6.4	Smith
Estonia	11.9	11.7	11.7	11.3	11.6	7.9	7.6	Ivanov
Australia	18.6	14.1	15.3	10.9	11.8	7.4	6.8	Smith
Norway	13.7	12.5	13.0	9.9	11.9	6.5	6.4	Hansen
USA	19.1	14.9	16.9	10.9	12.3	7.2	6.9	Smith
Facebook (Sp.)	20.3	13.5	16.6	9.5	10.4	6.4	6.4	Rodriguez
Facebook	23.1	16.6	19.2	12.3	14.0	8.3	8.0	Smith
Forenames, Mixed								
Spain	9.7	9.0	8.9	8.1	7.9	6.0	5.9	Jose
Iceland	8.9	8.5	8.3	7.9	7.7	5.9	5.8	Jón
Belgium	15.0	10.2	10.3	8.8	8.7	6.1	5.7	Maria
USA	16.7	11.2	14.0	8.7	8.6	6.2	5.9	Michael
Facebook (Sp.)	19.5	11.4	14.9	8.5	8.7	5.8	5.7	Maria
Facebook	22.7	13.6	17.8	10.7	10.7	7.6	7.5	David
Forenames, Female (♀)								
Spain	8.3	7.9	7.8	7.3	7.1	5.3	5.1	Maria
Iceland	7.9	7.5	7.3	6.9	6.8	5.1	4.9	Guðrún
Belgium	15.2	10.1	10.9	8.1	8.2	5.5	4.9	Maria
USA	15.1	10.9	12.9	8.7	8.3	6.5	6.3	Jennifer
Forenames, Male (♂)								
Spain	8.6	7.8	7.8	6.9	6.6	4.9	4.8	Jose
Iceland	7.9	7.5	7.3	6.9	6.8	5.0	4.8	Jón
USA	15.2	9.4	12.0	7.2	6.9	5.2	5.0	Michael
Belgium	15.0	9.7	10.4	8.2	7.8	6.1	5.7	Jean
Forenames by Birth Decade								
USA, 1950 (♀)	11.8	8.6	9.1	7.1	6.8	5.2	5.0	Mary
USA, 1950 (♂)	11.7	7.7	8.3	6.2	5.8	4.6	4.6	James
USA, 1960 (♀)	11.9	9.1	9.5	7.6	7.1	5.6	5.2	Lisa
USA, 1960 (♂)	11.9	7.9	8.6	6.4	5.9	4.7	4.6	Michael
USA, 1970 (♀)	12.1	9.7	10.3	7.7	7.6	5.5	4.8	Jennifer
USA, 1970 (♂)	12.1	8.4	9.3	6.7	6.3	5.0	4.6	Michael
USA, 1980 (♀)	12.2	9.7	10.4	7.7	7.6	5.4	5.3	Jessica
USA, 1980 (♂)	12.2	8.6	9.6	6.9	6.4	5.1	4.9	Michael
USA, 1990 (♀)	12.3	10.3	10.8	8.4	8.3	6.1	6.0	Jessica
USA, 1990 (♂)	12.3	9.3	10.0	7.5	7.1	5.7	5.5	Michael
USA, 2000 (♀)	12.4	10.8	11.1	9.1	9.0	6.6	6.5	Emily
USA, 2000 (♂)	12.2	9.9	10.4	8.2	7.8	6.4	6.2	Jacob
Pet Names								
Los Angeles	15.8	11.7	13.1	9.2	9.4	6.5	6.4	Lucky
Des Moines	13.6	11.6	12.4	9.4	9.7	6.5	6.2	Buddy
San Francisco	13.7	11.6	12.0	9.6	9.8	6.7	6.7	Buddy
Place Names								
UK Primary Schools	14.0	13.8	13.5	13.6	13.3	12.1	12.1	Essex
UK High Schools	8.7	8.5	8.2	8.3	8.0	7.4	7.3	Holyrood
School Mascots (US)	11.8	8.1	9.3	6.2	5.7	4.5	4.1	Eagles
UK Cities	9.2	8.5	8.8	5.9	8.7	4.4	3.0	London
Tourist Destinations	13.0	12.0	12.5	9.5	12.4	6.3	5.9	London

passwords [17], the Pass-Go user-drawn password system [22], the Passfaces graphical PIN system [29], the PassPoints visually-cued clicked password system [6] and a handwriting-recognition biometric system [1]. We also include a recently-leaked dataset of 32 M passwords from the gaming website rockyou.com. Aside from the badly-broken Passfaces system, personal knowledge questions compare unfavorably to other methods unless full names are required.

We summarise further interesting trends below:

Diversity effects. The difficulty of guessing surnames correlates with ethnic diversity. American surnames were the most difficult to guess in our survey, presumably because the population is a blend of immigrants from many ethnicities. Facebook provides even more diversity as a blend of users from around the world. Surnames from Japan and South Korea, which are ethnically homogeneous and have relatively few immigrants, provide low resistance to guessing.

Naming trends. Given names are a matter of fashion and vary in several interesting dimensions. In the countries studied, female names seem to provide slightly higher resistance to guessing than male names.⁶ Over the past 6 decades in the USA, diversity of forenames has been increasing slowly but steadily. Curiously, pet names are slightly harder to guess than human names.

Ethnic correlations. The Facebook data provides ample evidence that forenames and surnames are not independent variables. They are correlated via an individual's ethnicity and possibly further in that some name combinations are considered more pleasing to the ear. Maria Gonzalez and Jose Rodriguez are the most statistically over-represented names in our data set given the independent frequency of the forename and surname component. Each appears with extremely high statistical significance ($p \ll 0.001$ in a χ^2 test). Similarly, there are a number of highly statistically under-represented name pairs, mostly curious cross-cultural pairings like Francesco Smith or Juan Khan. Frequent names like Maria Gonzalez appear because both components share a common ethnicity (Hispanic). A χ^2 test on the entire forename distribution given a Spanish surname such as Gonzalez confirms with high significance ($p \ll 0.001$) that naming patterns change amongst individuals of this ethnicity.

This dependence between forenames and surnames indicates that guessing difficulty will be lower if an attacker knows the target's ethnicity. To quantify this, we clustered the names and identified a set of 250 common Spanish surnames, which cover 10.1% of all individuals in the dataset. The guessing difficulty for these 4 million individual's forenames is shown in Table 2 under "Facebook (Sp.)".⁷ We similarly took 250 common Spanish forenames, representing nearly 22 million people, and computed the guessing difficulty of their surnames. In both cases $\hat{\mu}_{\frac{1}{2}}$ and λ_3 drop by about a bit, indicating that identifying an individual's ethnicity may roughly double a statistical attacker's efficiency.

⁶ Security increases, of course, if a question doesn't specify gender.

⁷ Note that this is not the difficulty of guessing a typical Spanish forename, it is the difficulty of guessing the forename of a person with a typically Spanish surname.

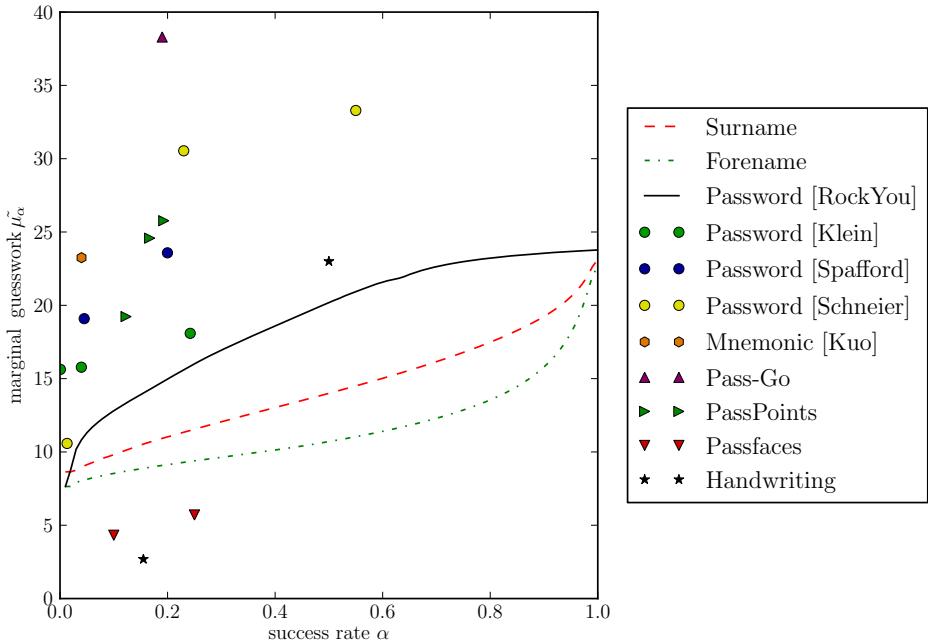


Fig. 1. Comparison of weak subspaces in name distributions (Facebook dataset) to those found in other authentication systems [16,28,27,29,22,6,1]

Power-law models. The frequencies of English surnames have previously been posited to be well-fitted by a discrete Pareto distribution [10], with the probability that a surname X 's frequency is $f(X)$ is greater than x being proportional to $x^{-(c+1)}$. Fox et al. found this to hold for $c \approx 1.4$. This is thought to occur because surnames are inherited but don't strongly correlate to reproductive fitness, leading to a Pareto-like distribution through random genetic drift.

We found the Pareto distribution with $c \approx 1$ to be a reasonable model for the Facebook surname dataset, though the head of the distribution skewed significantly away from the Pareto model, with the most common names being less popular than expected. Still, support for a power-law model of surname frequency suggests the inappropriateness of this distribution for security purposes.

Interestingly, our forename and pet name distributions were also approximated well by the Pareto distribution, with $c \approx 0.8$ in the Facebook data set. The reasons for this fit are less well-understood, though this is close to the classic Zipf distribution ($c = 1$) which is known to model many naturally-occurring phenomena such as word frequency in natural languages. If it is true that humans naturally produce names following the Zipf distribution, this too suggests that human-provided name spaces will not provide adequate guessing resistance.

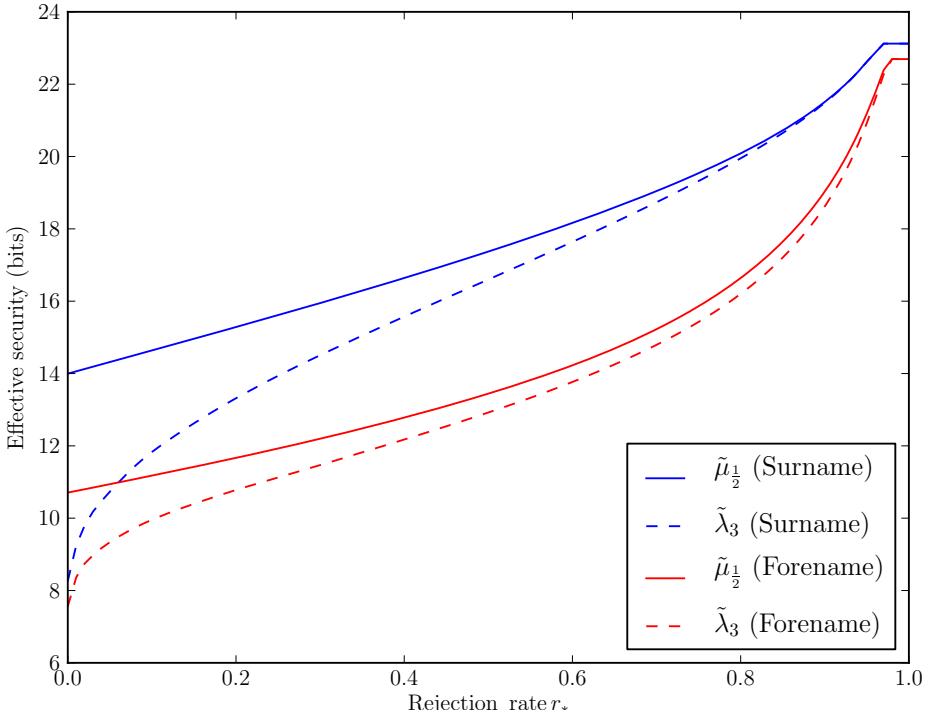


Fig. 2. Effectiveness of shaping a distribution as a function of r_*

6 Countermeasures

Up to this point, we have assumed a passive enrolment server which accepts any answers and has no influence on the resulting answer distribution \mathcal{X} . If we assume the server knows \mathcal{X} , it is possible to actively *shape* the answer space into a more secure distribution \mathcal{X}' by probabilistically rejecting some users' answers. There is a growing literature on proactively encouraging users to select diverse textual [9] or graphical [5] passwords, Bentley et. al previously considered the problem of “grooming” a skewed probability distribution to uniform [2].

The process of a user answering is equivalent to randomly drawing $X \xleftarrow{R} \mathcal{X}$. The server can examine the result and if $X = x_i$, reject with probability r_i and force the user to answer a differently-worded question with the same answer-space, in practice re-drawing $X \xleftarrow{R} \mathcal{X}$. We assume the process is recursive: the user's second answer x_j may also be rejected with some probability r_j . This process results in a modified distribution \mathcal{X}' of answers which are accepted.

If we are constrained by a maximum-allowable overall rejection probability r_* , it is simple to find the optimum rejection probabilities r_1, \dots, r_N which will most increase security. This comes from the observation that, given the ability to lower any single p_i by any fixed Δ , lowering p_1 will result in the greatest increase for each of H_α , \tilde{G} , $\tilde{\mu}_\alpha$ and $\tilde{\lambda}_\beta$. The optimal r_1, \dots, r_N are thus computed

by an iterative algorithm. First r_1 is increased until $p'_1 = p'_2$, namely by setting $r_1 = 1 - \frac{p_2}{p_1}$. Next, we increase r_1 and r_2 together until $p'_1 = p'_2 = p'_3$. We repeatedly increase r_1 through r_m so that $p'_1 = \dots = p'_{m+1}$, stopping when $r_* = \sum_{i=0}^m r_i \cdot p_i$ and we have reached our maximum overall rejection probability.⁸ The m most likely events are equiprobable in \mathcal{X}' . The remaining events are never rejected; their probabilities each increase by $\frac{1}{1-r_*}$.

Shaping is very effective at increasing $\tilde{\lambda}_\beta$ as the most likely events are greatly reduced in probability. As shown in Figure 2, shaping the name distributions in the Facebook corpus drives $\tilde{\lambda}_3$ close to $\tilde{\mu}_{\frac{1}{2}}$ even for reasonable $r_* < 0.5$. Even relatively mild shaping with $r_* = 0.1$ increases $\tilde{\lambda}_3$ by 3.6 bits for surnames. Although the overall rejection rate is low, though, it is highly unequal: for $r_* = 0.1$ the rejection rate r_1 for the surname “Smith” is 94.3%.

7 Concluding Remarks

We have applied marginal guessing metrics to the security analysis of common personal knowledge questions. We then used a diverse collection of real-world statistical data to estimate the strength of these questions against a trawling attacker with a large number of accounts to test. We believe this is an increasingly important attacker model and our methods provide a useful framework for evaluating human-computer authentication.

We have not assessed a ground-truth answer space; the actual distribution of surnames provided to a deployed authentication server will vary based on the precise question wording and specific user population. Still, we have found strong evidence that across a broad range of cultures and contexts, human-created names simply don't have enough diversity to provide serious resistance to guessing attacks. In combination with recent results demonstrating vulnerability to targeted attacks, our work casts serious doubt on the continued use of personal knowledge questions for backup authentication.

Acknowledgements

We thank our anonymous referees and our shepherd Lucas Ballard for their detailed and helpful comments. We also thank David Aspinall, Claudia Diaz, Andrew Lewis, and Hyoungshick Kim for assistance drafting our report. Just and Matthews were funded by UK EPSRC, Grant No. EP/G020760/1.

References

- Ballard, L., Kamara, S., Reiter, M.K.: The Practical Subtleties of Biometric Key Generation. In: SS 2008: Proceedings of the 17th Conference on Security, Berkeley, CA, USA, pp. 61–74. USENIX Association (2008)

⁸ The algorithm may terminate early if the distribution has reached uniformity, though this is probably impractical. For example, the Facebook surnames corpus requires a rejection rate of 95.5% to be shaped to uniformity.

2. Bentley, J., Mallows, C.: How Much Assurance Does a PIN Provide? In: Baird, H.S., Lopresti, D.P. (eds.) HIP 2005. LNCS, vol. 3517, pp. 111–126. Springer, Heidelberg (2005)
3. Boztas, S.: Entropies, Guessing, and Cryptography. Technical Report 6, Department of Mathematics, Royal Melbourne Institute of Technology (1999)
4. Cachin, C.: Entropy measures and unconditional security in cryptography. PhD thesis, ETH Zürich (1997)
5. Chiasson, S., Forget, A., Biddle, R., van Oorschot, P.C.: Influencing Users Towards Better Passwords: Persuasive Cued Click-Points. In: BCS-HCI 2008: Proceedings of the 22nd British HCI Group Annual Conference on HCI 2008, Swinton, UK, UK, pp. 121–130. British Computer Society (2008)
6. Davis, D., Monroe, F., Reiter, M.K.: On User Choice in Graphical Password Schemes. In: SSYM 2004: Proceedings of the 13th Conference on USENIX Security Symposium, Berkeley, CA, USA, p. 11. USENIX Association (2004)
7. Dragomir, S.S., Boztas, S.: Some estimates of the average number of guesses to determine a random variable. In: Proceedings of the 1997 IEEE International Symposium on Information Theory, p. 159 (1997)
8. Ellison, C., Hall, C., Milbert, R., Schneier, B.: Protecting Secret Keys with Personal Entropy. Future Gener. Comput. Syst. 16(4), 311–318 (2000)
9. Forget, A., Chiasson, S., van Oorschot, P.C., Biddle, R.: Improving Text Passwords Through Persuasion. In: SOUPS 2008: Proceedings of the 4th Symposium on Usable Privacy and Security, pp. 1–12. ACM, New York (2008)
10. Fox, W.R., Lasker, G.W.: The Distribution of Surname Frequencies. International Statistical Review, 81–87 (1983)
11. Frykholm, N., Juels, A.: Error-tolerant password recovery. In: CCS 2001: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 1–9. ACM, New York (2001)
12. Griffith, V., Jakobsson, M.: Messin' with Texas: Deriving Mother's Maiden Names Using Public Records. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 91–103. Springer, Heidelberg (2005)
13. Haga, W.J., Zviran, M.: Question-and-answer passwords: an empirical evaluation. Inf. Syst. 16(3), 335–343 (1991)
14. Jakobsson, M., Yang, L., Wetzel, S.: Quantifying the Security of Preference-based Authentication. In: DIM 2008: Proceedings of the 4th ACM Workshop on Digital Identity Management, pp. 61–70. ACM, New York (2008)
15. Just, M., Aspinall, D.: Personal choice and challenge questions: A security and usability assessment. In: Cranor, L. (ed.) SOUPS, ACM International Conference Proceeding Series. ACM, New York (2009)
16. Klein, D.: “Foiling the Cracker”: A Survey of, and Improvements to, Password Security. In: Proceedings of the 2nd USENIX Security Workshop, pp. 5–14 (1990)
17. Kuo, C., Romanosky, S., Cranor, L.F.: Human Selection of Mnemonic Phrase-based Passwords. In: SOUPS 2006: Proceedings of the Second Symposium on Usable Privacy and Security, pp. 67–78. ACM, New York (2006)
18. Lindamood, J., Kantarcioglu, M.: Inferring Private Information Using Social Network Data. Technical Report UTDCS-21-08, University of Texas at Dallas Computer Science Department (July 2008)
19. Malone, D., Sullivan, W.G.: Guesswork and Entropy. In: Proceedings of the 2004 IEEE International Symposium on Information Theory, vol. 50 (2004)
20. Massey, J.L.: Guessing and Entropy. In: Proceedings of the 1994 IEEE International Symposium on Information Theory, p. 204 (1994)

21. O'Gorman, L., Bagga, A., Bentley, J.L.: Call Center Customer Verification by Query-Directed Passwords. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 54–67. Springer, Heidelberg (2004)
22. van Oorschot, P.C., Thorpe, J.: On Predictive Models and User-Drawn Graphical Passwords. ACM Trans. Inf. Syst. Secur. 10(4), 1–33 (2008)
23. Pliam, J.O.: On the Incomparability of Entropy and Marginal Guesswork in Brute-Force Attacks. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 67–79. Springer, Heidelberg (2000)
24. Pond, R., Podd, J., Bunnell, J., Henderson, R.: Word Association Computer Passwords: The Effect of Formulation Techniques on Recall and Guessing Rates. Computers & Security 19(7), 645–656 (2000)
25. Rabkin, A.: Personal knowledge questions for fallback authentication: Security questions in the era of Facebook. In: Cranor, L.F. (ed.) SOUPS, ACM International Conference Proceeding Series, pp. 13–23. ACM, New York (2008)
26. Schechter, S., Brush, A.J.B., Egelman, S.: It's no secret: Measuring the security and reliability of authentication via 'secret' questions. In: IEEE Security and Privacy. IEEE, Los Alamitos (2009)
27. Schneier, B.: Real-world passwords (December 2006)
28. Spafford, E.: Observations on Reusable Password Choices. In: Proceedings of the 3rd USENIX Security Workshop (1992)
29. Thorpe, J., van Oorschot, P.C.: Human-Seeded Attacks and Exploiting Hot-Spots in Graphical Passwords. In: SS 2007: Proceedings of 16th USENIX Security Symposium, Berkeley, CA, USA. USENIX Association (2007)
30. Toomim, M., Zhang, X., Fogarty, J., Landay, J.A.: Access Control by Testing for Shared Knowledge. In: Czerwinski, M., Lund, A.M., Tan, D.S. (eds.) CHI, pp. 193–196. ACM, New York (2008)
31. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password Memorability and Security: Empirical Results. IEEE Security and Privacy Magazine 2(5), 25 (2004)

A Sources of Statistical Data

Below is a summary of statistical data sources used in compiling this paper. Complete information on the data sets is provided on our project website <http://groups.inf.ed.ac.uk/security/KBA/>.

- Chile Civil Identification and Registration Service
- Des Moines Register
- Eesti Ekspress
- Euromonitor International
- Finland Population Register Center
- Intellectual Property Australia
- Japanese Surname Dictionary
- Los Angeles Department of Animal Licensing
- San Francisco Animal Licensing Department
- Scottish Government School Education Statistics
- Spanish National Institute of Statistics
- Statistics Belgium
- Statistics Iceland
- Statistics Korea
- Statistics Norway
- United Kingdom Department for Children, Schools, and Families
- United Kingdom Office for National Statistics
- United States Census Bureau
- United States Social Security Administration

Cryptographic Protocol Analysis of AN.ON

Benedikt Westermann¹, Rolf Wendolsky²,
Lexi Pimenidis³, and Dogan Kesdogan^{1,4}

¹ Q2S*, NTNU, 7491 Trondheim, Norway

² JonDos GmbH, 93055 Regensburg, Germany

³ iDev GmbH, 50672 Cologne, Germany

⁴ Chair for IT Security, FB5, University of Siegen, 57068 Siegen, Germany

Abstract. This work presents a cryptographic analysis of AN.ON’s anonymization protocols. We have discovered three flaws of differing severity. The first is caused by the fact that the freshness of the session key was not checked by the mix. This flaw leads to a situation where an external attacker is able to perform a replay attack against AN.ON. A second, more severe, error was found in the encryption scheme of AN.ON. An internal attacker controlling the first mix in a cascade of length two is able to de-anonymize users with high probability. The third flaw results from the lack of checks to ensure that a message belongs to the current session. This enables an attacker to impersonate the last mix in a cascade.

The flaws we discovered represent errors that, unfortunately, still occur quite often and show the importance of either using standardized cryptographic protocols or performing detailed security analyses.

1 Introduction

In recent years anonymous communications have become an important building block for privacy-preserving systems. Anonymous channels are often an unconditional requirement for e-voting, e-health or anonymous credential systems. Many techniques have been proposed in theory, for example Tarzan[1] or MorphMix[2]. However, only a few systems have been widely deployed. In terms of number of users, the two major deployed anonymization systems are Tor[3] and AN.ON/JonDonym¹[4].

In general, publications concerning anonymous communications deal with attacks on the network layer, performance improvements or the consolidation of knowledge with regards to anonymous communication in general. Unfortunately, the underlying cryptographic protocols have not received the same attention, despite the fact that anonymity strongly depends on the correct practical combination, usage and implementation of cryptographic primitives.

* “Center for Quantifiable Quality of Service in Communication Systems, Center of Excellence” appointed by The Research Council of Norway, funded by the Research Council, NTNU and UNINETT. <http://www.q2s.ntnu.no>

¹ Also known as “JAP”, the name of the client software, or “Java Anon Proxy”.

This paper takes a closer look at the cryptographic protocols used for the anonymization process of AN.ON, and discovers several vulnerabilities. The following two sections describe the basic concepts of AN.ON, the attacker model and the underlying assumptions of the system. Section 4 continues with the authentication protocol involving the user and the first server. Section 5 presents an attack on the general encryption scheme used by AN.ON. Section 6 discusses a flaw in the mix initialization protocol. The previous work in this area is presented in Section 7, following a discussion in Section 8 about the reasons for these flaws. Finally, we present our conclusions.

2 Description of AN.ON

AN.ON, short for *Anonymity Online*, is a project that provides anonymity on the network layer. More precisely, it offers sender anonymity against the receiver of a message and relationship anonymity against a local attacker, such as an eavesdropper of an Internet connection. In terms of web browsing, sender anonymity means that a web server cannot identify a user via their IP address. Relationship anonymity means that a local attacker cannot identify the sender or receiver of a message. Thus, an attacker can at most identify the sender or the receiver, but not both [5].

In order to establish such a service, AN.ON uses the so-called *mix servers* otherwise known simply as *mixes*[6]. A mix is an intermediate entity between a sender and a receiver of a message. Its task is to establish anonymity for the user. A mix accomplishes this by hiding the relation between incoming and outgoing messages. In [4] the authors propose the usage of encryption and reordering of messages to establish anonymity. However, for performance reasons the reordering of messages is typically deactivated in AN.ON.

To provide relationship anonymity regarding the operator of a mix, several mixes are typically chained together. Such a chain is called a *cascade* (see Figure 1). The order of the mixes is chosen by their operators and cannot be changed by a user. Every packet which is received by the first mix in the cascade is forwarded

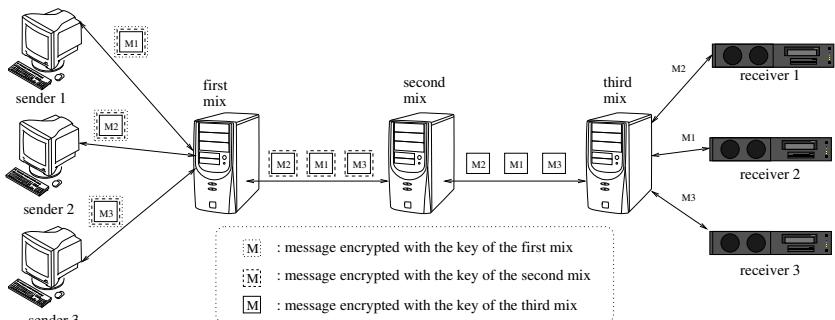


Fig. 1. Example for a cascade in AN.ON

to a second mix and so forth. By these mechanisms, AN.ON aims to establish anonymity for a user under the assumption that not all mix operators collude.

The basic anonymization process works in the following way: a sender encrypts a message, which includes the final destination, with a symmetric key. The key is shared between the last mix in a given cascade and the user. The result of the encryption is again encrypted with a symmetric key that is shared with the predecessor of the last mix. This procedure is repeated until the first mix in the cascade is reached and thereby the typical layered encryption is created.

Afterwards the message is sent to the first mix. This mix uses its shared symmetric key to decrypt the message and forwards the result to the next mix, which also decrypts the message. When the last mix in cascade is eventually reached, the mix performs the final decryption of the message and is thus able to get the destination of the message. Finally, the unencrypted message is forwarded to the destination.

In addition to senders, receivers and mixes, there is an additional party, the so-called *infoservice*. This service, operated by a third party, provides users of the system with the necessary information about the cascades.

3 Scope, Assumptions and Course

The analysis in this paper looks in detail at the protocols used in the mix server version 00.08.60 of AN.ON. This was the most recent version at the time of our analysis and was released in February 2009.

Our analysis focuses on the anonymization process and ignores the protocols that are involved in the information distribution process, which normally involves the infoservice. Due to this, we assume that a user knows the public signature keys of the mixes. Generally, we assume that every mix uses an uncompromised key pair to sign data.

Our assumed adversary has the capabilities of a *local active attacker*. Thus, the attacker is only able to eavesdrop some connections of either the user, the mixes or the final destination, but not all at the same time. The attacker is also able to add, modify, replay or drop packets passing an observed link. We further assume that the attacker can operate a single mix in the cascade. Finally, we assume that the attacker is not able to break basic cryptographic primitives, such as AES or RSA, but is assumed to possess all prior private keys that are no longer used by the mixes. Note that this attacker model is weaker than the attacker that was originally proposed in [4], in which the authors assume a global attacker. However, recent publications have shown that AN.ON is not able to resist this kind of attacker in practise[7,8].

In order to retrieve the information about the protocols we made use of a technical report describing the anonymization process[9]. We have also examined the source code in order to find undocumented changes in the protocols. The results of the analysis were discussed with developers of the AN.ON project, who also helped us to retrieve the mix authentication protocol, see Section 6.

4 Authentication Protocol of the First Mix

Before we describe the protocol it is necessary to take a brief look at the so-called *descriptors* that are provided by the infoservice. A descriptor is an *XML*-based document which describes the entities of a cascade. It contains general information about the cascade, a description for each mix and a signature for the whole document. The last of these is provided by the first mix in the cascade. The description of each mix is also signed by the corresponding mix. The signature aims to prevent a malicious modification of the mixes' descriptions. The description of a mix includes different public keys, a timestamp and X509 certificates for the included public keys.

In this section of the paper we assume that when a user receives a descriptor with a valid signature of a known authority, they possess every public key of the mixes in the cascade. In addition we assume that all keys are not compromised by an adversary and that the certificates are up-to-date.

The *mix authentication protocol* aims to create a session key between the JAP, which is the client application, and the first mix in a cascade. The preconditions for the protocol are that a user knows the public signature key of the first mix. The mix owns the corresponding key pair, which is only used to produce signatures. In addition to this key pair the mix also possesses a public key pair that can only be used for encryption. However, the key which is used for the encryption is not initially known by the JAP.

Figure 2 shows a message sequence chart of the mix authentication protocol at an abstract level. We assume that the JAP knows the public signing key K_{M^1} of the first mix. The *known* relation is represented by the \exists sign. The mix holds two key pairs: the first key pair is $(K_{M^1}^{-1}, K_{M^1})$ which is used only to sign documents. The second key pair is $(K_{M_e^1}^{-1}, K_{M_e^1})$. Its public key can be used by the JAP to encrypt messages for the mix.

In the first message that is sent by the first mix to the user, a descriptor for the cascade is transmitted. The descriptor includes, among other data, the public encryption key $K_{M_e^1}$ of the first mix. In order to simplify the presentation we represent the remaining information by m . For example, the keys of the other mixes are included in m . The whole descriptor is signed by the first mix, which is denoted by $\{H(m, K_{M_e^1})\}K_{M^1}^{-1}$. After the message is received by the JAP, it verifies the signature. If this succeeds it also knows the public encryption key of the first mix. In the next step, the JAP generates a symmetric session key and sends the key encrypted with the public encryption key $K_{M_e^1}$ to the first mix. Since the first mix owns the corresponding private key, it is able to decrypt the message. Thus, it also knows the symmetric session key. Finally, the mix sends a confirmation to show the possession of the symmetric key to the JAP. The confirmation is basically a hash of the descriptor, the session key and the public encryption key of the first mix.

A closer look at the protocol shows that a mix has no guarantee that the message containing the session key was created in the current session. An attacker can send, instead of a new message, a recorded message of an old session. This is known as a *replay*. The mix cannot check, in this protocol, whether the session

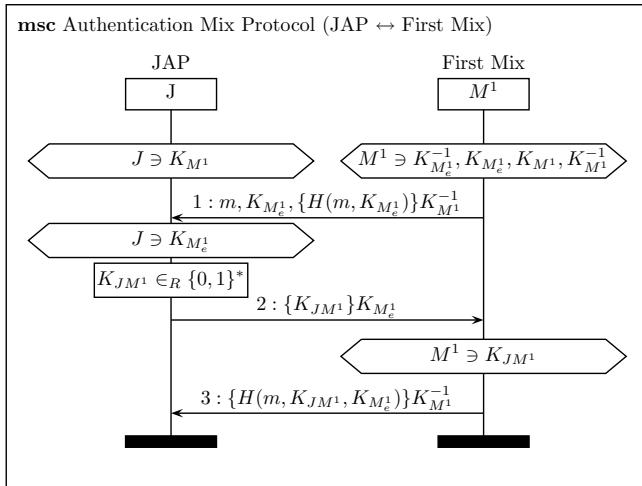


Fig. 2. Mix Authentication Protocol

is a replay or not. Thus, the whole anonymization process may be vulnerable to replay attacks.

As we did not find any replay protection in the other protocols, we tried to replay the whole session including several HTTP requests. We used the existing service “Dresden” in order to check if a replay is possible. To this end, we recorded a session of a short sequence of website queries and their replies. For testing purposes, we retrieved a website which was hosted on a server under our control. The second and third website were hosted on foreign servers. The last website again was located on a server which was under our own control.

To replay the session we simply connected to the mix and sent all the previously recorded raw packets in the mix. Shortly after the replay was started we were able to observe HTTP requests on both of our web servers. Both HTTP requests were sent by the last mix of the “Dresden” cascade. Thus, in the version we evaluated there was no protection against replay². This lack of protection leads to various different attacks threatening not only the anonymity of a user with respect to a global attacker[6]. For instance, an attacker could replay a post command which modifies data on the web server in order to threaten data integrity.

The replay attack can be limited to an internal attacker if the mix ensures that the received key is fresh. The simplest solution seems to be to establish a TLS connection between the user and the first mix. However, whether this solution suits AN.ON’s requirements is not within the scope of this paper.

This change only protects against an external attacker. In order to protect against replay by an internal attacker other countermeasures are necessary, such as the solution described in [10]. Here the authors try to find a trade-off between

² Replay protection is under development, but takes place at a higher level.

storing every used key in a database, performance and security. Their idea is to use small parts of a symmetric key to mark a time period. Some parts of the key are predictable, which subsequently lowers the strength of the symmetric key. An identifier of every used key is stored in a database during a given time period. Only keys that are within the current time period, and that are not already stored in the database, are accepted for a new connection.

5 Attack on AN.ON's Encryption/Decryption Scheme

In this section we concentrate on the encryption and decryption of exchanged messages.

5.1 Structure of Mix Packets

As briefly described in Section 2, a user forwards packets along a chain of mixes. For each hop, the user adds a layer of encryption around the message which is later removed by the corresponding mix.

To analyse the protocol it is necessary to examine the structure of the packets. Figure 3 illustrates the structure of a packet which is sent by a user to the first mix. The first part is the mix packet header. Its size is 6 bytes and it contains a channel ID and some flags. The remaining part can be used to transfer data from the user to the last mix in the cascade. It is important to notice that only the last 992 bytes travel along the whole cascade. The first 6 bytes (channel ID and flags) can be completely modified at each hop. It is therefore necessary to encrypt different parts of the packet in different ways with different keys.

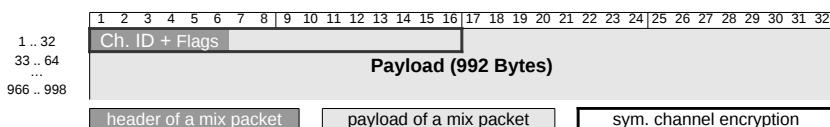


Fig. 3. The Structure of a Mix Packet

One encryption layer is added to protect the confidentiality of the channel ID and the flags during the transmission between adjacent entities. Obviously, the symmetric key for the encryption is shared by exactly two adjacent parties. For purposes of speed optimization the whole packet is not encrypted, only the first 16 bytes (128 bit). Thus, this layer of encryption includes the channel ID, the flags and the remaining 10 bytes of the payload of the message. For encryption, AES is used in the *output feedback mode* (OFB, see Section 5.2)).

The structure and the encryption of the payload depend on whether a mix packet is the first in an anonymous channel, or a packet of an already opened channel. If a mix receives a packet it checks, based on the channel ID and the

flags, if the packet opens a new stream. Such a packet is called a *channel-open packet*. In this case, the mix decrypts the first 128 bytes by using its RSA private key³. The first part of the decrypted bytes contains a symmetric key (16 bytes). This is used for decrypting both the remaining 864 bytes of the packet and the subsequent packets of the anonymous channel. If the mix is an intermediate mix, the remaining part of the packet is encrypted for the next mix (see Figure 4). Thus, the mix needs to forward the packet to the next mix. Before the packet is forwarded, the mix removes its key (16 bytes) from the packet and adds 16 bytes of random data to the end in order to preserve the length of the packet. Finally, the mix forwards the payload together with a matching header to the next mix. If the mix is the last mix in a cascade it decrypts the first 128 bytes with its private RSA key and the remaining bytes with the symmetric key which is stored in the first 16 bytes of the packet. In addition to the payload a mix packet for a last mix contains a header field after the key. This field indicates how many of the 992 bytes for the payload are used for data, as well as the type of the data. The remaining bytes are random data (see Figure 5).

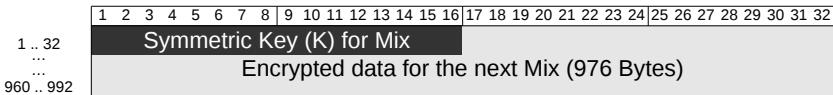


Fig. 4. The structure of the payload for intermediate mixes for the initial packet

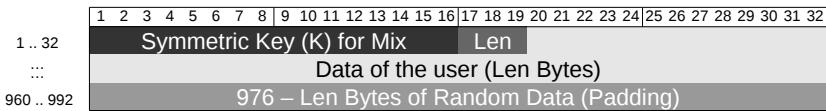


Fig. 5. The structure of the payload for the last mix for the initial packet

In the case of subsequent packets only one encryption scheme is used. The whole payload is encrypted symmetrically with the key that was sent within the channel-open packet. Each mix uses its key to decrypt the 992 bytes of payload. When the packet is decrypted by the last mix, the packet can again be divided in three parts: the header, the data and random data.

In order to process replies, every mix encrypts the data received by the successor mix and forwards it to the predecessor mix. The structure of the mix packet is equal to the structure a subsequent packet. Thus, the whole packet is symmetrically encrypted. The same symmetric keys are used by the mixes for both transmission directions. The algorithm used is AES in OFB-mode.

³ In the case of the first mix, this part is symmetrically encrypted. Since this is not important for the attack we omit a further explanation.

5.2 Output Feedback Mode (OFB)

AN.ON uses AES in OFB mode for its symmetrical encryption operations. The objective of the OFB mode is to produce an infinite key stream. To this end, it uses an initialization vector and a key. The initialization vector is encrypted with AES, which uses the key. The result of this is used twice: first, it is XORed with the plain text. Second, it is used in the subsequent round instead of the initialization vector. Figure 6 illustrates the mode of operation of OFB.

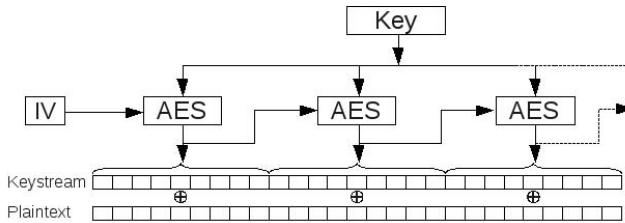


Fig. 6. Sketch of the output feedback mode

5.3 The Attack

As mentioned above, a mix uses the same key to decrypt messages in the sending direction as well as to encrypt the messages in the receiving direction. Moreover, a mix uses the same initialization vector for both directions. Therefore, a mix produces the same key stream for both directions.

We denote with k_i^m the byte of the i -th position in the key stream of mix m . Let d_i denote the i -th data byte in the data stream⁴ and with c_i the final encrypted message byte at position i . To distinguish the sending and receiving direction we use either the superscript r or s for c_i and d_i respectively.

At first we formalize the receiving situation where a packet travels from the last mix along the cascade to the user. Let p denote the position of the last mix that processed the packet and let n be the number of mixes in the cascade. Thus, if the user receives a packet, p is equal to 1 since the first mix was the last involved mix. Let i denote the byte position in the byte stream of a packet. Based on the notation we can describe the i th encrypted byte in the receiving direction by:

$$c_i^r(p) = d_i^r \bigoplus_{j=p}^n k_i^{m_j} \quad i \geq 0 \wedge 1 \leq p \leq n \quad (1)$$

However, if we take a look at the sending direction the situation is slightly different due to the payload structure of the initial packet in a data stream. Recall that if an intermediate mix in a cascade receives a channel-open packet in a data stream, the first 128 bytes are asymmetrically encrypted and the following bytes symmetrically. During processing, an intermediate mix removes its key

⁴ The data stream includes the 3 byte header of each packet as well as the padding.

from the payload (and adds 16 bytes to the end). Now the packet is forwarded to the next mix who expects again that the first 128 bytes of the payload are encrypted asymmetrically and the remaining bytes are encrypted symmetrically. Thus, the key streams of the mixes need to be shifted by a user by 16 byte per hop. The reason for the shift is the symmetric key that is stored in the first 16 bytes of the initial data stream packet.

By formalizing the encryption, we result in:

$$c_{i+128}^s(p+1) = d_{i+128}^s \bigoplus_{j=p+1}^n k_{i+(n-j)*16}^{m_j} \quad 0 \leq p \leq n-1 \wedge i \geq (n-p-1) \cdot 16 \quad (2)$$

for the i -th encrypted byte.

The equations 1 and 2 become interesting if we consider the first mix in a cascade of length 2. By using both formulas we result in:

$$c_i^r(2) = d_i^r \oplus k_i^{m_2} \quad (3)$$

$$c_{i+128}^s(2) = d_{i+128}^s \oplus k_i^{m_2} \quad (4)$$

The xor of both encrypted values result in:

$$c_i^r(2) \oplus c_{i+128}^s(2) = d_{i+128}^s \oplus d_i^r \quad (5)$$

In case of AN.ON most of the traffic that is transferred over the cascade is normal HTTP traffic, which includes the HTTP header. Moreover, the content of an HTTP header is partially known by the adversary. Thus, an attacker can use the known parts in order to decrypt unknown parts of the HTTP header with help of Formula 5. This becomes critical in AN.ON if the attacker is able to reconstruct the *request line* or the *Host* field in an HTTP request. The former includes the requested URL and the latter the queried host. Therefore, if an attacker is able to reconstruct parts of either the *Host* field or the request line he has deanonymized the user.

For a proof of concept, we have recorded the payload parts of the packets that were either sent from the first mix to the second mix ($p = 1$) or sent from the second mix to the first mix ($p = 2$) in the *Dresden-Dresden* cascade. We therefore assume an internal attacker on the first mix. In order to correct the offset of the sending stream we removed the first 128 bytes of the stream. The result of this was XORed with the receiving stream. To omit the 3 bytes of the payload header we removed the first 3 bytes of the result and XORed it with the most probable HTTP response line “*HTTP/1.0 200 OK\r\n*⁵”. This procedure resulted in the string “st: www.google.de” which is the last part of the *Host* field in the original HTTP request header. Thus, we uncovered the destination of the request simply through the use of two recorded encrypted packets that were sent and received by the first mix. Since the first mix in the cascade also knows the IP address of the user, an internal attacker is able to revoke the relationship

⁵ There are only a few different possibilities for the response line.

anonymity without the help of the second mix. Clearly, this contradicts to the objectives of AN.ON.

For our attack we have assumed a local internal attacker. Nevertheless, an external attacker is also able to perform the attack, even though it is slightly more difficult. This difficult is for three reasons: firstly, the attacker does not know the mapping between incoming and outgoing messages. Thus, they cannot map directly the IP address of the sender to the uncovered receiver. Secondly, the attacker cannot use the first 7 bytes of the HTTP response in the payload due to the channel encryption. Thus, they have less information available. Thirdly, the attacker cannot easily recognize which received packet belongs to which sent packet. These constraints are not, in our opinion, a significant challenge. The mapping can be received due to the fact that the packets are processed in a FIFO order. The fact that the attacker misses 7 bytes merely lowers the probability of success slightly. The last challenge can be addressed by probing which received packet leads to a useful output with respect to a recorded sent packet. If we assume the external attacker is able to master the first and third challenge, they are able to deanonymize the user in our example. The attacker is able to retrieve “.google.de” without any further guesses.

5.4 Discussion of the Attack

The attack presented above is based on several problems. Firstly, the plain text of the encrypted message is partially known. Secondly, the encryption is a XOR encryption and therefore an encrypted bit only depends on a single bit of the plain text as well as the key stream. Thirdly, the same parameters are used for both directions. The first and the second fact are difficult to avoid due to the design of AN.ON. Thus neither the *cipher feedback (CFB)* nor the *cipher block chaining (CBC)* mode can be used , due to the processing of the initial packet in a data stream. The *electronic codebook (EBC)* mode is also not suitable as it does not hide data patterns. Thus, it is only possible to change the parameters of the encryption, preferably the key. The key streams of both stream directions thus become different. In a conversation with the developer[11] it was mentioned that AN.ON recently became aware of these risks and that changes had been made in order to use different keys. This was independent of our analysis. Hence, the mix software has already been updated to reflect this issue.

6 Attack on the Mix Authentication Protocol

In this section we look at the cascade initialization protocol between mixes. The protocol aims to exchange a key with adjacent mixes in the cascade. In addition, it should also mutually authenticate the mixes.

Let m_1, \dots, m_n the mixes in a cascade. The protocol begins with the establishment of a TCP connection between the mixes m_i and m_{i+1} . Note that m_i initiates the connection to m_{i+1} .

Figure 7 depicts the protocol between two mixes. It starts with the generation of a nonce (n) and an asymmetric encryption key ($K_{M_e^{i+1}}^{-1}$) by the mix $i + 1$.

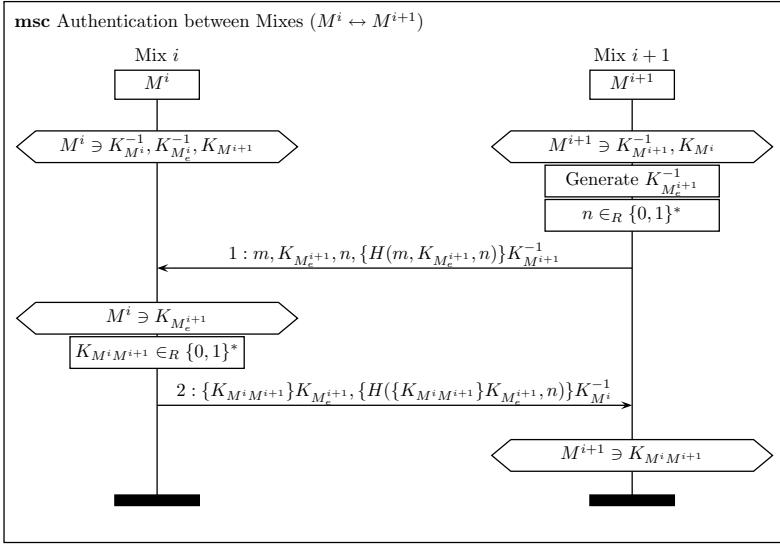


Fig. 7. Mix Authentication Protocol

Afterwards, mix $i + 1$ transmits its description (m), its public encryption key ($K_{M_e^{i+1}}$), the generated nonce and a signed hash of the triple ($m, K_{M_e^{i+1}}, n$) to mix i . Mix i checks whether the received signature is valid with respect to the known key of mix $i + 1$ or not. In the former case mix i generates a session key and encrypts the session key with the received key $K_{M_e^{i+1}}$. The result is sent together with a signature of the encrypted key and the nonce to the mix $i + 1$. Mix $i + 1$ checks the validity of the signature with respect to the configured public key of mix i . If it succeeds mix $i + 1$ uses the received key as symmetric key for the channel encryption between the two mixes.

A problem arises when an attacker compromises the private encryption key of the last mix in the cascade, possibly at a later point in time. In this case the attacker is able to replace the certified mix with his own mix. To this end, the attacker needs a recorded session of the authentication protocol in which the compromised key was used. In order to mount the attack the attacker redirects the TCP connection from the certified mix to its own mix. Afterwards, the mix replays the first message of the previous session to the mix i . Due to the fact that the signing key is usually changed only once a year, the mix i will most probably accept the signature of the “certified” mix. In correspondence with the protocol, mix i generates a session key and sends the encrypted session key together with its signature back to the attacker. The attacker who knows the private encryption key can now decrypt the session key and is therefore authenticated in the cascade as certified mix even though he does not possess the signature key. A user is unable to distinguish the attacker from the certified mix at a later date via the existing protocols. Therefore, the attacker is able to eavesdrop on all the outgoing data of the users, which may contain identifying

information. However the attacker is not able to deanonymize users solely based on this attack.

At a first glance, the attack looks impractical due to the fact that an attacker needs to compromise one of the private encryption keys. However, if we consider, for example, the recent OpenSSL bug in the Debian Linux distribution⁶[12], this attack becomes more practical. The mix software relies on OpenSSL, and thus any asymmetric encryption key generated by a mix which used a vulnerable OpenSSL library is potentially compromised. This means that an attacker can immediately retrieve the private key from a given public key generated by a vulnerable OpenSSL version. Hence, if an attacker once recorded a session in which a mix used a vulnerable key, he is able to impersonate the mix with the attack described above. The only way to circumvent the attack in the current version is to replace every signature key that has been used with a vulnerable OpenSSL version. Obviously, the protocol also needs to be fixed to counter the described attack.

It is worth noting that this protocol is based on the “Key Transport Mechanism 4” of the ISO/IEC 11770-3:2008 standard[13]. The author of the AN.ON protocol modified it slightly in order to authenticate mix_{i+1} as well. The author therefore included a signed version of the mix_{i+1} ’s encryption key as well as the descriptor of the mix. In addition he omitted the identity of mix from the encrypted secret, which could lead to other attacks. This example shows how dangerous it is to modify standardized cryptographic protocols to apply them beyond their intended use.

For the protocol in Section 4 we see no reason why a custom-made or a modified standard protocol is necessary for the authentication and encryption. TLS supports client and server authentication via X509 certificates and is additionally able to secure data transmitted later. This protocol should therefore be suitable for the communication and authentication between the mixes. One reason to choose another protocol might be performance, as some data is encrypted unnecessarily in this scheme.

7 Related Work

This paper is the first cryptographic protocol analysis of AN.ON’s anonymization process. In 2009, Westermann[14] performed a security analysis of AN.ON’s payment system, but did not take the anonymization process into account.

For I2P⁷, an anonymously developed anonymization service, there is no published description of the anonymisation protocol available and to the best of our knowledge also no publicly available security analysis.

In contrast to AN.ON and I2P, the cryptographic protocols of the Tor system have been analyzed[3] with the NRL protocol analyzer[15]. In 2006 Goldberg proved that the Tor authentication protocol is secure in the random oracle model[16]. In general, this does not guarantee that the implementation has no

⁶ A vulnerable version can only generate a limited number of keys.

⁷ <http://www.i2p2.de>

flaws with respect to the implementation of the protocol and the cryptographic primitives. An example of this is that, due to the lack of AES in counter mode in early OpenSSL versions, the Tor developers were forced to implement their own version. Unfortunately, there was a bug in this implementation that caused the counter to be reset after 16 bits. This clearly threatened the security of the system[17].

8 Discussion

In the field of low-latency anonymous communications, the main research focus seems to be on mechanisms that establish anonymity or improve performance. Many publications deal only with the general mechanisms for establishing anonymity by using idealized underlying protocols, and omit a clear and detailed cryptographic protocol description. However, most mechanisms are not implemented and thus this lack of detail is a minor problem. As soon as a protocol is implemented, however, it is crucial to publish and analyse the protocols that are composed or invented by the authors.

Tor is a good example of the right way to achieve this. The developers described and analysed the cryptographic protocols in a early stage of the project. Possible changes to the cryptographic protocols are published before they are implemented in Tor. In [18] the authors propose a more efficient way to establish circuits, however to the best of our knowledge this is not implemented yet, but is in discussion to be introduced in a later version.

In general, it is almost always a good idea to use standard cryptographic protocols for a product. However, building an anonymity network solely on standardized protocols, while possible, introduces a number of constraints[19]. In the case of high-latency anonymity networks, with regard to the protocols and mechanisms proposed so far, it seems almost unavoidable to compose cryptographic primitives and invent cryptographic protocols for novel, specific purposes. However, it seems that this area enjoys a stronger focus on proving the correctness of protocols compared to the field of low-latency networks.

Our analysis shows that referring to a technical report for cryptographic protocols is risky. We claim that a technical report is mostly read by developers, who are not necessarily cryptographic protocol experts. As a consequence, weaknesses in the protocols are more likely to be overlooked.

9 Conclusion

In this paper we have analysed the cryptographic protocols of AN.ON and discovered three flaws of differing severity. The first flaw is caused by the fact that the freshness of the session key was not checked by the mix. This flaw leads to a situation where an external attacker is able to perform a replay attack against AN.ON. However, when the replay detection techniques that are currently under development are integrated, the internal as well as the external attacker will

no longer able to replay a session. Nevertheless, the flaw in the authentication protocol must be addressed.

A second, more severe, error was found in the encryption scheme of AN.ON. An internal attacker controlling the first mix in a cascade of length two is able to de-anonymize users with high probability. The error was introduced by the reuse of keys with the same initialization vector. As of November 15, 2009, this error is fixed in version 00.08.84 by a slight protocol change. Due to compatibility reasons with older clients, some mix operators have still not updated, but plan to do so soon.

The third flaw discovered is, at a first glance, more theoretical than practical. It does, however, have practical relevance due to the OpenSSL flaw in Debian. The missing check for a message to belong to the current session enables an attacker to impersonate the last mix in a cascade. However, this can only be done if the attacker has compromised a private encryption key of the mix that was signed by the last mix in an older session.

The flaws we discovered represent errors that, unfortunately, still occur quite often. This again shows the importance of using standardized cryptographic protocols. As discussed in Section 8 it is not always possible to use a standard cryptographic protocol due to special requirements. In this case, a composition of cryptographic protocols and primitives becomes necessary. This does not necessarily lead to a secure system, as various examples and attacks in the past have shown[20,21]. Therefore, a proof or detailed analysis should be provided, as it has been given by Tor or by Sphinx[22].

References

1. Freedman, M.J., Morris, R.: Tarzan: a peer-to-peer anonymizing network layer. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 193–206. ACM, New York (2002)
2. Rennhard, M., Plattner, B.: Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002), Washington, DC, USA (November 2002)
3. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: USENIX Security Symposium, pp. 303–320. USENIX (2004)
4. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 115–129. Springer, Heidelberg (2001)
5. Pfitzmann, A., Hansen, M.: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management - a consolidated proposal for terminology, vol. 0.31 (February 2008)
6. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 4(2), 84–88 (1981)
7. Kesdogan, D., Agrawal, D., Pham, V., Rautenbach, D.: Fundamental limits on the anonymity provided by the mix technique. In: SP 2006: Proceedings of the 2006 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 86–99. IEEE Computer Society, Los Alamitos (2006)

8. Berthold, S., Böhme, R., Köpsell, S.: Data retention and anonymity services - introducing a new class of realistic adversary models. In: The Future of Identity in the Information Society. IFIP Advances in Information and Communication Technology, vol. 298, pp. 92–106 (2009)
9. Köpsell, S.: AnonDienst - Design und Implementierung. Technical report, TU Dresden University (2004)
10. Köpsell, S.: Vergleich der Verfahren zur Verhinderung von Replay-angriffen der Anonymisierungsdienste AN.ON und Tor. In: Dittmann, J. (ed.) Sicherheit. LNI, vol. 77, pp. 183–187, GI (2006)
11. Köpsell, S.: Private discussion with the developer (May 2009)
12. Common Vulnerability and Exposure: CVE-2008-0166 (2008), <http://www.cve.mitre.org> (last visited: 15.12.2009)
13. ISO/IEC 11770-3:2008: Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques. ISO, Geneva, Switzerland
14. Westermann, B.: Security analysis of AN.ON's payment scheme. In: Jøsang, A., Maseng, T., Knapskog, S.J. (eds.) NordSec 2009. LNCS, vol. 5838, pp. 255–270. Springer, Heidelberg (2009)
15. Meadows, C.: The NRL protocol analyzer: An overview. *The Journal of Logic Programming* 26(2), 113–131 (1996)
16. Goldberg, I.: On the security of the Tor authentication protocol. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 316–331. Springer, Heidelberg (2006)
17. Dingledine, R.: Security and Anonymity Vulnerabilities in Tor: Past, Present, and Future. Talk at DefCon 16 (August 2008)
18. Øverlier, L., Syverson, P.: Improving efficiency and simplicity of Tor circuit establishment and hidden services. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 134–152. Springer, Heidelberg (2007)
19. Panchenko, A., Westermann, B., Pimenidis, L., Andersson, C.: Shalon: Lightweight anonymization based on open standards. In: Proceedings of 18th International Conference on Computer Communications and Networks, San Francisco, CA, USA (August 2009)
20. Simmons, G.J.: Cryptanalysis and protocol failures. *Communications of the ACM* 37(11), 56–65 (1994)
21. Gligoroski, D., Andova, S., Knapskog, S.J.: On the importance of the key separation principle for different modes of operation. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 404–418. Springer, Heidelberg (2008)
22. Danezis, G., Goldberg, I.: Sphinx: A compact and provably secure mix format. In: IEEE Symposium on Security and Privacy, pp. 269–282. IEEE Computer Society, Los Alamitos (2009)

A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys

Sven Schäge and Jörg Schwenk

Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Germany
`{sven.schaege,joerg.schwenk}@rub.de`

Abstract. In this work we present a new CDH-based ring signature scheme with some striking advantages. On the one hand it is secure without random oracles, perfectly anonymous, and unforgeable solely under the CDH assumption in bilinear groups. This makes the security of our ring signature schemes rely on weaker (and less) assumptions than all previous (full) ring signature schemes secure without random oracles. On the other hand the scheme is very space efficient; a public key consists of just a single group element and a ring signature accounts for only $n + 1$ group elements, where n is the size of the ring. This is only about half the number of components when compared to other ring signature schemes that do not exploit ring re-use. As all computations are in groups of prime order, we do not need a trusted setup procedure. All these features do not come for free. The main drawback of our scheme is that it only provides security against chosen subring attacks where the attacker is not allowed to query private keys.

Keywords: CDH assumption, bilinear group, ring signature scheme, programmable hash function.

1 Introduction

The CDH assumption became practical for standard model signature schemes with the introduction of bilinear pairings into cryptography. In 2005, Waters showed the existence of a hash-and-sign signature scheme that is secure under the CDH assumption in the standard model [31]. Since then several signature schemes, including ring signature schemes [27], sequential aggregate signature schemes, multisignature schemes, and verifiably encrypted signature schemes [23] have been proposed that are secure in the standard model. In this work we develop a new and efficient ring signature schemes without random oracles that is solely based on the CDH assumption in symmetric bilinear groups.

A ring signature scheme allows a signer to sign on behalf of a group of users, the so-called ring; the only condition is that the signer must also be part of this ring. Technically, a ring is represented by the set of public keys that correspond to the identities of the ring members. Using his private key, the signer can now sign a message such that anyone can check whether the signature has been generated by one of the ring members. At the same time, there exists no possibility to

discover the actual signer. Ring signatures provide signer anonymity in a very strong sense. In contrast to group signature schemes [14], the anonymity of the signer cannot be revoked. What makes ring signature schemes very flexible is that no central management is needed and that the signer can freely choose the public keys in the ring even without their owners' consent. Direct applications for ring signature schemes include designated verifier signatures [22] and secret leaking [26], but ring signature schemes are in general useful in applications where signer anonymity is desired.

1.1 Related Work

The first (explicit) ring signature scheme by Rivest, Shamir and Tauman [26] was proven secure in the random oracle/ideal cipher model [2, 16]. Since then, several ring signature schemes have been proposed in the random oracle model. In 1998, Canetti, Goldreich and Halevi showed the existence of a signature scheme that is provably secure in the random oracle model but insecure when instantiated with any hash function [12], thus raising serious doubts on the usefulness of the random oracle model for real world protocols. Since then, research on cryptographic primitives that are secure in the standard model has gained much attention. However, today only a handful of ring signature schemes proven secure without random oracles exist.

While the scheme of Chow et al. [15] published in 2006 provides unconditional anonymity, unforgeability is based on a new strong assumption that is given without any evidence for its validity. In the same year, Bender, Katz and Morselli proposed a ring signature scheme based on trapdoor permutations, but since it uses generic ZAPs for NP it is impractical for real world applications [4]. In the same work the authors also presented two 2-user ring signature schemes without random oracles that are secure under the CDH and the LRSW assumption. Disadvantageously, these schemes only allow to issue signatures on rings of maximal size 2. This is security critical since in a ring signature scheme the provided level of signer anonymity is primarily determined by the number of ring members. Thus, dependent on the application and his requirements on an appropriate security level *the user* should decide on the size of the ring. In 2007, Shacham and Waters presented a ring signature scheme [27] that is full key-exposure anonymous, a strong security notion stemming from [4], under the Subgroup Decision assumption [5]. Unfortunately, this assumption relies on groups with composite order such that a trusted setup procedure is necessary in the setup phase. Also, the representation of group elements is rather large (about 1024 bits). Unforgeability is based on the CDH assumption and the signature size is $2n + 2$ group elements, where n is the size of the ring. In the same year, Boyen presented a new signature scheme with perfect anonymity [7]. Unforgeability of this scheme is based on a new complexity assumption, the Pluri-SDH assumption, while evidence for its usefulness is provided by a security analysis in the generic group model. The signature size consist of n group elements and n integers (of 160 bits) while each public key consists of at least three group elements. Recently, Chandran, Groth and Sahai proposed a new signature scheme with perfect anonymity

that is secure under the Subgroup Decision assumption and the Strong Diffie-Hellman assumption [13]. Since the above remarks concerning the trusted setup of [27] also hold here, the authors present two variants of their ring signature scheme. The second variant accounts for maliciously generated common reference strings by heuristically guaranteeing (by using a factorization algorithm) that the composite group order output by the setup algorithm is hard to factor.

Except for the schemes by Chandran et al. [13] and Dodis et al. [17] (in the random oracle model), all existing ring signature schemes offer signature sizes that are at least linear in the ring size. Both, [13] and [17] provide better (asymptotic) efficiency when several messages are signed using the same ring.

1.2 Contribution

In this work we present a new ring signature scheme for rings of arbitrary size. Anonymity is perfect, unforgeability solely relies on the CDH assumption in bilinear groups. Security is proven in the fully untrusted common reference string model. The signature size is very small, accounting for only $n+1$ group elements. Since we use programmable hash functions [20], a drawback of our scheme is that we require relatively large global parameters, consisting of around 160 group elements. However, these parameters can be re-used for all instantiations of the scheme that use the same bilinear group. Advantageously, in our ring signature scheme, each public key consists of a *single* group element such that for large groups (e.g. >1000), the public parameters only account for a small portion of the data required for signature generation and verification. Finally we provide a new proof technique for Waters-like signature schemes which is very clean and compact at the same time. The main drawback of our scheme is that it only provides security under chosen subring attacks, where the attacker is not allowed to query secret keys of honest ring members. We stress that our ring signature scheme is much more practical than the CDH-based scheme by Bender, Katz, and Morselli that is also secure under the CDH assumption. First, our scheme can be used to sign messages for rings of arbitrary length, not only for 2-user rings. Second, in our scheme a public key contains only a single group element whereas in the Bender et al. scheme a public key consists of a complete group hash function.

2 Preliminaries

Before presenting our constructions we briefly review the necessary preliminaries.

2.1 Ring Signature Scheme

A ring signature scheme \mathcal{RSG} consists of three algorithms. Given the security parameter 1^κ , the probabilistic polynomial time (PPT) algorithm **KeyGen** generates a secret and public key (SK, PK) . The PPT algorithm **Sign** takes as input a tuple of public keys $R = (PK_1, \dots, PK_n)$, a secret key SK_i with

$i \in \{1, \dots, n\}$ and a message m and outputs a signature σ . Finally, the deterministic polynomial time algorithm **Verify** processes R , a message m and a signature σ to check whether σ is a legitimate signature on m signed by a holder of a secret key corresponding to one of the public keys in R . Accordingly, the algorithm outputs 1 to indicate a successful verification and 0 otherwise. Note that for simplicity, we do not assume an explicit setup algorithm. In the following, all global parameters depend on 1^κ . We stress that we do not rely on a trusted setup authority.

2.2 Ring Unforgeability

In our paper, we concentrate on unforgeability against chosen subring attacks that is formalized in the following attack game between a challenger and an adversary.

Setup. The challenger runs **KeyGen** n times to obtain the key pairs $(SK_1, PK_1), \dots, (SK_n, PK_n)$. Next, $R = (PK_1, PK_2, \dots, PK_n)$ is given to the adversary.

Adaptive signature queries. The adversary adaptively sends q signature queries to the challenger. For $i \in \{1, \dots, q\}$, each query Q_i consists of a message m_i , a set $R_i \subseteq R$ of public keys and an index $e_i \in \{1, \dots, n\}$. When the challenger receives the i 'th query $Q_i = (m_i, R_i, e_i)$, he computes $\sigma_i = \mathbf{Sign}(R_i, SK_{e_i}, m_i)$ and sends it to the adversary.

Output. The attacker outputs (m^*, R^*, σ^*) with $m^* \notin \{m_1, \dots, m_q\}$.¹

We denote the success probability of an adversary \mathcal{A} (taken over the random coins of the challenger and the adversary) to win the above game as $Adv_{\mathcal{R}SIG, \mathcal{A}, \text{unf}}$.

Definition 1 (Ring unforgeability). We say that a ring signature scheme is (t, ϵ, q) -secure, if no t -time attacker has success probability at least ϵ in the above attack game after making q signature queries.

2.3 Ring Anonymity

The strongest notion of anonymity for ring signature schemes is perfect anonymity. Formally, we consider the following attack game between a challenger and an *unbounded* adversary.

Setup. The challenger runs **KeyGen** n times to obtain the key pairs $(SK_1, PK_1), \dots, (SK_n, PK_n)$. The set of the so computed public keys $R = (PK_1, PK_2, \dots, PK_n)$ is given to the adversary.

Adaptive signature and corrupt queries. The adversary adaptively sends q signature queries Q_1, \dots, Q_q to the challenger and receives the corresponding answers $\sigma_1, \dots, \sigma_q$. At the same time, the adversary may adaptively query up to n secret keys SK_i with $i \in \{1, \dots, n\}$.

¹ We note that a ring signature scheme which is secure under this security definition can easily be adapted to meet the slightly stronger security notion in [4] which solely requires $(m^*, R^*) \notin \{(m_1, R_1), \dots, (m_q, R_q)\}$: given message m and subring R we simply sign $\bar{m} = h(h(m)||h(R))$ instead of m where h is a collision-resistant hash function. For any new (m^*, R^*) , \bar{m}^* will now be distinct from all previous values.

Output. Finally, the attacker outputs a message m^* , a set of public keys $R^* \subseteq R$ and two distinct indices $i_0, i_1 \in \{1, \dots, n\}$ such that $PK_{i_0}, PK_{i_1} \in R^*$. The challenger randomly chooses $b \in \{0, 1\}$, computes $\sigma^* = \text{Sign}(m^*, R^*, SK_b)$, and sends σ^* to the attacker. The attacker then outputs b' , indicating his guess for b .

We denote the advantage of an adversary \mathcal{A} (taken over the random coins of the challenger and the adversary) to win the above game as

$$\text{Adv}_{\mathcal{R}\mathcal{S}\mathcal{T}\mathcal{G}, \mathcal{A}, \text{ano}} = |\Pr[\mathcal{A} \text{ outputs } b' = b] - \Pr[\mathcal{A} \text{ outputs } b' \neq b]|.$$

Definition 2 (Perfect ring anonymity). We call a ring signature scheme perfectly anonymous, if even an unbounded adversary has no advantage ($\text{Adv}_{\mathcal{R}\mathcal{S}\mathcal{T}\mathcal{G}, \mathcal{A}, \text{ano}} = 0$) in winning the above game.

2.4 Complexity Assumptions

Definition 3 (Computational Diffie-Hellman problem). Let \mathbb{G} be a group of prime order. The computational Diffie-Hellman problem (CDH) in \mathbb{G} is, given $g, g^a, g^b \in \mathbb{G}$, to compute $g^{ab} \in \mathbb{G}$.

We say that algorithm \mathcal{A} (t, ϵ) -solves the CDH problem in \mathbb{G} when, in time t , \mathcal{A} has success probability at least ϵ in breaking the CDH problem such that

$$\Pr[g^{ab} \leftarrow \mathcal{A}(g, g^a, g^b)] \geq \epsilon,$$

where the probability is over g, a, b and the random coin tosses of \mathcal{A} .

Definition 4. We say that the (t, ϵ) -CDH assumption holds, if no attacker can (t, ϵ) -solve the CDH problem.

2.5 Bilinear Groups

In the following, we briefly recall some of the basic properties of bilinear groups. Definitions 6 and 7 help to support the intuition behind our security proof. In [6], Boneh, Mironov and Shoup use a similar approach to describe their tree-based signature scheme. However, in contrast to [6], we focus on proving security under the classical CDH assumption, where the challenge and the solution consist of elements from a single group \mathbb{G} . We therefore concentrate on symmetric bilinear groups. We stress that, after some minor modifications, we can base our signature schemes on asymmetric bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with an efficient homomorphism $\Phi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$. However, security is then based on the co-CDH assumption.

Definition 5 (Bilinear group). Let \mathbb{G} and \mathbb{G}_T be groups of prime order p . Let g be a generator of \mathbb{G} . The function

$$e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$$

is a bilinear map (pairing) if it holds that $\forall a, b \in \mathbb{G} \ \forall x, y \in \mathbb{Z} : e(a^x, b^y) = e(a, b)^{xy}$ (bilinearity), $e(g, g) \neq 1_{\mathbb{G}_T}$ is a generator of \mathbb{G}_T (non-degeneracy), and e is efficiently computable (efficiency). We call $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ a symmetric bilinear group.

Definition 6 (Secure bilinear map). A bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is (t, ϵ) -secure if for all t -time adversaries \mathcal{A} it holds that

$$\Pr [e(g, g') = e(h, \mathcal{A}(g, g', h)) \mid g, g', h \in_R \mathbb{G}, h \neq 1_{\mathbb{G}}] \leq \epsilon,$$

where the probability is taken over the random coin tosses of \mathcal{A} and the random choices of g , g' , and h .

One can easily see that in symmetric bilinear groups, breaking the security of a bilinear map is equivalent to breaking the CDH assumption.

Lemma 1. Let $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ be a symmetric bilinear group. Then, e is (t, ϵ) -secure if and only if the (t, ϵ) -CDH assumption holds in \mathbb{G} .

The proof is straight-forward. For completeness, a proof of Lemma 1 can be found in Appendix A. Let again $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ be a bilinear group with a secure bilinear map e .

Definition 7 (Collision generator for bilinear groups). A collision generator for e is a polynomial time algorithm that on input two elements $g, h \in \mathbb{G}$ outputs a collision $(g', h') \in \mathbb{G}$ such that

$$e(g, g') = e(h, h').$$

For symmetric pairings there exists an efficient collision generator that can output *all* possible collisions: given g, h randomly choose $r \in \mathbb{Z}_p$ and compute $g' = h^r$ and $h' = g^r$.

2.6 Multi-generator Programmable Hash Function

In our (ring) signature schemes we use the multi-generator programmable hash function by Hofheinz and Kiltz in groups with known prime order [20] which in turn is based on the CDH-based signature scheme by Waters [31].

Definition 8 (Multi-Generator PHF). The multi-generator programmable hash function consists of four algorithms.

1. Given 1^κ , $l = l(\kappa)$ and a group \mathbb{G} of prime order p , **GHGen** returns $l + 1$ random group generators $u_0, u_1, \dots, u_l \in \mathbb{G}$.
2. Given the u_i and a message $m \in \{0, 1\}^l$, **GHEval** outputs

$$u(m) = u_0 \prod_{i=1}^l u_i^{m_i},$$

where (m_l, \dots, m_1) is the binary representation of m : $m = \sum_{i=1}^l m_i 2^{i-1}$. The pair $(\mathbf{GHGen}, \mathbf{GHEval})$ is called a group hash function.

3. On input 1^κ , l and generators $g, h \in \mathbb{G}$, the algorithm **PHTrapGen** randomly chooses $a'_0, a_1, \dots, a_l \in \{-1, 0, 1\}$ and $b_0, b_1, \dots, b_l \in \mathbb{Z}_p$. Next, it sets $a_0 = a'_0 - 1$ and outputs $l + 1$ group elements $u_i = g^{a_i} h^{b_i}$ for $i = 0, 1, \dots, l$ and the trapdoor $(a_0, a_1, \dots, a_l, b_0, b_1, \dots, b_l)$.
4. Now, given $(a_0, a_1, \dots, a_l, b_0, b_1, \dots, b_l)$ and a message m , the algorithm **PHTrapEval** outputs $a(m) = a_0 + \sum_{i=1}^l a_i m_i$ and $b(m) = b_0 + \sum_{i=1}^l b_i m_i$. Note that when the u_i have been computed by **PHTrapGen** it clearly holds that $u(m) = u_0 \prod_{i=1}^l u_i^{m_i} = g^{a(m)} h^{b(m)}$.

Hofheinz and Kiltz showed that for every fixed polynomial $q = q(\kappa)$ the multi-generator programmable hash function is $(1, q, 0, P_{q,l})$ -programmable where $P_{q,l} = \mathcal{O}\left(\frac{1}{q\sqrt{l}}\right)$. This means, that the group elements output by **GHGen** and **PHTrapGen** are equally distributed. Furthermore it holds for all possible input parameters to **PHTrapGen** and all $M_1, \dots, M_{q+1} \in \{0, 1\}^l$ with $M_{q+1} \neq M_i$ for $i \leq q$ that

$$\Pr[a(M_{q+1}) = 0 \wedge a(M_1), \dots, a(M_q) \neq 0] \geq P_{q,l}.$$

The corresponding proof and further details on programmable hash functions can be found in the original paper [20]. A similar but weaker result ($P_{q,l} = \frac{1}{8(l+1)q}$) was implicitly given by Waters in [31].

3 Efficient Ring Signature Scheme \mathcal{RS}

In this section we present our ring signature scheme \mathcal{RS} that allows for very short public keys and signatures. In \mathcal{RS} , the global parameters consist of $l + 2$ random elements $h, u_0, u_1, \dots, u_l \in \mathbb{G}$ that give rise to a group hash function $u(m) = u_0 \prod_{j=1}^l u_j^{m_j}$ and a symmetric bilinear group $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ with a secure bilinear map.

KeyGen(1^κ). Each user i chooses a random element $x_i \in \mathbb{Z}_p$ as his secret key SK_i . The corresponding public key is $PK_i = g^{x_i}$.

Sign($PK_1, \dots, PK_n, SK_t, m$). Given a ring of n public keys, the holder of secret key SK_t with $t \in \{1, \dots, n\}$ can sign a message $m \in \{0, 1\}^l$ in the following way: for all $i \in \{1, \dots, n+1\} \setminus \{t\}$ he chooses $r_i \in_R \mathbb{Z}_p$ and sets

$$s_i = g^{r_i}.$$

Then, he computes

$$s_t = \left(h \cdot \prod_{\substack{i=1 \\ i \neq t}}^n PK_i^{-r_i} \cdot \left(u_0 \prod_{j=1}^l u_j^{m_j} \right)^{-r_{n+1}} \right)^{1/x_t}.$$

The final signature is $\sigma = (s_1, \dots, s_{n+1})$.

Verify($PK_1, \dots, PK_n, m, \sigma$). Given a set of n public keys, a message m , and a ring signature $\sigma = (s_1, \dots, s_{n+1})$, verify the following equation:

$$\prod_{i=1}^n e(s_i, PK_i) \cdot e\left(s_{n+1}, u_0 \prod_{j=1}^l u_j^{m_j}\right) \stackrel{?}{=} e(g, h) .$$

4 Security

In this section, we show that \mathcal{RS} provides ring unforgeability and perfect ring anonymity according to Definition 1 and 2 (correctness can easily be verified by inspection).

4.1 Ring Unforgeability

Theorem 1. Suppose the $(t_{CDH}, \epsilon_{CDH})$ -CDH assumption holds in the group \mathbb{G} . Then the ring signature scheme \mathcal{RS} is (t, ϵ, q) -secure against chosen subring attacks provided that

$$\epsilon \leq \epsilon_{CDH}/P_{q,l}, \quad t \approx t_{CDH}.$$

Proof. By contradiction. Assume there exists an adversary \mathcal{A} that breaks the security of the ring signature scheme in time t with probability ϵ after q signature queries. Then, one can construct an algorithm \mathcal{B} that uses \mathcal{A} as a black box to solve the CDH assumption. We assume that \mathcal{B} is given a random challenge for the CDH-problem: $(\bar{g}, \bar{g}^a, \bar{g}^b) \in \mathbb{G}^3$. The main idea behind our proof is the following. Recall Definition 7 and Lemma 1. Given two group elements $g, h \in \mathbb{G}$, it is easy to generate all pairs $(g', h') \in \mathbb{G}^2$ such that $e(g, g') = e(h, h')$.

On the other hand, given three group elements g, g', h , the problem of finding a corresponding h' is as hard as solving the CDH problem. Our aim is to transfer this situation to the unforgeability game of our ring signature scheme: the simulator has to choose the input parameters for the attacker such that answering signature queries is as easy as computing collisions and computing a forgery is as hard as breaking the CDH assumption.

In the following, we provide a proof of security that proceeds in a sequence of games [28, 3]. Let $\Pr[S_i]$ denote the success probability for an attacker to successfully forge signatures in Game i .

Game₀. This is the original attack game. By assumption, attacker \mathcal{A} (t, ϵ, q) -breaks \mathcal{RS} when interacting with the challenger. We have,

$$\Pr[S_0] = \epsilon \tag{1}$$

Game₁. This game is like the previous one except that \mathcal{B} constructs the global parameters and the secret and public keys using the algorithms of the programmable hash function and the CDH challenge. First, \mathcal{B} randomly chooses: n elements $x_i \in_R \mathbb{Z}_p$ for $i = 1, \dots, n$, $l + 1$ elements $a'_0, a_1, \dots, a_l \in_R \{-1, 0, 1\}$, and

$l + 1$ elements $b_0, b_1, \dots, b_l \in_R \mathbb{Z}_p$. Let $a_0 = a'_0 - 1$. Then, for all $i \in \{1, \dots, n\}$ and $j \in \{0, \dots, l\}$ \mathcal{B} computes

$$g := \bar{g}^a, h := \bar{g}^b, PK_i := \bar{g}^{x_i}, u_j := h^{a_j} \bar{g}^{b_j}.$$

using the CDH challenge. Due to the properties of the multi-generator programmable hash function the distribution of the so computed values is equal to the distribution in the previous game. Thus,

$$\Pr[S_1] = \Pr[S_0]. \quad (2)$$

Game2. Now, \mathcal{B} simulates the challenger in the attack game by answering \mathcal{A} 's signature queries (m_j, R_j, e_j) . For convenience, let $a(m) = a_0 + \sum_{i=1}^l a_i m_i$ and $b(m) = b_0 + \sum_{i=1}^l b_i m_i$. Let $I[j] \subset \{1, \dots, n\}$ be the set of all indices $i \in \{1, \dots, n\}$ such that PK_i is a component of R_j . When receiving a signature query, \mathcal{B} at first tests whether $a(m_j) = 0$. In this case, \mathcal{B} outputs the failure signal F_1 and aborts. Otherwise \mathcal{B} chooses $r \in_R \mathbb{Z}_p$ and computes a collision $(d_{\bar{g}}, d_h)$ as $d_{\bar{g}} = h^r$ and $d_h = \bar{g}^r$. Note that by construction $e(d_{\bar{g}}, \bar{g}) = e(d_h, h)$.

The aim of \mathcal{B} is to compute $s_{n+1} \in \mathbb{G}$ and $|I[j]|$ values $s_i \in \mathbb{G}$ (for all $i \in I[j]$) such that

$$\prod_{i \in I[j]} e(s_i, PK_i) \cdot e(s_{n+1}, u(m_j)) = e(g, h)$$

or equivalently

$$e \left(s_{n+1}^{b(m_j)} \cdot \prod_{i \in I[j]} s_i^{x_i}, \bar{g} \right) = e \left(g s_{n+1}^{-a(m_j)}, h \right).$$

In the next step, \mathcal{B} chooses $y \in_R I[j]$ and for all $i \in I[j] \setminus \{y\}$ $s_i \in_R \mathbb{G}$. The values s_y and s_{n+1} are computed in the following way:

$$s_{n+1} = (gd_h^{-1})^{1/a(m_j)}, s_y = \left(d_{\bar{g}} \cdot s_{n+1}^{-b(m_j)} \cdot \prod_{i \in I[j] \setminus \{y\}} s_i^{-x_i} \right)^{1/x_y}.$$

\mathcal{B} outputs the ring signature $\sigma = (s_1, s_2, \dots, s_n, s_{n+1})$. The probability for \mathcal{B} to win this game is

$$\Pr[S_2] = \Pr[S_1 \wedge \bar{F}_1]. \quad (3)$$

Game3. In this game \mathcal{B} uses \mathcal{A} 's forgery $(m^*, R^*, \sigma^* = (s_1^*, s_2^*, \dots, s_{n+1}^*))$ to break the CDH assumption. Adversary \mathcal{B} at first checks whether $a(m^*) = 0$. If not, \mathcal{B} outputs the failure signal F_2 and aborts. We get that

$$\Pr[S_3] = \Pr[S_2 \wedge \bar{F}_2]. \quad (4)$$

Otherwise, \mathcal{B} computes the solution to the CDH problem as follows. Since $a(m^*) = 0$, we get that

$$e \left((s_{n+1}^*)^{b(m^*)} \cdot \prod_{i \in I[*]} (s_i^*)^{x_i}, \bar{g} \right) = e(g, h) \Leftrightarrow \bar{g}^{ab} = (s_{n+1}^*)^{b(m^*)} \cdot \prod_{i \in I[*]} (s_i^*)^{x_i}$$

what constitutes a solution to the CDH challenge.

We finally have

$$\Pr[S_3] = \epsilon_{\text{CDH}}. \quad (5)$$

Now, let us analyze the probabilities for an abort, i.e. for one of the events F_1 or F_2 to occur. Surely, the probability that both failure events do not occur depends on the number of signature queries q and the bit size l of the messages. Since, $u(m)$ is generated by the multi-generator programmable hash function as defined in Sect. 2.6, we can directly apply the results from [20] to show that

$$\Pr[\bar{F}_1 \wedge \bar{F}_2] \geq P_{q,l}.$$

Putting (1-5) together, we get that

$$\epsilon_{\text{CDH}} = \Pr[S_0 \wedge \bar{F}_1 \wedge \bar{F}_2] = \Pr[S_0 | \bar{F}_1 \wedge \bar{F}_2] \cdot \Pr[\bar{F}_1 \wedge \bar{F}_2] \geq \epsilon \cdot P_{q,l}$$

which proves Theorem 1.

4.2 Ring Anonymity

Theorem 2. *The ring signature scheme \mathcal{RS} is perfectly secure.*

We give an information theoretic argument. Given a ring signature, we have to show that each ring member could possibly have created it. Consider a ring signature on message m , that has been created using SK_z . We show that with the same probability it could have been created using SK_y with $y \neq z$. The proof is straight-forward.

Proof. Fix an arbitrary ring R of n public keys and choose two indices $y, z \in_R \{1, \dots, n\}$. Next, fix a random $m \in \{0, 1\}^l$ and $n - 1$ random values r_i with $i \in \{1, \dots, n + 1\} \setminus \{y, z\}$. We show that for any r_y there exists an r_z such that the final signatures generated by Sign with either (r_y, SK_z) or (r_z, SK_y) are equal. Since \mathbb{G} is a cyclic group with prime order p , there exists $t \in \mathbb{Z}_p$ and $b(M) = b_0 + \sum_{i=1}^l M_i b_i$ with $b_i \in \mathbb{Z}_p$ such that $h = g^t$ and $u(M) = g^{b(M)}$ for all $M \in \{1, \dots, n\}$.

Let the ring signature consist of all $s_i = g^{r_i}$ with $i \in \{1, \dots, n\} \setminus \{y, z\}$. Then, the remaining s_y, s_z are computed using SK_z and the Sign algorithm as

$$s_y = g^{r_y}, \quad s_z = \left(h \cdot \prod_{\substack{i=1 \\ i \neq z}}^n PK_i^{-r_i} \cdot \left(u_0 \prod_{j=1}^l u_j^{m_j} \right)^{-r_{n+1}} \right)^{1/x_z}.$$

Now, let $r_z = \frac{t - \sum_{i=1, i \neq z}^n r_i x_i - r_{n+1} b(m)}{x_z}$. Using SK_y we get $s_z = g^{r_z}$ and

$$s_y = \left(h \cdot \prod_{\substack{i=1 \\ i \neq y}}^n PK_i^{-r_i} \cdot \left(u_0 \prod_{j=1}^l u_j^{m_j} \right)^{-r_{n+1}} \right)^{1/x_y} = g^{r_y}$$

with $r_y = \frac{t - \sum_{i=1, i \neq y}^n r_i x_i - r_{n+1} b(m)}{x_y}$ what concludes the proof of Theorem 2.

4.3 Digital Signature Schemes

Our new proof technique can also be applied to other CDH based signature schemes. For example, we can surprisingly easily obtain as a special case ($n = 1$) of our ring signature scheme a variant \mathcal{S} of the Waters signature scheme that has distinct setup and sign algorithms but the same verification equation. We briefly compare it with the original scheme by Waters in Table 1. For completeness, we also describe a third variant \mathcal{S}_0 where the group hash function constitutes the public key of the user. Both schemes can easily be proven secure under the standard notion of security for digital signatures by Goldwasser, Micali and Rivest [18] by adapting the proof of Theorem 1.

Table 1. Comparison of the Waters signature scheme and \mathcal{S} and \mathcal{S}_0 . Unless not stated otherwise, all values are elements of \mathbb{G} . We set $u(m) = u_0 \prod_{i=1}^l u_i^{m_i}$ and $x(m) = x_0 + \sum_{i=1}^l x_i m_i$.

	Waters [31]	\mathcal{S}	\mathcal{S}_0
publ. params.	g_0, h, u_0, \dots, u_l	g, h, u_0, \dots, u_l	g_0, g, h
SK	h^x	$x \in \mathbb{Z}_p$	$x_0, \dots, x_l \in \mathbb{Z}_p$
PK	$g = g_0^x$	$g_0 = g^x$	$u_0 = g^{x_0}, \dots, u_l = g^{x_l}$
s_1	$h^x \cdot (u(m))^r$	$(h \cdot (u(m))^r)^{\frac{1}{x}}$	g^{-r}
s_2	g_0^{-r}	g^{-r}	$(hg_0^r)^{\frac{1}{x(m)}}$
verification		$e(s_1, g_0) \cdot e(s_2, u(m)) \stackrel{?}{=} e(g, h)$	

5 Conclusion

In this work, we presented an efficient and perfectly anonymous ring signature scheme that is secure under chosen subring attacks in symmetric bilinear groups with a secure bilinear map. Additionally, we developed a new technique for proving Waters-like signature schemes secure that uses $(1, poly)$ -programmable hash functions and results in very clean and compact security proofs. In our ring signature scheme, each public key consists of a single group element, while the signature size only accounts for $n + 1$ group elements, where n is the size of the ring. When compared to all other ring signature schemes that are proven secure in the standard model and do not assume ring re-use, this is extremely efficient.

Finally, we stress that using the generic transformation by Huang, Wong and Zhao [21] all presented schemes can be made strongly unforgeable, meaning that we also consider new signatures on previously queried messages as forgeries in the attack game. The overhead of this transformation is very small; the signature is extended by just a public key and an one-time signature, while no additional key material is required.

Acknowledgements. We would like to thank Tibor Jager and Maike Ritzenhofen for helpful comments on earlier versions of this work.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k -TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
3. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay (ed.) [29], pp. 409–426
4. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, Rabin (eds.) [19], pp. 60–79
5. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
6. Boneh, D., Mironov, I., Shoup, V.: A secure signature scheme from bilinear maps. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 98–110. Springer, Heidelberg (2003)
7. Boyen, X.: Mesh signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
8. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 132–145. ACM, New York (2004)
9. Camenisch, J., Van Herreweghen, E.: Design and implementation of the *demix* anonymous credential system. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 21–30. ACM, New York (2002)
10. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
11. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
12. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology. In: STOC, pp. 209–218 (1998), revisited (preliminary version)
13. 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)

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. Chow, S.S.M., Wei, V.K.-W., Liu, J.K., Yuen, T.H.: Ring signatures without random oracles. In: Lin, F.-C., Lee, D.-T., Lin, B.-S., Shieh, S., Jajodia, S. (eds.) ASIACCS, pp. 297–302. ACM, New York (2006)
16. Coron, J.-S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: Wagner (ed.) [30], pp. 1–20
17. 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)
18. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
19. Halevi, S., Rabin, T. (eds.): TCC 2006. LNCS, vol. 3876. Springer, Heidelberg (2006)
20. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner (ed.) [30], pp. 21–38
21. Huang, Q., Wong, D.S., Zhao, Y.: Generic transformation to strongly unforgeable signatures. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 1–17. Springer, Heidelberg (2007)
22. 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)
23. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay (ed.) [29], pp. 465–485
24. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, Rabin (eds.) [19], pp. 80–99
25. Persiano, G., Visconti, I.: An efficient and usable multi-show non-transferable anonymous credential system. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 196–211. Springer, Heidelberg (2004)
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. 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)
28. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs, November 30 (2004) (manuscript); Revised version from January 18 (2006/2004)
29. Vaudenay, S. (ed.): EUROCRYPT 2006. LNCS, vol. 4004. Springer, Heidelberg (2006)
30. Wagner, D. (ed.): CRYPTO 2008. LNCS, vol. 5157. Springer, Heidelberg (2008)
31. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A Proof of Lemma 1

Proof. By contradiction. Let $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ be our bilinear group. First, assume attacker \mathcal{A} can break the security of the bilinear map in time t with advantage at least ϵ . Then, algorithm \mathcal{B} can solve the CDH assumption in \mathbb{G} in time t with advantage ϵ by using \mathcal{A} as a black-box. Let $\bar{g}, \bar{g}^a, \bar{g}^b$ be \mathcal{B} 's CDH challenge in group \mathbb{G} . \mathcal{B} sets $\tilde{g} = \bar{g}^a$, $\tilde{g}' = \bar{g}^b$, and $\tilde{h} = \bar{g}$ and runs attacker \mathcal{A} on $(\tilde{g}, \tilde{g}', \tilde{h})$. As a result, \mathcal{A} outputs \tilde{h}' such that $e(\tilde{g}, \tilde{g}') = e(\tilde{h}, \tilde{h}')$. Since equivalently $e(\bar{g}^a, \bar{g}^b) = e(\bar{g}, \bar{h}')$, \tilde{h}' is a solution to the CDH problem.

Now, assume adversary \mathcal{A} (t, ϵ) -breaks the CDH assumption in \mathbb{G} . Let $\tilde{g}, \tilde{g}', \tilde{h} \in \mathbb{G}$, $\tilde{h} \neq 1_{\mathbb{G}}$ be \mathcal{B} 's challenge against the security of the bilinear map. Since \tilde{h} is a generator, there exist $a, b \in \mathbb{Z}_p$ such that $\tilde{h}^a = \tilde{g}$, and $\tilde{h}^b = \tilde{g}'$. \mathcal{B} runs \mathcal{A} on $\tilde{h}, \tilde{g}, \tilde{g}'$. Because \mathcal{A} outputs \tilde{h}^{ab} , we have that $e(\tilde{g}, \tilde{g}') = e(\tilde{h}, \tilde{h}^{ab})$, and thus \mathcal{A} 's output is a correct solution to \mathcal{B} 's challenge.

Practical Private Set Intersection Protocols with Linear Complexity

Emiliano De Cristofaro and Gene Tsudik

University of California, Irvine*

Abstract. The constantly increasing dependence on anytime-anywhere availability of data and the commensurately increasing fear of losing privacy motivate the need for privacy-preserving techniques. One interesting and common problem occurs when two parties need to privately compute an intersection of their respective sets of data. In doing so, one or both parties must obtain the intersection (if one exists), while neither should learn anything about other set elements. Although prior work has yielded a number of effective and elegant Private Set Intersection (**PSI**) techniques, the quest for efficiency is still underway. This paper explores some **PSI** variations and constructs several secure protocols that are appreciably more efficient than the state-of-the-art.

1 Introduction

In today's increasingly electronic world, privacy is an elusive and precious commodity. There are many realistic modern scenarios where private data must be shared among mutually suspicious entities. Consider the following examples:

1. A government agency needs to make sure that employees of its industrial contractor have no criminal records. Neither the agency nor the contractor are willing to disclose their respective data-sets (list of convicted felons and employees, respectively) but both would like to know the intersection, if any.
2. Two national law enforcement bodies (e.g., USA's FBI and UK's MI5) want to compare their respective databases of terrorist suspects. National privacy laws prevent them from revealing bulk data, however, by treaty, they are allowed to share information on suspects of common interest.
3. Two real estate companies would like to identify customers (e.g., homeowners) who are double-dealing, i.e., have signed exclusive contracts with both companies to assist them in selling their houses.
4. Federal tax authority wants to learn whether any suspected tax evaders have any accounts with a certain foreign bank and, if so, obtain their account records and details. The bank's domicile forbids wholesale disclosure of account holders and the tax authority clearly can not reveal its list of suspects.
5. Department of homeland security (DHS) wants to check its list of terrorist suspects against the passenger manifest of a flight operated by a foreign air carrier. Neither party is willing to reveal its information, however, if there is a (non-empty) intersection, DHS will not give the flight permission to land.

* This research was supported by the U.S. Intelligence Advanced Research Projects Activity (IARPA) under grant #: FA8750-09-2-0071.

Such scenarios provide with interesting examples that motivate the need for privacy-preserving set operations, in particular, set intersection protocols. Such protocols are especially useful whenever one or both parties who do not fully trust each other must compute an intersection of their respective sets (or some function thereof). As discussed in Section 4 below, prior work has yielded a number of interesting techniques. As usually happens in applied cryptography, the next step (and the current quest) is to improve efficiency. To this end, this paper’s main goal is to consider several flavors of Private Set Intersection (**PSI**) and construct provably secure protocols that are more efficient than the state-of-the-art.

2 **PSI** Flavors

Generally speaking, **Private Set Intersection (PSI)** is a cryptographic protocol that involves two players, Alice and Bob, each with a private set. Their goal is to compute the intersection of their respective sets, such that minimal information is revealed in the process. In other words, Alice and Bob should learn the elements (if any) common to both sets and nothing (or as little as possible) else. This can be a mutual process where, ideally, neither party has any advantage over the other. Examples 1-3 above require mutual **PSI**. In a one-way version of **PSI**, Alice learns the intersection the two sets, however, Bob learns (close to) nothing. Examples 4 and 5 correspond to one-way **PSI**.

Since mutual **PSI** can be easily obtained by two instantiations of one-way **PSI** (assuming that no player aborts the protocol prematurely), in the remainder of this paper we focus on the latter. Hereafter, the term **PSI** denotes the one-way version and, instead of proverbial Alice and Bob, we use client (C , i.e., the entity receiving the intersection) and server (S) to refer to the protocol participants.

One natural extension is what we call **PSI with Data Transfer** or **PSI-DT**. In this setting, one or both parties have data associated with each element in the set e.g., a database record. In **PSI-DT**, data associated with each element in the intersection must be transferred to one or both parties, depending whether mutual or one-way version of **PSI** is used. Example 4 corresponds to **PSI-DT**. It is also easy to see that **PSI-DT** is quite appealing in terms of actual database (rather than plain set) applications.

Another twist on **PSI** is the authorized version – **APSI** – where each element in the client set must be authorized (signed) by some recognized and mutually trusted authority. This requirement could be applicable to Examples 2 and 4. In the former, one or both agencies might want to make sure that names of terrorist suspects held by its counterpart are duly authorized by the country’s top judiciary. In example 4, the bank could demand that each suspected tax cheat be pre-vetted by some international body, e.g., Interpol. In general, the main difference between **PSI** and **APSI** is that, in the former, the inputs of one or both parties might be arbitrarily chosen, i.e., frivolous.

Clearly, other more interesting or more exotic variations are possible, e.g., the notion of *group PSI* with its many types of possible outputs. However, we limit the scope of this paper to the **PSI** flavors described above.

3 Roadmap

In contrast to prior work, we do not start with constructing **PSI** protocols and piling on extra features later. Instead, somewhat counter-intuitively, we begin with prior work on a specific type of protocols – called Privacy-preserving Policy-based Information Transfer (PPIT) – that provide **APSI-DT** (one-way authorized private set intersection with data transfer) for the case where one party has a set of size one. PPIT matches a typical database query scenario where client has a single keyword or a record identifier and server has a database.

We start by seeing how some previously-proposed PPIT protocols can be trivially extended into inefficient **PSI** and **APSI** protocols, with and without data transfer. We then construct several efficient (and less trivial) provably secure **PSI** and **APSI** protocols that incur **linear** computation and communication overhead. Concretely, this work makes several contributions:

1. We evaluate and compare existing **PSI** and **APSI** protocols in terms of efficiency (computation and communication overhead), security model (random oracle vs standard) and adversary type (honest-but-curious vs malicious).
2. We investigate whether **APSI** protocols can yield (efficient) **PSI** counterparts.
3. We present an **APSI** protocol and its **PSI** more efficient than prior work.
4. We construct another **PSI** protocol geared for scenarios where the server can perform some pre-computation and/or the client is computationally weak.

4 Prior Work

This section overviews relevant prior results, which fall into several categories: (1) **PSI** protocols, (2) OPRF constructs, and (3) **APSI** variations. Also, we note that most **PSI** variations can be realized via general secure multi-party techniques. However, it is usually far more efficient to have dedicated protocols; which is the direction we pursue in this paper.

PSI Protocols. The work by Freedman, et Al. (FNP) [15] addressed the problem of private set intersection by means of Oblivious Polynomial Evaluation (OPE). In [15], the idea is to represent of a set as a polynomial, and the elements of the set as its roots. Specifically, a client C represents elements in its private set, $\mathcal{C} = (c_1, \dots, c_v)$, as the roots of a v -degree polynomial over a ring R , i.e. $f = \prod_{i=1}^v (t - c_i) = \sum_{i=0}^k \alpha_i t^i$. Then, assuming pk_C to be C 's public key of any additively homomorphic cryptosystem (such as Paillier [22]), C encrypts the coefficients with pk_C , and sends them to server S . S 's private set is denoted with $\mathcal{S} = (s_1, \dots, s_w)$. S evaluates f at each $s_j \in \mathcal{S}$ homomorphically. Note that $f(s_j) = 0$ if and only if $s_j \in \mathcal{C} \cap \mathcal{S}$. Hence S , for each $s_j \in \mathcal{S} = (s_1, \dots, s_w)$ returns $u_j = E(r_j f(s_j) + s_j)$ to C (where r_j is chosen at random). If $s_j \in \mathcal{C} \cap \mathcal{S}$ then C learns s_j upon decrypting. If $s_j \notin \mathcal{C} \cap \mathcal{S}$ then u_j decrypts to a random value. Therefore, the number of server's operations is related to the evaluation of client's encrypted polynomial, with v coefficients, on w points in \mathcal{S} . Using Horner's rule (and assuming Paillier encryption) this would take $O(vw)$ of m -bit mod 2048-bit exponentiations, where m is the number of bits needed for representing

each entry. On the other hand, the number of client operations is $O(v + w)$, i.e., 1024-bit exponentiations mod 2048 bits. However, certain optimizations can be applied to reduce the total number of server's exponentiations to $O(w \log(\log(v)))$. Such protocol is proved secure against an Honest-but-Curious (HbC) adversary in the standard model, and can be extended for malicious adversaries in the Random Oracle Model (ROM), with an increased cost.

Subsequently, the work by Kissner and Song (KS) [20] has proposed OPE-based protocols that apply to several set operations (e.g., union, intersection, etc.) and may involve more than two players. [20] contributes constructions secure in the standard model against HbC (with similar complexity to [15]) and also malicious adversaries. The latter incurs into quadratic computation overhead, i.e., $O(wv)$, and involves expensive zero-knowledge proofs, whereas a more efficient construction has been recently proposed by [9], specifically to $O(wk^2 \log^2(v))$ communication and to $O(wvk \log(v) + wk^2 \log^2(v))$ computation complexity—being k the security parameter.

Protocols based on Oblivious Pseudo Random Functions. Other constructs rely on so-called Oblivious Pseudo-Random Functions (OPRFs), introduced in [14]. An OPRF is a two-party protocol (between a sender and a receiver) that securely computes a pseudorandom function $f_k(\cdot)$ on key k contributed by the sender and input x contributed by the receiver, such that the former learns nothing from the interaction and the latter learns only the value $f_k(x)$.

OPRF-based PSI protocols work as follows: Server S holds a secret random key k . For each $s_j \in \mathcal{S}$ (of size w), S precomputes $u_j = f_k(s_j)$, and publishes (sends to client) the set $\mathcal{U} = \{u_1, \dots, u_w\}$. Then, C and S engage in an OPRF computation of $f_k(c_i)$ for each $c_i \in \mathcal{C}$ (of size v), such that S learns nothing about \mathcal{C} (except the size) and C learns $f_k(c_i)$. Finally, C learns that $c_i \in \mathcal{C} \cap \mathcal{S}$ if and only if $f_k(c_i) \in \mathcal{U}$.

The idea of using OPRFs for PSI protocols is due to Hazay and Lindell [16]. Their protocol is secure in the standard model in the presence of a malicious server and an HbC client. It has been since improved by Jarecki and Liu [18], who proposed a protocol secure in the standard model against both malicious parties, based on the Decisional q-Diffie-Hellman Inversion assumption, in the Common Reference String (CRS) model, where a safe RSA modulus must be pre-generated by a trusted party. Encryption operations are performed using an additively homomorphic encryption scheme, such as the one presented by Camenisch and Shoup, CS for short [7]. As pointed out in [18], such solution can be further optimized, inspiring to the concurrent work by Bellenkiy, et Al. [1]. In fact, the OPRF construction could work on groups with a 160-bit prime order unrelated to the RSA modulus, instead of the more expensive composite order groups. Assuming such improved construction, [18] incurs in the following computational complexity. The server S needs to perform $O(w)$ PRF evaluations with w inputs, more precisely $O(w)$ modular exponentiations of m -bit exponents (where m is the number of bits needed to represent each entry) mod n^2 (e.g., 2048 bits). Moreover, the client C needs to compute $O(v)$ CS encryptions (i.e., $O(v)$ m -bit exponentiations mod 2048 bits, plus $O(v)$ 1024-bit exponentiations mod 1024 bits). Whereas, S computes (online) $O(v)$ CS decryptions, (i.e., $O(v)$ 1024-bit exponentiations mod 2048 bits). As discussed in [18], complexity in the malicious model grows by a factor of 2.

Finally, the work-in-progress in [19] leverages an idea similar to the OPRF, namely the *Unpredictable Function* (UPF) $f_k(x) = (H(x))^k$ in the Random Oracle Model. Authors construct a two-party computation of this function, with a server S contributing the key k and a client C the argument x : C picks a random exponent α and sends $y = (H(x))^\alpha$ to S , that replies with $z = y^k$, so that C recovers $f_k(x) = z^{1/\alpha}$. Note that random exponents, given that the hash functions are carefully chosen, can be taken from a subgroup (e.g., they can be 160-bits long). Similarly to OPRF-based solutions, the UPF can then be used to implement the secure computation of *Adaptive Set Intersection*, under the *One-More-Gap-DH* assumption in ROM [2]. We remark that this solution is similar to the ones given before by [17] and [13]. However, in such works, security is only superficially analyzed and no proof is provided, whereas [19] provides security also against malicious players. The computational complexity of the UPF-based PSI (in presence of honest-but-curious adversaries) amounts to $O(w + v)$ (resp. $O(v)$) exponentiations with short exponents (e.g., 160-bit mod 1024-bit).

APSI Protocols. We now briefly review related work in Authorized Private Set Intersection protocols. Recently, a new PSI-related concept was introduced, called *Privacy-preserving Policy-based Information Transfer* (PPIT) [11]. It is targeted for scenarios where a client holding an authorization (i.e., a signature by a trusted authority) on some identifier needs to retrieve information matching that identifier from a server, such that: (1) the client only gets the information it is entitled to, and (2) the server knows that the client is duly authorized to obtain information but does not learn what information is retrieved. Besides requiring the client to be authorized, PPIT is focused on the situation where the client holds a single identifier, i.e., PPIT offers **APSI** where Alice (client) has a set of size one. [11] gives three PPIT protocols, based respectively on: RSA [23], Schnorr [24], and Identity-based Encryption (IBE) [4]. In this paper, we only discuss RSA-PPIT since it serves as a starting point for the work in this paper. In RSA-PPIT, client's authorizations are essentially RSA signatures on a set of record identifiers. As shown in [11], it is easy to extend PPIT to support the case of the client holding multiple authorizations and thus obtain a full-blown **APSI** protocol. The result is also secure in ROM for honest-but-curious parties. However, the complexity (both communication and computation) becomes quadratic. We will review such construction in Section 5.3.

Another recent result [6] has addressed a problem similar to PPIT, by means of an IBE-based technique inspired by Public-Key Encryption with Keyword Search (PEKS) [3]. It enhances PEKS by introducing a *Committed Blind Anonymous* IBE scheme. With such a scheme, the client privately obtains trapdoors from the CA, hence not revealing anything about its inputs to the CA (unlike PPIT). Nevertheless, the client commits to the inputs, so that the CA can later ask the client to prove statements on them. Although this scheme does not require the Random Oracle Model, its efficiency is much lower than PPIT. First, whereas IBE-PPIT uses Boneh-Franklin IBE [4], the underlying IBE scheme is a modification of Boyen-Waters (BW) IBE [5] which is less time and space efficient. The server has to compute $O(w)$ (BW) encryptions (each requiring 6 exponentiations and a representation of 6 group elements). Furthermore, the client has to test each $O(w)$ PEKS against its $O(v)$ trapdoors, hence performing $O(vw)$ (BW) decryptions (each requiring 5 bilinear map operations).

Finally, Camenisch and Zaverucha [8] have introduced the notion of *Certified Sets* to the private set intersection problem. This allows a trusted third party to ensure that all protocol inputs are valid and bound to each protocol participant. The proposed protocol builds upon oblivious polynomial evaluation and achieves asymptotic computation (quadratic) and communication overhead similar to that of FNP [15] and KS [20].

5 Towards Efficient PSI and APSI Protocols

In this section, we explore the design of efficient PSI and APSI. Before proceeding to the actual protocols, we provide some definitions and assumptions.

5.1 Preliminaries

Recall that PSI involves two parties: client and server.

Definition 1. *PSI consists of two algorithms: $\{\text{Setup}, \text{Interaction}\}$. Setup: a process wherein all global/public parameters are selected. Interaction: a protocol between client and server that results in the client obtaining the intersection of two sets.*

APSI involves three parties: client, server and (off-line) CA.

Definition 2. *APSI is a tuple of three algorithms: $\{\text{Setup}, \text{Authorize}, \text{Interaction}\}$. Setup: a process wherein all global/public parameters are selected. Authorize : a protocol between client and CA resulting in client committing to its input set and CA issuing authorizations (signatures), one for each element of the set. Interaction: a protocol between client and server that results in the client obtaining the intersection of two sets.*

The following assumptions are made throughout. In APSI, we assume that CA does not behave maliciously. Also, server is honest-but-curious, however, client might not have authorizations for all elements in its set. Finally, in PSI we assume that both client and server are honest-but-curious, leaving modified constructions and proofs in the malicious model as part of future work.

5.2 Security Properties

We now informally describe security requirements for PSI and APSI.

Correctness. A PSI scheme is *correct* if, at the end of *Interaction*, client outputs the exact (possibly empty) intersection of the two respective sets.

Server Privacy. Informally, a PSI scheme is server-private if the client learns no information (except the upper bound on size) about the subset of elements on the server that are NOT in the intersection of their respective sets.

Client Privacy. Informally, client privacy (in either PSI or APSI) means that no information is leaked about client's set elements to a malicious server, except the upper bound on the client's set size.

Client Unlinkability (optional). Informally, client unlinkability means that a malicious server cannot tell if any two instances of *Interaction* are related, i.e., executed on the same inputs by the client.

Table 1. Notation

$a \leftarrow A$	variable a is chosen uniformly at random from set A
τ	security parameter
n, e, d	RSA modulus, public and private exponents
g	group generator; exact group depends on context
p, q	large primes, where $q = k(p - 1)$ for some integer k
$H()$	full-domain hash function
$H'()$	regular cryptographic hash function: $H' : \{0, 1\}^* \rightarrow \{0, 1\}^\tau$
\mathcal{C}, \mathcal{S}	client's and server's sets, respectively
v, w	sizes of \mathcal{C} and \mathcal{S} , respectively
$i \in [1, v], j \in [1, w]$	indices of elements of \mathcal{C} and \mathcal{S} , respectively
c_i, s_j	i -th and j -th elements of \mathcal{C} and \mathcal{S} , respectively
h_{c_i}, h_{s_j}	$H(c_i)$ and $H(s_j)$, respectively
$R_{c:i}, R_{s:j}$	i -th and j -th random value generated by client and server, respectively

Server Unlinkability (optional). Informally, server unlinkability means that a malicious client cannot tell if any two instances of *Interaction* are related, i.e., executed on the same inputs by the server.

For **APSI**, the Correctness and Server Privacy requirements are amended as follows:

Correctness (APSI). An **APSI** scheme is *correct* if, at the end of *Interaction*, client outputs the exact (possibly empty) intersection of the two respective sets and each element in that intersection has been previously authorized by CA via *Authorize*.

Server Privacy (APSI). Informally, an **APSI** scheme is server-private if the client learns no information (except the upper bound on size) about the subset of elements on the server that are NOT in the intersection of their respective sets (where client's set contains only authorizations obtained via *Authorize*).

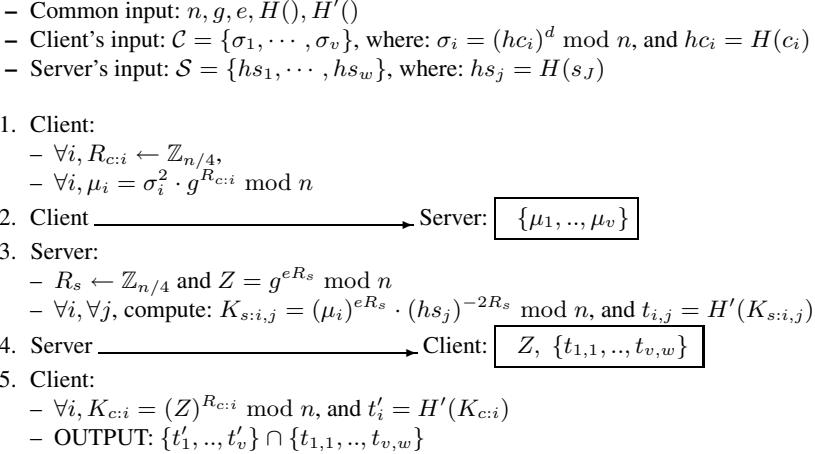
5.3 Baseline: **APSI** from RSA-PPIT

The starting point for our design is an **APSI** protocol derived from RSA-PPIT [11]. This protocol is only sketched out in [11]; since our new protocols are loosely based on it, we specify it in Fig.1. Actually, the protocol in [11] is **APSI-DT**; however, for ease of illustration we omit the data transfer component at this point. Also, all **PSI** and **APSI** protocols in this paper include only the *Interaction* component; *Setup* and *Authorize* (if applicable) are both intuitive and trivial. Our notation is reflected in Table 1. It is easy to see that this protocol is correct, since: for any (σ_i, c_i) held by the client and s_j held by the server, if: (1) σ_i is a genuine CA's signature on c_i , and (2) $c_i = s_j$ (hence, $h_{c_i} = h_{s_j}$):

$$\begin{aligned} K_{c:i} &= (Z)^{R_{c:i}} = g^{eR_s \cdot R_{c:i}} \\ K_{s:i,j} &= (\mu_i)^{eR_s} \cdot (h_{s_j})^{-2R_s} = (\sigma_i^2 \cdot g^{R_{c:i}})^{eR_s} \cdot (h_{s_j})^{-2R_s} = \\ &= ((h_{c_i})^{d2} \cdot g^{R_{c:i}})^{eR_s} \cdot (h_{s_j})^{-2R_s} = h_{c_i}^{2R_s} \cdot g^{eR_s \cdot R_{c:i}} \cdot h_{s_j}^{-2R_s} = g^{eR_s \cdot R_{c:i}} \end{aligned}$$

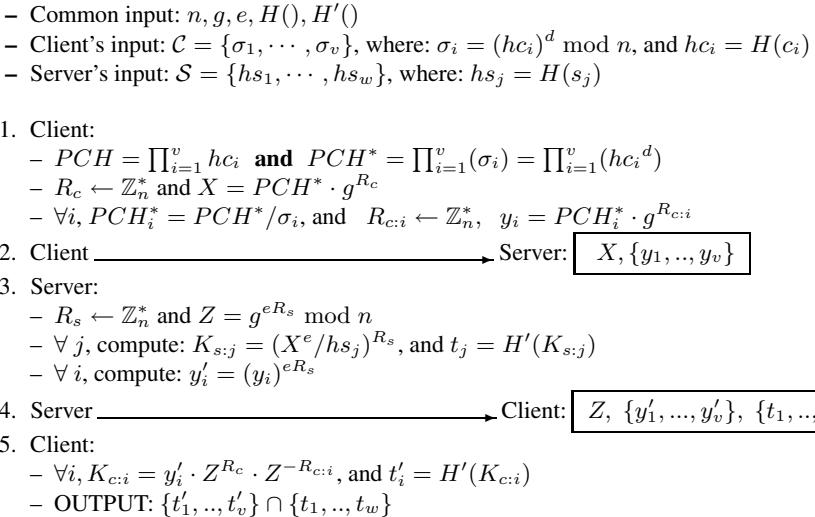
We point out that the protocol in Fig.1 incurs in quadratic computation overhead by the server and quadratic communication.

It is possible to reduce the number of on-line exponentiations on the server to $O(v)$ by precomputing all values $(h_{s_j})^{-2R_s}$ in Step 3. Nonetheless, the number of multiplications needed to compute all $K_{s:i,j}$ would still remain quadratic, i.e., $O(vw)$, as would the communication overhead.

**Fig. 1.** APSI Protocol derived from RSA-PPIT

5.4 APSI with Linear Costs

Although the trivial realization of APSI obtained from RSA-PPIT is relatively inefficient, we now show how to use it to derive an efficient protocol, shown in Fig.2.

**Fig. 2.** APSI Protocol with linear complexity

This protocol incurs **linear** computation (for both parties) and communication complexity. Specifically, the client performs $O(v)$ exponentiations and the server – $O(v + w)$. Communication is dominated by server's reply in Step 4 – $O(v + w)$. To see that

the protocol is correct, observe that, for any (σ_i, c_i) held by the client and s_j held by the server, if: (1) σ_i is a genuine CA's signature on c_i , and (2) $c_i = s_j$, hence, $hc_i = hs_j$:

$$\begin{aligned} K_{c:i} &= y'_i \cdot Z^{R_c} \cdot Z^{-R_{c:i}} = (PCH_i^*)^{eR_s} \cdot g^{R_{c:i}eR_s} \cdot g^{eR_s R_c} \cdot g^{-eR_s R_{c:i}} = \\ &= (PCH_i)^{R_s} \cdot g^{eR_c R_s} = (PCH_i)^{R_s} \cdot g^{eR_c R_s} \\ K_{s:j} &= (X^e / hs_j)^{R_s} = [(PCH^* \cdot g^{R_c})^e / hs_j]^{R_s} = \\ &= (PCH / hs_j \cdot g^{eR_c})^{R_s} = (PCH_i)^{R_s} \cdot g^{eR_c R_s} \end{aligned}$$

Note that: $(PCH^*)^e = \prod_{i=1}^v (\sigma_i^e) = PCH$ and: $(PCH_i^*)^e = PCH_i$

We claim that the **APSI** Protocol in Fig. 2 is a: (1) Server-Private (**APSI**), (2) Client-Private, (3) Client-Unlinkable, and (4) Server-Unlinkable **APSI**. (See Appendix B).

5.5 Deriving Efficient **PSI**

We now convert the above **APSI** protocol into a **PSI** variant, shown in Fig.3. In doing so, the main change is the obviated need for the RSA setting. Instead, the protocol operates in Z_p where p is a large prime and q is a large divisor of $p - 1$. This change makes the protocol more efficient, especially, because of smaller ($|q|$ -size) exponents. Nonetheless, the basic complexity remains the same: linear communication overhead – $O(v + w)$, and linear computation – $O(v + w)$ for the server and $O(v)$ for the client. However, we note that, in Step 3b, the server can precompute all values of the form: $(hs_j)^{-R_s}$. Thus, the cost of computing all $K_{s:j}$ values can be reduced to $O(w)$ multiplications (from $O(w)$ exponentiations). In fact, the same optimization applies to the protocol in Fig.2. Correctness of the protocol is self-evident, since its essential operation is very similar to that of the **APSI** variant.

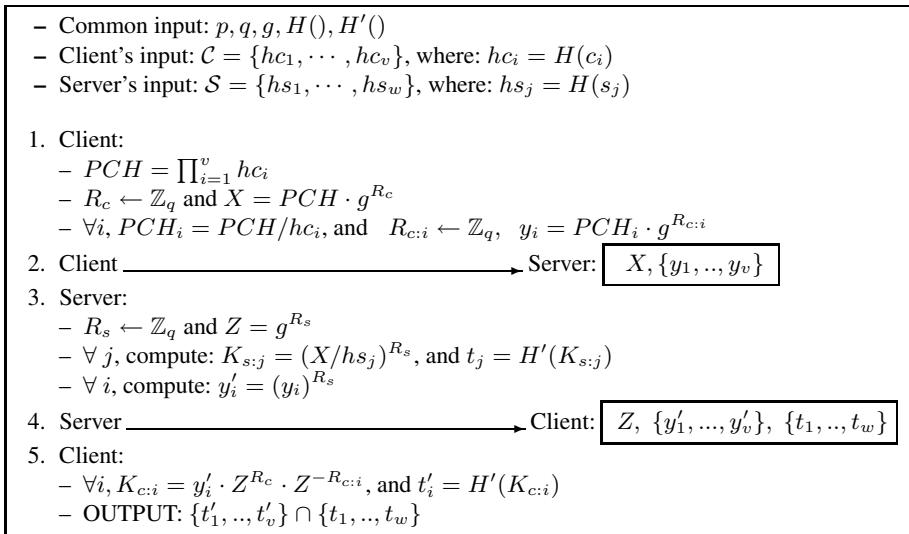


Fig. 3. **PSI** Protocol with linear complexity

We defer formal proofs for the above **PSI** protocol to extended version of the paper [12]. Proofs basically mirror the proofs of the protocol constructed in the next section (see Appendix C). Simulations of the two schemes follow the same approach, except that, while proofs in Appendix C rely on the One-More-RSA assumption, privacy of protocol in Fig. 3 is based on the One-More-Gap-DH assumption. With the exception of authorization, security and privacy features of this protocol are the same as that of its **APSI** counterpart described above.

Note that the our work-in-progress proofs against fully malicious players for protocols in Figures 2 and 3 seem to depend on the product of hashes, i.e., the *PCH* structure. However, for HbC adversaries these protocols can be described in a simplified version reported in Appendix D, for the sake of completeness. We present only the **PSI** version, since the description of its **APSI** counterpart is straightforward.

5.6 More Efficient **PSI**

Although efficient in principle, the **PSI** protocol in Fig.3 is *sub-optimal* for application scenarios where the client is a resource-poor device, e.g., a PDA or a cell-phone. In other words, $O(v)$ exponentiations might still represent a fairly heavy burden. Also, if the server's set is very large, overhead incurred by $O(w)$ modular multiplications might be substantial.

To this end, we present an even more efficient **PSI** protocol (see Fig. 4) where the client does not perform any modular exponentiations on-line. Instead, it only needs $O(v)$ on-line modular multiplications (Step 7). Also, server's on-line computation overhead is reduced to $O(v)$ exponentiations in Step 5. Server precomputation in Step 1

- Common input: $n, e, H(), H'()$
- Client's input: $\mathcal{C} = \{hc_1, \dots, hc_v\}$, where: $hc_i = H(c_i)$
- Server's input: $d, \mathcal{S} = \{hs_1, \dots, hs_w\}$, where: $hs_j = H(s_j)$

OFF-LINE:

1. Server:
 - $\forall j$, compute: $K_{s:j} = (hs_j)^d \bmod n$ and $t_j = H'(K_{s:j})$
2. Client:
 - $\forall i$, compute: $R_{c:i} \leftarrow \mathbb{Z}_n^*$ and $y_i = hc_i \cdot (R_{c:i})^e \bmod n$

ON-LINE:

3. Client $\xrightarrow{\quad}$ Server: $\{y_1, \dots, y_v\}$
4. Server:
 - $\forall i$, compute: $y'_i = (y_i)^d \bmod n$
5. Server $\xrightarrow{\quad}$ Client: $\{y'_1, \dots, y'_v\}, \{t_1, \dots, t_w\}$
6. Client:
 - $\forall i$, compute: $K_{c:i} = y'_i / R_{c:i}$ and $t'_i = H'(K_{c:i})$
 - OUTPUT: $\{t'_1, \dots, t'_v\} \cap \{t_1, \dots, t_w\}$

Fig. 4. Blind RSA-based **PSI** Protocol with linear complexity

amounts to w exponentiations – RSA signatures. Client precomputation in Step 2 involves $O(v)$ multiplications, since, as is well-known that, e can be a small integer.

The main idea behind this protocol comes from the Ogata and Kurosawa's adaptive Oblivious Keyword Search [21]. However, we adapt it for the PSI scenario: instead of encrypting a string of 0's the server reveals the key as the hash of the signature for all elements in her set. We show that the resulting protocol in Fig. 4 is a: (1) Server-Private, (2) Client-Private, and (3) Client-Unlinkable PSI (see Appendix C).

Although this protocol uses the RSA setting, RSA parameters are initialized *a priori* by the server. This is in contrast to the protocol in Fig. 2 where the CA sets up RSA parameters. To see that the present protocol is correct, consider that: $K_{s:j} = (hs_j)^d$ in Step 1, and, in Step 6:

$$K_{c:i} = y'_i / R_{c:i} = (hc_i \cdot (R_{c:i})^e)^d / R_{c:i} = (hc_i)^d \implies K_{c:i} = K_{s:j} \text{ iff } hc_i = hs_j$$

Drawbacks: although very efficient, this PSI protocol has some issues. First, it is unclear how to convert it into an APSI version. Second, if precomputation is somehow impossible, its performance becomes worse than that of the PSI protocol in Fig. 3, since the latter uses much shorter exponents at the server side. Privacy features of this protocol also differ from others discussed above. In particular, it lacks server unlinkability. (Recall that this feature is relevant only if the protocol is run multiple times.) We note that, in Step 1 the server computes tags of the form $t_j = H'(hs_j)^d$. Consequently, running the protocol twice allows the client to observe any and all changes in the server's set.

There are several ways of patching the protocol to provide this missing feature. One is for the server to select a new set of RSA parameters for each protocol instance. This would be a time-consuming extra step at the start of the protocol; albeit, with precomputation, no extra on-line work would be required from the server. On the other hand, the client would need to be informed of the new RSA public key (e, n) before Step 2, which means that, at the very least (using $e = 3$), v multiplications in Step 2 would have to be done on-line. Also, two additional initial messages would be necessary: one from the client – to “wake up” the server, and the other – from the server to the client bearing the new RSA public key and (perhaps) $\{t_1, \dots, t_w\}$, thus saving space in the last message. Another simple way of providing server unlinkability is to change the hash function $H()$ for the server each protocol instance. If we assume that the client and server maintain either a common protocol counter (monotonically increasing and non-wrapping) or sufficiently synchronized clocks, it is easy to select/index a distinct hash function based on such unique and common values. One advantage of this approach is that we no longer need the two extra initial messages.

5.7 From PSI (APSI) to PSI-DT (APSI-DT)

It is easy to add data transfer functionality to the protocols in Fig. 1, 2, 3 and 4, and provide APSI-DT and PSI-DT. Following the approach outlined in [11], we assume that an additional secure cryptographic hash function $H'' : \{0, 1\}^* \rightarrow \{0, 1\}^\tau$ is chosen during setup. In all aforementioned protocols, we then use H'' to derive a symmetric key for a semantically secure symmetric cipher, such as AES [10]. For every j , server computes $k_{s:j} = H''(K_{s:j})$ and encrypts associated data using a distinct key $k_{s:j}$. For its part, the client, for every i , computes $k_{c:i} = H''(K_{c:i})$ and decrypts ciphertexts

corresponding to the matching tag. (Note that $k_{s:j} = k_{c:i}$ iff $s_j = c_i$ and so $t_j = t_i$). As long as the underlying encryption scheme is semantically secure, this extension does not affect the security or privacy arguments for any protocol discussed thus far.

5.8 Evaluation

We now highlight the differences between existing **PSI** techniques and protocols proposed in this paper. We focus on performance in terms of server and client computation and communication complexities. We use w and v to denote the number of elements in the server's and client's sets, respectively. Let m be the number of bits needed to represent each element. We count only the number of *online* operations. The results are summarized in Table 2 and compared choosing parameters that achieve similar degrees of security. The Table also includes communication overhead, for completeness.

Table 2. Performance Comparison of **PSI** and **APSI** protocols

Protocol	Model	Adv	Commun.	Server Prec	Server Ops	Client Ops	Mod Bits
APSI [6]	Std	Mal	$O(w)$	-	$O(w)$ encrs in [5]	$O(vw)$ decrs in [5]	
APSI Fig.1	ROM	HbC	$O(vw)$	$O(w)$ 1024-bit exps	$O(v)$ 1024-bit exps $O(vw)$ mults	$O(v)$ 1024-bit exps	1024
APSI Fig.2	ROM	HbC	$O(v+w)$	$O(w)$ 1024-bit exps	$O(v)$ 1024-bit exps	$O(v)$ 1024-bit exps	1024
PSI [15]	Std	HbC	$O(v+w)$	-	$O(vw)$ m -bit exps	$O(v + w)$ 1024-bit exps	2048
PSI [18]	Std	HbC	$O(v+w)$	$O(w)$ 1024-bit exps mod 1024 exps	$O(v)$ 1024-bit exps mod 2048 exps	$O(v)$ 1024 mod 1024-bit m -bit mod 2048-bit exps	1024/ 2048
PSI [19]	ROM	HbC	$O(v+w)$	$O(w)$ 160-bit exps	$O(v)$ 160-bit exps	$O(v)$ 160-bit exps	1024
PSI Fig.3	ROM	HbC	$O(v+w)$	$O(w)$ 160-bit exps $O(w)$ mults	$O(v)$ 160-bit exps	$O(v)$ 160-bit exps	1024
PSI Fig.4	ROM	HbC	$O(v+w)$	$O(w)$ 1024-bit exps	$O(v)$ 1024-bit exps	$O(v)$ 1024-bit mults	1024

We remark that: (1), each encryption in [5] (i.e., Boyen-Waters' IBE) requires 6 exponentiations and a representation of 6 group elements, and each decryption requires 5 bilinear map operations, and (2), the complexity for the **PSI** solution against Malicious model in [18] and [19] grows by a factor of 2. All protocols proposed in this paper have been implemented in ANSI C (using the well-known OpenSSL library) and tested on a Dell Precision PC on a 2.33GHz CPU and 8GB RAM. The prototype's code is available upon request. To confirm the claimed efficiency of our protocols, we compared on-line run-times of our protocols to those of prior work. In case a solution provides security both against HbC and malicious adversary, we implement the former. We omit run-times for operations that can be precomputed. We also do not measure all prior techniques discussed in Section 4, whereas we pick only the three that offer the best performance: the **APSI** adaptation of RSA-PPIT [11] in Fig.1, and **PSI**'s from [18] and [19]. We remark that since the efficiency of [18] is influenced by records' length, we assume a conservative stance and we choose items to be 160-bits long, similar to the output of a hash function.

Measured *online* computation overhead for the tested protocols is reflected in Table 3. As the results illustrate, among **APSI** protocols, the one in Fig.2 performs noticeably better than its PPIT-based counterpart from [11] when both server and client have sets of size 5,000 (and this advantage accelerates for larger set sizes). Looking at

Table 3. On-line computation overhead (in ms)

Player	Server	Client	Server	Client	Server	Client
Set size	5,000	1	1	5,000	5,000	5,000
APSI Fig.1	20	5	12,710	24,407	99,118	24,159
APSI Fig.2	23	10	12,228	25,769	12,037	25,959
PSI [18]	5	24	27,654	118,676	27,862	118,947
PSI [19]	0	1	2,029	4,227	2,108	4,249
PSI Fig.3	19	1	2,145	5,502	2,072	5,344
PSI Fig.4	1	0	4,651	1,407	4,662	1,422

PSI protocols, the toss-up is between protocols in Fig. 3 and 4; the choice of one or the other depends on whether client or server overhead is more important. If client is a weak device, the blind-RSA-based protocol in Fig.4 is a better bet. Otherwise, if server burden must be minimized, we opt for the protocol of Fig.3.

6 Conclusions

In this paper, we proposed efficient protocols for plain and authorized private set intersection (PSI and APSI). Proposed protocols offer appreciably better efficiency than prior results. The choice between them depends on whether there is a need for client authorization and/or server unlinkability, as well as on server's ability to engage in pre-computation. Our efficiency claims are supported by experiments with prototype implementations. Future work includes analysis of our protocols against malicious parties, as well as extensions to a group setting.

Acknowledgements. We would like to thank Nikita Borisov, Stanislaw Jarecki, Xiaomin Liu, Markulf Kohlweiss, and Jiye Kim for the helpful discussion.

References

1. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable Proofs and Delegatable Anonymous Credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
2. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. Journal of Cryptology 16(3), 185–215 (2008)
3. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
4. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
5. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
6. Camenisch, J., Kohlweiss, M., Rial, A., Sheedy, C.: Blind and Anonymous Identity-Based Encryption and Authorised Private Searches on Public Key Encrypted Data. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 196–214. Springer, Heidelberg (2009)

7. 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)
8. Camenisch, J., Zaverucha, G.: Private intersection of certified sets. In: Financial Cryptography and Data Security 2009 (2009)
9. 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)
10. Daeman, J., Rijmen, V.: AES proposal: Rijndael (1999)
11. De Cristofaro, E., Jarecki, S., Kim, J., Tsudik, G.: Privacy-Preserving Policy-Based Information Transfer. In: Goldberg, I., Atallah, M.J. (eds.) Privacy Enhancing Technologies. LNCS, vol. 5672, pp. 164–184. Springer, Heidelberg (2009)
12. De Cristofaro, E., Tsudik, G.: Practical Private Set Intersection Protocols. In: Cryptology ePrint Archive (2009), <http://eprint.iacr.org/2009/491.pdf>
13. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: PODS 2003, pp. 211–222 (2003)
14. Freedman, M., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudo-random functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005)
15. 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)
16. 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)
17. Huberman, B., Franklin, M., Hogg, T.: Enhancing privacy and trust in electronic communities. In: ACM Conference on Electronic Commerce, pp. 78–86 (1999)
18. 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)
19. Jarecki, S., Liu, X.: Fast Secure Computation of Set Intersection. Manuscript available from the authors (2009)
20. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
21. Ogata, W., Kurosawa, K.: Oblivious keyword search. Journal of Complexity 20(2-3), 356–371 (2004)
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)
23. 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)
24. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)

A: Cryptographic Assumptions

RSA assumption. Let $RSASetup(\tau)$ be an algorithm that outputs so-called RSA instances, i.e., pairs (N, e) where $N = pq$, e is a small prime that satisfies $gcd(e, \phi(N)) = 1$,

and p, q are randomly generated τ -bit primes. We say that the RSA problem is (τ, t) -hard on τ -bit RSA moduli, if for every algorithm \mathcal{A} that runs in time t we have:

$$\Pr[(N, e) \leftarrow \text{RSASetup}(\tau), \alpha \leftarrow \mathbb{Z}_N^* : \mathcal{A}(n, e, \alpha) = \beta \text{ s.t. } \beta^e = \alpha \pmod{N}] \leq \tau$$

One-More-RSA assumption. Informally, the One-More-RSA assumption [2] indicates that the RSA problem is hard even if the adversary is given access to an RSA oracle. Formally, let $(N, e, d) \leftarrow \text{KeyGen}(\tau)$ the RSA Key-Generation algorithm, and let $\alpha_j \leftarrow \mathbb{Z}_N^*$ (for $j = 1, \dots, ch$), we say that the One-More-RSA problem is (τ, t) -hard on τ -bit RSA moduli, if for every algorithm \mathcal{A} that runs in time t we have

$$\Pr \left[\{(\alpha_i, (\alpha_i)^d)\}_{i=1, \dots, v+1} \leftarrow \mathcal{A}^{(\cdot)^d \bmod N}(N, e, \tau, \alpha_1, \dots, \alpha_{ch}) \right] \leq \tau$$

where \mathcal{A} made at most v queries to the RSA oracle $(\cdot)^d \bmod N$.

B: APSI Protocol in Fig. 2

We now consider security and privacy properties of the protocol in Fig. 2.

Client Privacy. Recall that APSI is client-private if no information is leaked to the server about client's private inputs. It is easy to show that client's inputs are polynomially indistinguishable from a random distribution. This is because, in Step 1, the client selects all values uniformly and at random, i.e., $[R_c, \{R_{c:1}, \dots, R_{c:v}\}] \leftarrow \mathbb{Z}_n^*$. Thus, $X = PCH \cdot g^{R_c}$ and $\{y_i = PCH_i^* \cdot g^{R_{c:i}}\}$ form a random sequence. We defer the formal proof to the extended version of the paper.

Server privacy. To claim server privacy, we need to show that no efficient \mathcal{A} has a *non-negligible* advantage over $1/2$ against a challenger Ch in the following game. Our proof works in the random oracle model (ROM) under the RSA assumption.

1. Ch executes $(PK, SK) \leftarrow \text{Setup}(1^\tau)$ and gives PK to \mathcal{A} .
2. \mathcal{A} invokes *Authorize* on c_i of its choice and obtains the corresponding signature σ_i .
3. \mathcal{A} generates elements c_0^*, c_1^* different from every c_i mentioned above.
4. \mathcal{A} participates in the protocol as the client with messages X^* and y_0^*, y_1^* .
5. Ch picks one record pair by selecting a random bit b and executes the server's part of the interaction on public input PK and private input (c_b^*) with message (Z, y', t) as described in the protocol.
6. \mathcal{A} outputs b' and wins if $b = b'$.

Let $HQuery$ be an event that \mathcal{A} ever queried H' on input K^* , where K^* is defined (as the combination of message X^* sent by \mathcal{A} and message Z sent by Ch), as follows: $K^* = (X^*)^{eR_s} \cdot (h^*)^{-R_s} \bmod N$, where $Z = (g)^{eR_s}$ and $h^* = H(c^*)$. In other words, $HQuery$ is an event that \mathcal{A} computes (and invoked hash function H' on input of) the key-material K^* for the challenging protocol.

Unless $HQuery$ happens, \mathcal{A} 's view of interaction with Ch on bit $b = 0$ is indistinguishable from \mathcal{A} 's view of the interaction with Ch on bit $b = 1$.

Since the distribution of $Z = g^{eR_s}$ is independent from (c_b) , it reveals no information about which c_b is related in the protocol. Also, since y_0^*, y_1^* are not related to

$H(c_0)^d$ nor $H(c_1)^d$, $y' = (y_b)^{eR_s}$ reveals no information about which c_b is related in the protocol (y' is similar to an RSA encryption). Finally, assuming that H' is modeled as a random oracle, the distribution with $b = 0$ is indistinguishable from that with $b = 1$, unless \mathcal{A} computes $k^* = H'(K^*)$, in the random oracle model, by querying H' , i.e., HQuery happens.

If event HQuery happens with non-negligible probability, then \mathcal{A} can be used to violate the RSA assumption.

We construct a reduction algorithm called RCh using a modified challenger algorithm. Given the RSA challenge (N, e, α) , RCh simulates signatures on each c_i by assigning $H(c_i)$ as $\sigma_i^e \bmod N$ for some random value σ_i . This way, RCh can present the authorization on c_i as σ_i . RCh embeds α to each H query, by setting $H(c_i) = \alpha(a_i)^e$ for random $a_i \in \mathbb{Z}_N$. Note that, given $(H(c_i))^d$ for any c_i , the simulator can extract $\alpha^d = (H(c_i))^d/a_i$.

RCh responds to \mathcal{A} and computes $(H(c_i))^d$ (for some c_i) as follows: On \mathcal{A} 's input message X^*, y_0^*, y_1^* , RCh picks a random $m \leftarrow \mathbb{Z}_N$, computes $Z = g^{(1+em)}$, and sends Z and $y' = (y_b)^{1+em}$. We see that $g^{1+em} = g^{e(d+m)}$. On the HQuery event, RCh gets $K^* = (X^*)^{e(d+m)}(h^*)^{-(d+m)}$ from \mathcal{A} . Since RCh knows X^* , h^* , e , and m , it can compute $(h^*)^d$.

C: PSI Protocol in Fig. 4

We now consider privacy properties of the protocol in Fig. 4.

Client Privacy. As in Appendix B, we claim it is easy to show that client's inputs to the protocol are statistically close to random distribution. We defer formal proof to the extended version of the paper.

Server Privacy. We present a concise construction of an ideal (*adaptive*) world SIM_c from a honest-but-curious real-world client C^* , and show that the views of C^* in the real game with the real world server and in the interaction with SIM_c are indistinguishable, under the *One-More-RSA* assumption (presented in Appendix A) in the random oracle model.

First, SIM_c runs $(N, e, d) \leftarrow \text{RSA-Keygen}(\tau)$ and gives (N, e) to C^* . SIM_c models the hash function H and H' as random oracles. A query to H is recorded as $(q, h = H(q))$, a query to H' as $(k, h' = H'(k))$, where q and h' are random values. Finally, SIM_c creates two empty sets A, B . During interaction, SIM_c publishes the set $T = \{t_1, \dots, t_w\}$, where t_j is taken at random. Also, for every $y_i \in \{y_1, \dots, y_v\}$ received from C^* (recall that $y_i = H(c_i) \cdot (R_{c;i})^e$), SIM_c answers according to the protocol with $(y_i)^d$.

We now describe how SIM_c answers to queries to H' . On query k to H' , SIM_c checks whether it has recorded a value h s.t. $h = k^e$ (i.e., $h^d = k$).

If $\exists h$ s.t. $h = k^e$, SIM_c answers a random value h' and record (k, h') as mentioned above.

If $\exists h$ s.t. $h = k^e$, SIM_c can recover the q s.t. $h = H(q)$ and $h = k^e$. Then, it checks whether it has previously been queried on the value k .

If $\exists k$ s.t. k has already been queried, then SIM_c checks whether $q \in A$. If $q \notin A$, it means that C^* queried q to H (which returned h), and also made an independent query

k to H' s.t. $h = k^e$. In this case SIM_c aborts the protocol. However, it is easy to see that this happens with negligible probability. Instead, if $q \in A$, SIM_c returns the value h' previously stored for k .

If $\exists k$ s.t. k has already been queried, this means that SIM_c is learning one of C^* 's outputs. Hence, $A = A \cup \{q\}$. Then, SIM_c checks if $|A| > v$.

If $|A| \leq v$, then SIM_c checks if $q \in \mathcal{C} \cap \mathcal{S}$ by playing the role of the client with the real world server. If $q \in \mathcal{C} \cap \mathcal{S}$, SIM_c answers to the query on k with a value $t_j \in T \setminus B$, records the answer (k, t_j) and sets $B = B \cup \{t_j\}$. If $q \notin \mathcal{C} \cap \mathcal{S}$, SIM_c answers with a random value h' and records the answer.

If $|A| > v$, then we can construct a reduction Red breaking the *One-More-RSA* assumption.

The reduction Red can be constructed as follows. Red answers to C^* 's queries to H with RSA challenges $(\alpha_1, \dots, \alpha_{ch})$. During interaction, on C^* 's messages $y_i \in \{y_1, \dots, y_v\}$, Red answers $(y_i)^d$ by querying the RSA Oracle. Finally, if the case depicted above happens, it means that at the end of the protocol the set B will contain at least $(v + 1)$ elements, where v is the number of RSA challenges, thus breaking the *One-More-RSA assumption*. As a result, we have shown that the views of C^* in the real game with the real world server and in the interaction with SIM_c are indistinguishable.

We remark that the structure of the above proof has been inspired from the one based on the UPF secure under the One-More-Gap-DH assumption from [19], as well as the notion of *adaptiveness*. The adaptiveness allows the client to adaptively make queries, i.e., she does not need to specify all her inputs at once. In fact, we argue that the signing algorithm of unique signature is indeed an unpredictable function, hence its hash in the random oracle model results in a PRF.

D: Simplified Description of PSI in Fig. 3

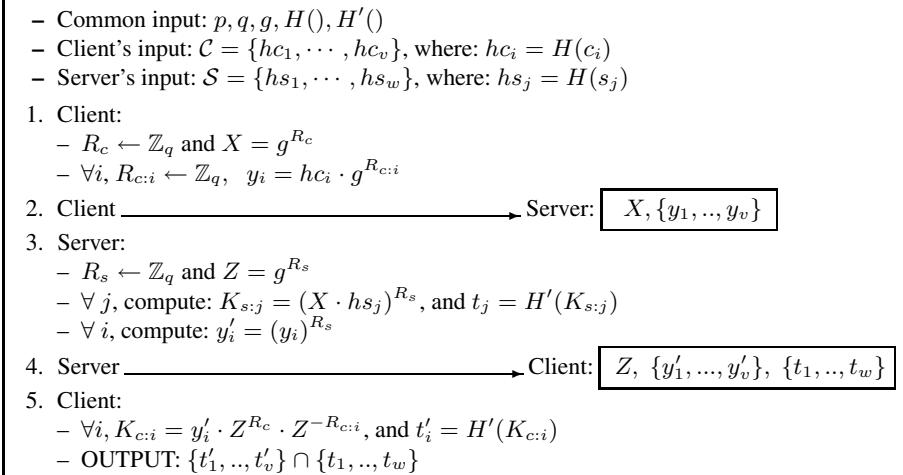


Fig. 5. PSI Protocol with linear complexity

Design and Implementation of a Key-Lifecycle Management System

Mathias Björkqvist, Christian Cachin, Robert Haas, Xiao-Yu Hu, Anil Kurmus,
René Pawlitzek, and Marko Vukolić

IBM Research - Zurich

Abstract. Key management is the Achilles' heel of cryptography. This work presents a novel *Key-Lifecycle Management System (KLMS)*, which addresses two issues that have not been addressed comprehensively so far.

First, KLMS introduces a *pattern-based* method to *simplify* and to *automate* the *deployment* task for keys and certificates, i.e., the task of associating them with endpoints that use them. Currently, the best practice is often a *manual* process, which does not scale and suffers from human error. Our approach eliminates these problems and specifically takes into account the lifecycle of keys and certificates. The result is a centralized, scalable system, addressing the current demand for automation of key management.

Second, KLMS provides a novel form of *strict access control* to keys and realizes the first cryptographically sound and secure access-control policy for a key-management interface. Strict access control takes into account the cryptographic semantics of certain key-management operations (such as key wrapping and key derivation) to prevent attacks through the interface, which plagued earlier key-management interfaces with less sophisticated access control.

Moreover, KLMS addresses the needs of a variety of different applications and endpoints, and includes an interface to the Key Management Interoperability Protocol (KMIP) that is currently under standardization.

1 Introduction

Cryptography is used to secure many information-technology systems, ranging from encrypting data on storage and establishing virtual private networks to protecting communication with mobile devices and using SSL certificates for e-commerce over the Internet. All uses of cryptography rely on the proper keys being present. Key management deals with the *lifecycle* of cryptographic keys, with operations for creating, importing, storing, reading, updating, exporting, and deleting them, and with distributing keys before they are used in cryptographic functions. An important aspect is to manage the attributes of keys that govern their usage and their relation to other keys.

Complications with key distribution are seen as the source of most operational problems with secure systems using cryptography. A key-management system must *provide* the appropriate keys and *deploy* them to *endpoints*, the entities that consume keys and use them for cryptographic functions. Managing a large number of keys manually does not scale, suffers from human error, and is prohibitively expensive. As a result, there is a great demand for *automated* key-management today, provided by centralized, scalable systems.

In an enterprise context, multiple users associated with many endpoints access the key-management system and perform operations on the objects that it maintains. These objects include *symmetric keys*, *public keys*, *private keys*, and *certificates*. A key-management system focuses on attribute handling rather than on cryptographic functions. But a comprehensive key-management system will also support a small set of cryptographic operations, including creating a key, issuing a certificate, to *derive* a new key (a deterministic operation that creates a symmetric key from an existing one), and to *wrap* or *unwrap* a key with another key (wrapping means to encrypt a target key with another key for export and transfer to another system).

In this paper, we describe the design and implementation of a prototype *Key-Lifecycle Management System (KLMS)*. It unifies key management for local and remote endpoints and handles many different types of cryptographic objects in a flexible way. KLMS addresses enterprise-level key management and covers many endpoints that require cryptographic keys, from heterogeneous applications, servers, and network devices to storage devices and media (e.g., tape cartridges). The system can provision keys to any application that can receive keys through the Java KeyStore (JKS) interface, as for example, file-based keystores in several formats, and it provides a prototype server for the Key Management Interoperability Protocol (KMIP), which is under standardization by OASIS [16].

KLMS introduces two novel features of a key-management system: first, the notion of *deployment patterns* to *automate* the administration and the deployment tasks for keys and certificates; and second, a *strict* implementation of *access control* to keys, which takes into account the cryptographic semantics of certain key-management operations and realizes the first cryptographically sound access-control policy for a key-management interface. We now briefly describe these two contributions.

Automated deployment. Often multiple related keys must be administered by the management system. To simplify this task, several keys can be grouped and *deployed* together to one or more *endpoints*. Such deployments can be structured according to certain *patterns*. In the example of a TLS-protected web server connected to the Internet, which is clustered for high availability, the same private key and certificate should be deployed to all nodes in the cluster. On the other hand, for a communication setup where multiple servers identify each other using their client-certificates through TLS-connections, every server should receive its own private key plus the public keys of all servers.

For supporting such scenarios, KLMS provides a novel pattern-based method for automated key and certificate deployment. A flexible *deployment manager (DM)* automates deployment and shields the system administrator from the lower-level key creation and distribution tasks. Once a suitable pattern and the supporting policies for a particular application are defined, KLMS automatically generates, distributes, and maintains as many keys or key pairs as necessary, and responds dynamically to changes of the topology, when new endpoints are added to the application.

KLMS also takes care of automatically managing the lifecycle of keys. It may create keys ahead of time and only maintain them internally, provisioned for a certain application. At the time of activation of a key, KLMS automatically deploys it to endpoints for the duration of its active life-time, and withdraws the key again from all endpoints when it expires. The key-lifecycling logic is tightly coupled with the deployment manager.

Strict access control. Every key-management system serves keys to users, the principals that invoke its operations. In the usual *basic* form of access control, the system decides about access to a key only by consulting an *access-control list (ACL)* associated with the key. But because the operations of the system allow users to create complex relationships between keys, through key derivation and key wrapping, basic access control may have security problems. For example, if there exists a key k_1 that some user is not allowed to read, but the user may wrap k_1 under another key k_2 and export the wrapped representation, the user may nevertheless obtain the bits of k_1 . Another example is a key that was derived from a parent key; when a user reads the parent key, the user implicitly also obtains the cryptographic material of the derived key.

In general, a cryptographic interface that manages keys and allows the creation of such dependencies among keys poses the problem that access to one key may give a user access to many other keys. This issue has been identified in the APIs of several cryptographic modules [2, 7, 9, 11] and may lead to serious security breaches when one does not fully understand all implications of an API.

Therefore, KLMS provides a *strict* mode of access control, in which decisions take the semantics of the key-management API into account, and implements a cryptographically sound access-control policy on all symmetric keys and private keys. The above issues with basic access control are eliminated with strict access control. Our strict access-control policy builds on the work of Cachin and Chandran [8], which describes a secure cryptographic token interface and introduces a cryptographically strong security policy. A strict access-control decision not only depends on the ACL of the corresponding key, but also takes into account the ACLs of related keys and the history of past operations executed on them. It prevents *any* unauthorized disclosure of a symmetric key or a private key.

Related Work. The need for building enterprise-scope key management systems has been widely recognized [5], and the US National Institute of Standards and Technology (NIST) has issued a general recommendation for key management [4]. Several commercial enterprise key-management systems are on the market, including HP StorageWorks Secure Key Manager, IBM Distributed Key Management System (DKMS), IBM Tivoli Key Lifecycle Manager, NetApp Lifetime Key Management, Sun StorageTek Crypto Key Management System, and Thales/nCipher keyAuthority.

In order to integrate these proprietary solutions, multiple efforts are currently underway to build and standardize key-management standards for open networks: the W3C XML Key Management Specification (XKMS), the IEEE P1619.3 Key Management Project, and the Key Management Interoperability Protocol (KMIP) standardization under the auspices of OASIS are some of the most prominent ones. Cover [10] gives an up-to-date summary of the current developments.

There is a rich literature on using *patterns* in software engineering and in systems management. One representative work that links this area with deployment on a system are the *deployment patterns for a service-oriented architecture* of Arnold et al. [3]; these patterns distribute applications to servers automatically by an algorithm that is available with the IBM Rational Software Architect product.

Access control and management of cryptographic keys are related in many ways. One prominent line of research, starting with Akl and Taylor [1], has concentrated on

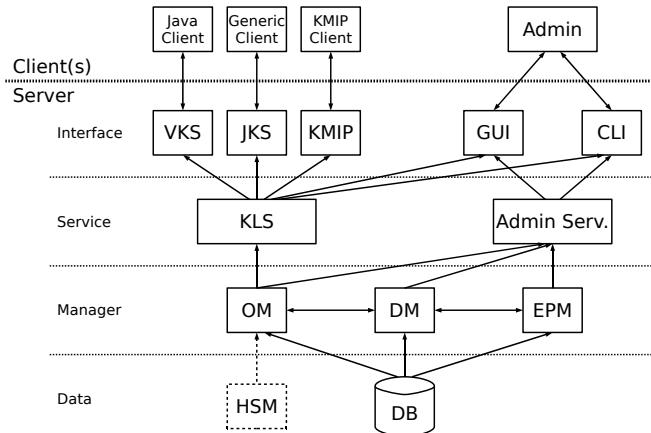


Fig. 1. Key-Lifecycle Management System architecture (see text)

representing a hierarchy (modeled as a partially ordered set) of users and their access privileges in a directed graph of keys, where a user inherits the privileges of all users below; this and subsequent work focuses on minimizing the number of keys stored by a user.

Because the key-management server provides the above-mentioned cryptographic functions, it represents a *cryptographic security API* accessible over a network. Security APIs stand at the boundary between untrusted code and trusted modules capable of maintaining internal state. Cryptographic security APIs are typically provided by cryptographic tokens [2], hardware-security modules (HSM) like IBM's 4764 cryptoprocessor that supports the IBM CCA interface [13, 15] and generic PKCS #11-compliant [17] modules, smartcards, or the Trusted Platform Module [18]. The study of cryptographic security APIs has so far been limited to programming APIs and to libraries; this paper extends their study to protocols for open networks.

Organization of the Paper. Section 2 describes the architecture of KLMS and its data model, and motivates some design choices. The main contributions, automated deployment and strict access control, are described in Sections 3 and 4, respectively. We conclude the paper in Section 5 by describing the implementation and the evaluation.

2 Model

System Architecture. The architecture of KLMS is shown in Figure 1. It is foreseen that the server is implemented in Java and runs on a web-application server. The KLMS server interacts with the clients (endpoints) through several types of interfaces, as described below. Administrators use a different interface than the clients to access the server. The server itself is structured in four layers; bottom-up, these are a data layer, a manager layer, a service layer, and an interface layer, as described next.

Data Layer. The data layer stores all information in a persistent *Database (DB)*. Internally, DB accesses a standard SQL database through the JDBC interface. All state information of KLMS is maintained by DB, so that KLMS does not lose any data because of a system crash. Meta-data about keys and certificates is stored in DB, as well as the cryptographic material itself. Some corporate security policies mandate that certain keys exist in cleartext only in a hardware-security module (HSM); the architecture supports this feature, by including a master key stored in an HSM and using it to encrypt all cryptographic material in DB.

Manager Layer. KLMS contains three components that provide low-level functionalities: an *Object Manager (OM)*, a *Deployment Manager (DM)*, and an *Endpoint Manager (EPM)*.

First, OM provides a simple interface to manipulate the *cryptographic objects* supported by KLMS. OM can add new objects, read, modify, search, and delete them in the DB, and maintains an in-memory object cache that is used to speed up read operations.

Second, DM takes care of administering *deployments* and *deployment bundles*. A deployment is an association between an object and an endpoint in the sense that KLMS provisions the object for use in cryptographic operations by the endpoint. The deployment policy realized by the DM dictates when and under which condition a deployed object finally becomes available to an endpoint through an interface; the deployment policy is described in Section 3. A *deployment bundle* is a set of deployments, which are grouped to support a given application.

Finally, EM controls the endpoints in the interface layer of the server, registering them in KLMS, potentially creating new file-backed JKS endpoints, and listening to protocol ports to which KMIP clients connect. EM unifies the different types of endpoints towards the rest of the server.

Service Layer. The service layer provides two modules: a *Key-Lifecycle Service (KLS)*, which is used by endpoints and by an administrator, and an *Admin Service*, which is only accessed by the administrator.

KLS represents the core of the server. It implements all operations related to keys and certificates that are available to endpoints and to users, drives automated deployment and lifecycle operations in conjunction with DM, and enforces access control. KLS can distinguish between different *users*, the principals that access it; every invocation of an operation occurs in the context of a *session*, which represents a user that has been securely authenticated by KLMS. The data model and the operations of KLS are described in the next section.

The Admin Service controls the allocation of endpoints and deployments through EPM and DM, respectively. Access to its operations also occurs in the context of a session, but is restricted to users with the corresponding permission. The Admin Service also allows archive and recovery operations for individual keys and for the whole database. Both modules, KLS and Admin Service, generate audit events.

Interface Layer. Three types of *endpoint interfaces* interact with the clients. The *Virtual Keystore (VKS)* interface emulates the provider of a Java KeyStore, for applications that are hosted by the same application server as KLMS. The client reads and writes keys

via VKS by issuing the “get” and “set” operations of the Java KeyStore interface. VKS is a *pull-style* synchronous interface, i.e., KLS can forward client calls to VKS directly to OM and DM.

The *Java Keystore (JKS)* interface accesses a named Java KeyStore as a client. A Java KeyStore is usually passive and its default implementation is a file, but depending on the installed Java Cryptography Extension (JCE) provider, many different entities may receive key material through the JKS interface (in particular, such clients need not be implemented in Java). JKS is a *push-style* asynchronous interface, because KLS calls the Java KeyStore interface and clients may retrieve keys from JKS at a later time.

A protocol interface provides an experimental implementation of the *Key Management Interoperability Protocol (KMIP)* draft standard [16]. KMIP is mostly a client-to-server protocol that offers rich functionality to manipulate keys and certificates. Many of its operations can be forwarded directly to KLS, but other operations are realized by an adapter module inside the KMIP interface. Ignoring the (optional) server-to-client operations in KMIP, the protocol interface is again *pull-style* and synchronous, similar to VKS. Clients connecting through KMIP need not be implemented in Java.

For the two keystore-based interfaces (JKS and VKS), EPM statically configures the user with which KLS is accessed. For the protocol-based interface (KMIP), it is possible to take the user from the client context. For the pull-style interfaces (VKS and KMIP), access control occurs when the client calls KLS. On the other hand, for the push-style JKS interface, access control must be enforced at the time when the deployment occurs.

Administrators access KLMS through a web-based *Graphical User Interface (GUI)* (built using the Hamlets framework [14]) or through a *Command-Line Interface (CLI)*; they both provide operations to deal with endpoints and to manage deployments. Note that clients who access the system through one of the endpoint interfaces cannot deploy keys or certificates in KLMS.

Data Model and Operations. KLMS manages *symmetric keys*, *public keys*, *private keys*, and *certificates*, which we summarily call *cryptographic objects* or simply *objects*. All key formats supported by the underlying Java platform and the JCE are available, including RSA, Diffie-Hellman, ElGamal, DSA, and EC-variations for public-key algorithms and symmetric keys of arbitrary length. Objects are composed of attributes and (possibly) cryptographic material. Attributes contain meta-data about the use of the cryptographic material, and they are the main concern of key management. Attributes may be read and sometimes also modified by clients in KLMS. KLMS also provides templates that simplify the handling of attributes for multiple objects, but we do not describe them in detail here, as they are not our primary concern.

KLMS currently supports close to 50 different object attributes. Rather than listing them all, we focus on the subset that is relevant for lifecycle operations, for automated deployment, and for access control (see Table 1). Every object has a unique *identifier*. A crucial attribute is the *state* of an object in its lifecycle. Objects in KLMS follow a lifecycle according to NIST [4, Section 7]. NIST distinguishes between using a key for *protect* purposes when applying cryptographic armor (through encryption, wrapping, signing, and so on), and *process* purposes when consuming previously protected data (through decryption, unwrapping, verification, and so on). A key may at certain times be used for one or both purposes, or for none at all. The lifecycle of a cryptographic object

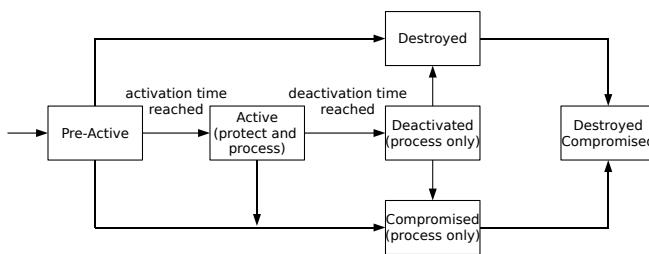
Table 1. Object attributes relevant for key lifecycle and access-control features

Key lifecycle		Access control	
<i>State</i>	<i>Deactivation time</i>	<i>Usage</i>	<i>Creator</i>
<i>Initialization time</i>	<i>Compromise time</i>	<i>Digest</i>	<i>Dependents</i>
<i>Activation time</i>	<i>Destroy time</i>	<i>Strict</i>	<i>Ancestors</i>
		<i>ACL</i>	<i>Readers</i>

progresses from a *Pre-Active* state, where it is not to be used for any cryptographic operation, through an *Active* state, where it may be used to protect and to process data, to a *Deactivated* state, where it may only be used to process data (see [4]). State transitions may be triggered directly by modifications to the lifecycle-relevant attributes, such as *state*, *activation time*, and *deactivation time*, or indirectly, as a side-effect of operations (e.g., when destroying an object). State transitions may cause actions by the automated deployment mechanism, as described in Section 3.

The operations of KLMS fall in two categories: those that manipulate objects, provided by KLS, and those that affect deployments, provided by Admin Service.

The most important operations on objects are: (1) *create*, which generates a new key or certificate and stores it, with attributes supplied by the client; (2) *store*, which stores a key or certificate and uses the cryptographic material supplied by the client in cleartext; (3) *import*, which stores a key and uses the cryptographic material supplied by the client in wrapped form (i.e., encrypted with another key); (4) *derive*, which creates a new symmetric key from an existing symmetric key; (5) *read*, which returns the key or certificate with a given identifier to the client, including attributes and cryptographic material in cleartext; (6) *export*, which returns the key with a given identifier to the client, including attributes and cryptographic material in wrapped form; (7) *read attributes*, which is the same as *read*, except that it omits the cryptographic material; (8) *set attributes*, which modifies the attributes of an object; (9) *search*, which locates all objects matching a given search condition and returns their identifiers; (10) *destroy*, which deletes the cryptographic material of an object, but leaves its attributes intact; (11) *delete*, which deletes the entire object; (12) *archive*, for writing some objects to off-line storage; and (13) *recover*, for reading objects back from off-line storage.

**Fig. 2.** Key states and transitions [4]. Transitions are triggered when an appropriate attribute is set or at the time specified by a time-related attribute.

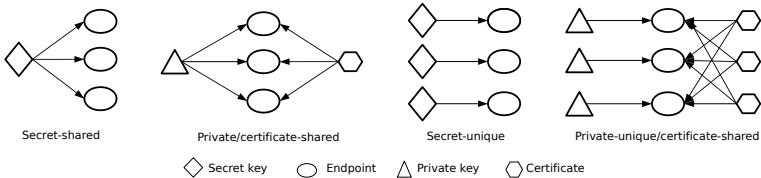


Fig. 3. Deployment patterns

The relevant operations of Admin Service on deployments are: (1) *specify*, which creates a deployment or a deployment bundle; (2) *activate*, which executes a deployment or a deployment bundle and distributes all objects to the specified endpoints; (3) *withdraw*, which reverses the effects of *activate* on a deployment or on a deployment bundle; and (4) *remove*, which removes a deployment or a deployment bundle from the system.

3 Automated Deployment

This section describes the pattern-based automated deployment in KLMS and the interaction between key-lifecycle management and key deployment. Recall that a deployment is an association between an object and an endpoint and that a deployment bundle is a set of deployments.

3.1 Deployment Patterns

A *deployment pattern* is a rule for generating deployment bundles with a defined structure. A deployment pattern is described in terms of an *object list*, an ordered set of keys and/or certificates to be deployed, and an *endpoint list*, an ordered set of endpoints, to which the objects are to be deployed. The pattern defines how the objects relate to the endpoints. Given an object list and an endpoint list, a deployment pattern yields a unique deployment bundle that complies with the pattern. Deployment patterns enable an administrator to focus on the requirements of an application, without having to worry about deploying individual keys and certificates.

We now describe four different deployment patterns, depicted in Figure 3.

Secret-shared: This pattern associates each element of the object list with every element of the endpoint list. For example, it is used to deploy a (set of) symmetric key(s) to multiple endpoints, such that they all share the same key(s).

When KLMS instantiates a *secret-shared* pattern, only the endpoint list is a mandatory input. The object list may be left out and a desired number n of symmetric keys can be given instead. In this case, KLMS generates n symmetric keys on the fly, taking their attributes from a template that is also included, and deploys all keys to each endpoint. (The generated keys are also stored in the DB.)

Private/certificate-shared: This pattern associates each private-key/public-key pair or private-key/certificate pair in the object list with every element of the endpoint list.

It is similar to the *secret-shared* pattern, but applies to asymmetric key pairs only (where public keys and certificates are used interchangeably).

A typical use-case for this pattern arises in a cluster of nodes that implement the same service, using replication and/or a fail-over strategy for increasing throughput and fault-tolerance. For example, when the cluster nodes serve web content over HTTPS for multiple domains, one private-key/certificate pair for TLS per domain must be available to every cluster node.

Secret-unique: This pattern associates the i -th key of the object list (containing only symmetric keys) with the i -th endpoint in the endpoint list, for $i = 1, \dots, n$. When KLMS instantiates a *secret-unique* pattern, the object list can be omitted and a template describing attributes for the keys can be given; KLMS then generates as many keys automatically as there are endpoints in the list, and deploys them. This pattern can be used for generating unique master keys for a range of secure devices, each of which is identified by a symmetric master key.

Private-unique/certificate-shared: This pattern takes an object list of n asymmetric key pairs (i.e., private key/public key or private key/certificate pairs) and a list of n endpoints as inputs, and associates the i -th private key with the i -th endpoint, for $i = 1, \dots, n$ and each public key/certificate with every endpoint. As with other patterns, when KLMS instantiates the pattern and the object set is omitted, then KLMS automatically generates the necessary key pairs from the attributes given in a template (such automatically generated certificates can only be self-signed).

This pattern addresses a typical infrastructure key-distribution model, where every entity is identified by a private key/public key pair, and every entity must know the public keys of all others. This could be a cluster of J2EE servers, whose communication is secured using those keys. Since the certificates are self-signed, all servers must have a copy of every other server's self-signed certificate in their keystore.

Note that the above list of four patterns is not exhaustive: one can define more patterns analogously, for example, a *private-unique/certificate-unique* pattern similar to *secret-unique*, by extending the generic association rules between object list and endpoint list above.

3.2 Administering Deployments

Recall that deployments are specified by an administrator using the Admin Service. Information on all objects deployed to endpoints is kept in a deployment table. Every deployment has a state that is either *OnHold* or *Active*.

When a deployment is in state *OnHold*, the deployment information is present in the deployment table, but the deployment should not yet or no longer take effect. Only during the time when a deployment is in *Active* state should the object be distributed to the endpoint and available to clients at the endpoint.

The administrator schedules the transition of a deployment from *OnHold* to *Active* state and vice versa by invoking the *activate* and *withdraw* operations of the Admin Service, respectively. After such a state change has been registered in the deployment table, it is the responsibility of a *distribution process* in DM to move or remove objects to or from the affected endpoints. Because its operations take time and may fail because

of network failures, the distribution process operates asynchronously in the background. This design shields the administrator from the different semantics of the endpoints. When DM distributes deployed objects to endpoints, it respects a *deployment policy* that affects its operation as described in Section 3.3.

In order to fully realize the power of automated deployment, the Admin Service allows dynamic modification of all pattern-based deployment bundles even after they have been created and activated. It is possible to add and to delete objects and endpoints to and from an existing deployment bundle. For example, when a deployment bundle d has been created by instantiating a pattern p , then an endpoint e can be added to it, and this will *specify* and *activate* a new deployment in d that affects e according to p . Likewise, when an endpoint e is deleted from d , this will *withdraw* and *remove* all deployments from d that contain e . Hence, the operator can manipulate deployments in a convenient way.

Note that a deployment of an object o to an endpoint e may be created through multiple ways, through individual deployments, deployment bundles, and pattern-based deployments. In this case, DM regards the deployment (o, e) to be in *Active* state whenever at least one of the sources is in *Active* state.

3.3 Deployment Policy

The policy followed by DM is called the *key-lifecycle deployment policy* and affects the behavior of the distribution process. The policy distinguishes *state-aware* endpoints that can interpret the *state* attribute of an object (such as those connecting through the KMIP interface) from *state-oblivious* endpoints that are not capable of expressing the notion of a lifecycle state (those connecting via VKS and JKS interfaces). The policy acts as follows. When DM distributes a deployment that associates an object o with an endpoint e , and when e is state-aware, then DM always distributes o to e ; on the other hand, when e is state-oblivious, then DM only distributes o to e if o is in *Active* state. This ensures that a state-oblivious endpoint never uses a key in *Pre-Active* state for a cryptographic operation, as this undermines the idea of managing the lifecycle of keys.

In order to support this policy, DM includes a *key lifecycle scheduler*, which executes the time-triggered state changes that can be specified by setting certain object attributes, like *activation time* or *deactivation time*. For example, a key with an *activation time* in the future can already be deployed to a state-oblivious endpoint; but the key is not distributed to the endpoint until the *activation time* is reached, when the key lifecycle scheduler executes that action.

Moreover, the policy involves access control when the deployment specifies the push-style endpoint, as explained in Section 2.

4 Strict Access Control

The difficulty of enforcing strict access control comes from relations between keys, which are introduced through *wrapping* and *key derivation*, such that a simple ACL no longer adequately represents a permission on a key. This section explains the basics of our strict access-control policy implemented in KLMS, which is based on the theoretical model by Cachin and Chandran [8]. For lack of space, we give only a summary here.

In short, strict access control guarantees that a user may only retrieve the information she is authorized to, i.e., that she cannot abuse the API to violate the access control policy. To achieve this, traditional access control (with ACLs independent among different objects) is not sufficient, since the interdependencies among different keys, arising from key wrapping and key derivation operations, may open security holes in cryptographic APIs [2, 7, 9, 11].

The KLMS server supports *key wrapping* (encrypting a target key under a different key, called wrapping key) through the *export* and *import* operations of KLS. To enforce strict access control, only key wrapping schemes are allowed that also provide strong authenticity, such as authenticated symmetric-key encryption in *CCM* [19] or *GCM* [12] padding mode. Through *key derivation*, a new symmetric key is generated from a parent key. Multiple keys derived from each other form a hierarchy, where knowledge of one key implies knowledge of all keys below in the hierarchy. Our system supports key derivation through the *derive* operation of KLS.

The KLS module authenticates users and distinguishes between different *users* who may execute its operations. The access-control policy is described by the attributes of the affected object. For some operations, which do not refer to any existing object (e.g., *create*), the access-control policy is governed by a list of permissions associated with every user.

Every object contains an *ACL* attribute, containing pairs of *users* and *permissions*. These permissions are: *Admin* (permits all operations), *Derive* (permits key derivation using the key as a parent key), *Destroy* (permits *destroy* and *delete* operations), *Export* (permits a key to be exported in a wrapped form), *Read* (permits reading the key in cleartext), *ReadAttributes* (permits gaining knowledge about key attributes), *Unwrap* (permits a key to be used for unwrapping in the *import* operation), and *Wrap* (permits a key to be used for wrapping in the *export* operation).

The system ensures for every ACL that the *Admin* permission always implies every other permission, that the *Export* and *Read* permissions imply permission to read the attributes, and that the *Read* permission implies the *Export* permission.

There is a boolean attribute *strict* for every symmetric key and every private key, which determines whether the object falls under the strict access-control policy and benefits from its guarantees.

For the implementation of strict access control, the server maintains three special attributes for every object. First, the *dependents* attribute contains the identifiers of those objects whose cleartext value can be computed from knowledge of the cleartext value of the object itself. Conversely, the *ancestors* attribute contains the set identifiers of those objects on which the given object depends, i.e., all objects whose *dependents* attribute contain the given object. Third, the *readers* attribute contains the set of users who have, or may potentially have, obtained the cleartext value of the given object. This may arise because they have either executed a *read* operation for the given object and obtained its cleartext value or because they have executed a *read* operation for another object that is contained in the *dependents* attribute of the other object.

Whenever the server has to authorize an operation executed by user u on an object o , it checks that the operation is allowed at least by the *basic access-control policy*, which is determined directly by the presence of a corresponding permission in the *ACL* attribute

of o . When $o.\text{strict} = \text{true}$, however, the server additionally verifies that the operation is permitted under the *strict access-control policy*. This takes the dependencies between cryptographic objects into account, and the server examines the *dependents*, *ancestors*, and *readers* attributes of o and possibly of further objects.

To give an example, consider the *read* operation. It returns the attributes of an object o and its cryptographic material in cleartext. For the operation to be permitted, user u must at least have the *Read* permission for o ; furthermore, if $o.\text{strict} = \text{true}$, then u must also have the *Read* permission for all objects k that are contained in $o.\text{dependents}$. This ensures that no key dependent on o is inadvertently leaked to a user that does not have sufficient privileges.

During all operations that change the dependencies between objects, the server must update the corresponding attributes, which adds some complexity to the implementation. For example, if an *export* operation creates an external representation of an object o wrapped with a symmetric key w using a secure key-wrapping method, then o becomes dependent on w ; the server adds o to the *dependents* attribute of w and adds the *identifier* of w to the *ancestors* attribute of o .

The detailed description of all operations can be found in the full version [6].

5 Implementation and Evaluation

Implementation. Our implementation of the KLMS server, together with the prototype support for KMIP, measures over 70k lines of Java code. Out of this, the core of the server (below the Interface layer) takes roughly 20k lines of Java code, with automated deployment taking slightly over 5.5k lines, and strict access control support around 1.5k lines. Currently, KLMS supports the four automated deployment patterns presented in Section 3.1, yet additional patterns can be added in a modular manner. The implementation of strict access control takes into account the possible size of the object attributes *dependents* and *readers*; these grow with the system’s age and may pose performance issues if implemented sub-optimally. To cope with this, our implementation uses two separate global tables in the DB layer for these two attributes. For the *readers* table, as for the representation of *ACL*, DB maintains the identity of a user determined from an LDAP directory server in the form of the string representation of the user’s LDAP Distinguished Name.

KLMS support for hardware-security modules (HSM) is foreseen by the architecture, but has not been implemented yet. Currently, the DB layer is based on a small-footprint Apache Derby database. The experimental integration of support for KMIP includes the portable portion of KMIP client/server code (around 28.5k lines of Java code) and the KMIP/KLMS adapter code (slightly over 4k lines). With this architecture, the support for KMIP can be easily transferred to a different key server core.

Evaluation. We have measured the performance difference between operations under the strict access-control policy and operations under the basic access-control policy, to determine the cost of the additional protection.

The benchmarking server is an IBM x345/6 system with two hyper-threaded Intel Xeon CPUs running at 3.06GHz and 2GB RAM. It runs our Java prototype of KLMS on an IBM J9 JVM (build 2.3, J2RE 1.5.0) with Apache Derby 10.3 as the database.

Table 2. The table shows the average times taken by four representative operations and the corresponding 95%-confidence intervals, in milliseconds

Operation	Policy	Average latency	95%-confidence interval
<i>Create</i>	basic	4.81	[4.44; 5.19]
	strict	5.10	[4.69; 5.51]
<i>Search</i>	basic	2.50	[2.39; 2.61]
	strict	2.61	[2.50; 2.72]
<i>Read</i>	basic	2.64	[2.60; 2.69]
	strict	3.75	[3.67; 3.83]
<i>Delete</i>	basic	9.38	[8.97; 9.79]
	strict	9.97	[9.49; 10.44]

The first experiment consists of running 100 *create* operations and measuring the elapsed time or latency. The operations are executed and measured at the service layer (in KLS). The created keys are then used to separately measure 100 *search*, 100 *read* and 100 *delete* operations. Because each operation takes only a few milliseconds, measurements are done over 100 operations. Each measurement was also performed 100 times (i.e. a total of 100×100 operations each, for strict and for basic access control). Table 2 summarizes the average time taken by one operation.

Our second experiment measures the scalability of the implementation of the strict access-control policy in KLMS, when a tree of dependencies grows deeper. The experiment starts by creating a new key and deriving from it a linear hierarchy of 10 keys, yielding 11 keys in total. We measure the time taken by the *derive* operation to derive one key, depending on the depth where this occurs. We chose maximal depth 10 because it is the maximum depth allowed by certain key-management products; in particular, the IBM CCA interface [15] allows maximal derivation depth 10. As the latencies are small (a few dozens of milliseconds), each data point is obtained by taking an average over 100 identical successive operations, and a total of 100 data points are collected per derivation level ($10 \times 100 \times 100$ derivations in total). Then the ACL of every key in the hierarchy is modified by adding the *read* permission for a fixed user, starting from the deepest key in the hierarchy up to the initial key. We measure the time taken by the *set attributes* operation. The measurements are done as previously for the *derive* operation.

Table 2 shows the overhead of the strict access-control policy for the four operations in the first experiment. One can see that most operations perform comparably except for *read*, which takes on average about 41% longer with strict access control. The reason is that, unlike with basic access control, the strict policy requires the *read* operation to modify the *Readers* attribute of an object. This explains the extra time taken by the operation.

Figure 4 shows the results of the second experiment. The measurements demonstrate that the strict policy scales well with the depth of the derivation tree for the *derive* operation, showing, roughly, a constant twofold overhead with respect to basic access control, with a slight increase for derivation trees deeper than six levels. For the *set attributes* operation, we observe a increasing overhead for modifying the ACL of those keys that have six or more dependent keys.

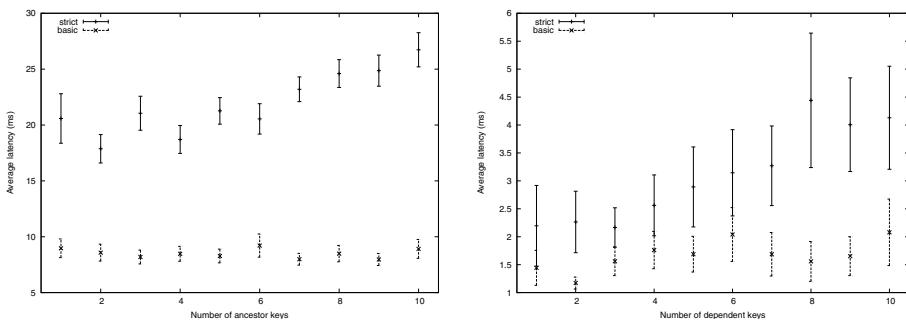


Fig. 4. The graphs show the average times for the *derive* operations (left) and the *set attributes* operations (right), for varying tree depth

References

- [1] Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems* 1(3), 239–248 (1983)
- [2] Anderson, R., Bond, M., Clulow, J., Skorobogatov, S.: Cryptographic processors — a survey. *Proceedings of the IEEE* 94(2), 357–369 (2006)
- [3] Arnold, W., Eilam, T., Kalantar, M.H., Konstantinou, A.V., Totok, A.: Pattern based SOA deployment. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 1–12. Springer, Heidelberg (2007)
- [4] Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management. NIST special publication 800-57, National Institute of Standards and Technology, NIST (2007)
- [5] BITS Security Working Group, Enterprise key management. Whitepaper, BITS Financial Services Roundtable (2008)
- [6] Björkqvist, M., Cachin, C., Haas, R., Hu, X.-Y., Kurmus, A., Pawlitzek, R., Vukolić, M.: Design and implementation of a key-lifecycle management system. In: Research Report RZ 3739, IBM Research (June 2009)
- [7] Bond, M.: Attacks on cryptoprocessor transaction sets. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 220–234. Springer, Heidelberg (2001)
- [8] Cachin, C., Chandran, N.: A secure cryptographic token interface. In: Proc. Computer Security Foundations Symposium (CSF-22). IEEE, Los Alamitos (2009)
- [9] Clulow, J.: On the security of PKCS#11. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 411–425. Springer, Heidelberg (2003)
- [10] Cover pages: Cryptographic key management (2009), <http://xml.coverpages.org/keyManagement.html>
- [11] Delaune, S., Kremer, S., Steel, G.: Formal analysis of PKCS#11. In: Proc. Computer Security Foundations Symposium (CSF-21). IEEE, Los Alamitos (2008)
- [12] Dworkin, M.: Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. In: NIST special publication 800-38D, National Institute of Standards and Technology, NIST (2003)
- [13] Dyer, J.G., Lindemann, M., Perez, R., Sailer, R., van Doorn, L., Smith, S.W., Weingart, S.: Building the IBM 4758 secure coprocessor. *IEEE Computer* 34(10), 57–66 (2001)
- [14] Hamlets, <http://hamlets.sourceforge.net>

- [15] International Business Machines Corp., CCA Basic Services Reference and Guide for the IBM 4758 PCI and IBM 4764 PCI-X Cryptographic Coprocessors (2008)
- [16] OASIS Key Management Interoperability Protocol Technical Committee, Key Management Interoperability Protocol (2009)
- [17] RSA Laboratories, PKCS #11 v2.20: Cryptographic Token Interface Standard (2004),
<http://www.rsa.com/rsalabs/>
- [18] Trusted Computing Group, “Trusted platform module specifications (2008),
<http://www.trustedcomputinggroup.org>
- [19] Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). RFC 3610 (2003)

Measuring the Perpetrators and Funders of Typosquatting

Tyler Moore¹ and Benjamin Edelman²

¹ Harvard School of Engineering and Applied Sciences

tmoore@seas.harvard.edu

² Harvard Business School

bedelman@hbs.edu

Abstract. We describe a method for identifying “typosquatting”, the intentional registration of misspellings of popular website addresses. We estimate that at least 938 000 typosquatting domains target the top 3 264 .com sites, and we crawl more than 285 000 of these domains to analyze their revenue sources. We find that 80% are supported by pay-per-click ads, often advertising the correctly spelled domain and its competitors. Another 20% include static redirection to other sites. We present an automated technique that uncovered 75 otherwise legitimate websites which benefited from direct links from thousands of misspellings of *competing* websites. Using regression analysis, we find that websites in categories with higher pay-per-click ad prices face more typosquatting registrations, indicating that ad platforms such as Google AdWords exacerbate typosquatting. However, our investigations also confirm the feasibility of significantly reducing typosquatting. We find that typosquatting is highly concentrated: Of typo domains showing Google ads, 63% use one of five advertising IDs, and some large name servers host typosquatting domains as much as four times as often as the web as a whole.

1 Introduction

At the dawn of commercial Internet activity, aggressive website registrants discovered that they could profit by registering domain names matching others’ company names, product names, and trademarks – “cybersquatting,” as the practice came to be known. Initially, cybersquatting promoted competitors, as in Princeton Review’s 1994 registration of kaplan.com to divert Internet traffic intended for a competing test preparation service. Once domain names started requiring annual renewals, squatters raced to grab domain names when the prior registrant failed to renew [3]. By 1999, squatters began “typosquatting” – intentionally registering misspellings of popular websites in anticipation that users mistype those domains and reach squatters’ sites [5].

Cybersquatters have employed several strategies to profit from their registrations. After grabbing particularly valuable domains, some squatters sought small ransoms from the organizations that most wanted those domains. In one notorious case [3], a squatter redirected thousands of expired domains to adult

websites, making it all the less palatable to leave the domains with the squatter, and all the more tempting to pay to get the domains back. Separately, following in Princeton Review's footsteps, other squatters redirected squatting domains to direct competitors of the sites users expected. Through such redirects, a squatter could either profit directly (if the squatter also operated the destination site) or indirectly (through marketing commissions paid by the destination site). Finally, a growing share of squatters found profits through advertising – typically, showing pay-per-click ads through the web's top ad networks.

As the Internet matured, cybersquatting domain registrations came to be viewed as both disruptive and improper. In 1999, countermeasures to squatting were introduced, including an arbitration procedure (the UDRP) and a new federal law (the ACPA), both discussed in Section 7.1. The subsequent decade featured more than 45 000 UDRP disputes and more than \$40 million of ACPA damage awards. Yet our paper shows that cybersquatting and especially typosquatting remain widespread.

In this article, we explore modern typosquatting. Our methodological contribution is the development of a software system that effectively identifies typosquatting using telltale patterns in domain registrations and configurations. Our substantive contribution is a characterization of the typosquatting problem, including estimating its size, assessing who is responsible, and identifying factors that put some brands, marks, and domains at heightened risk of typosquatting.¹

2 Structure and Strategy of the Domaining Business

Most large domain registrants present themselves as “domain parkers” or *domainer*s. Figure 1 outlines the relationship between domainers, advertisers and Internet authorities. Domainers submit registration requests to *registrars* (e.g., GoDaddy), which provide domain names after confirming availability with *registries* (e.g., VeriSign), which in turn are authorized by the Internet Corporation for Assigned Names and Numbers (ICANN) which coordinates certain Internet identifiers under contract with the United States Department of Commerce. At each step, money changes hands: domainers pay registrars which pay registries, and both registrars and registries pay fees to ICANN.

As Figure 1 depicts, the domaining business can feature numerous variations. For example, a large domainer may elect to become a registrar – simplifying relationships and eliminating an intermediary. In 2006 litigation, Neiman Marcus alleged exactly that in a 155-page complaint claiming that Dotster (a large domain registrar) hoarded typosquatting domains for its own benefit, failed to disclose domain ownership via Whois records, and improperly extracted fees from companies wanting domains Dotster had collected [9]. There is also variation in the structure of domainers' relationships with *ad platforms* (e.g., Google AdSense): Large domainers typically work directly with ad platforms, while smaller domainers typically work through traffic aggregators (e.g., information.com) that combine traffic from many domainers.

¹ <http://www.benedelman.org/typosquatting/> details data collected for the paper.

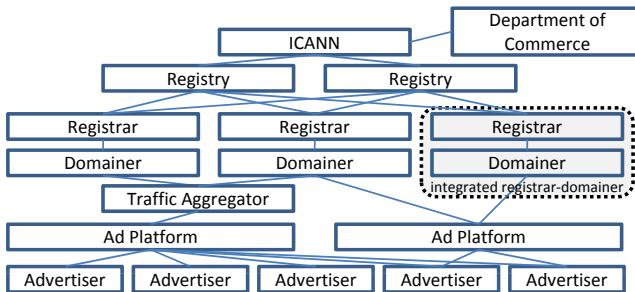


Fig. 1. Diagram of relationships

The domain parking business is premised on users arriving at parking sites. But *why* do users go to parking sites? Some users seem to type in domain names randomly, rather than using search engines to find the materials that meet their requirements – requesting a site like *discounthoteldeals.com* when seeking “discount hotel deals.” Such users might end up at domain parking sites matching the generic keywords that embody their requests. But there is another way for users to arrive at parked domains: misspelling the address of a more popular site. This practice, typosquatting, is the focus of our paper.

3 Measuring Typosquatting

3.1 Identifying Typosquatting Domains

The first step in studying typosquatting is to identify which domains are similar enough to be deemed typos. We start by gathering a list of *popular domains* that could tempt a squatter to register many *typo domains*. For this paper, we decided to study the 3 264 .com domains at least 5 characters long appearing in the most popular 6 000 domains according to Alexa’s June 29, 2009 ranking. We focus on the most popular sites because, all else equal, popular sites are more likely to be targeted for typosquatting: The more users seek to visit a domain, the more users are likely to mistype the domain’s address. (Section 5 empirically examines the factors affecting typosquatting prevalence.) We excluded very short domains (4 characters or less) because even one-letter variations may reflect intentional requests for other short domains, rather than typos of a popular site. Finally, we only consider .com domains due to .com’s ubiquity and because the zone file listing all .com domains is publicly available.

Next, we generated a list of plausible misspellings of the 3 264 popular domains. To identify plausible misspellings, we rely on the Damerau-Levenshtein distance [2,6]: the minimum number of insertions, deletions, substitutions or transpositions required to transform one string into another. For example, `facebook1k`, `facebook`, `facebook1k`, and `facebook0ko` each have a Damerau-Levenshtein distance of 1 from `facebook`. Damerau found that 80% of spelling errors are caused by one such operation. We also created a list of typos with `www` and `.com` appended to the start

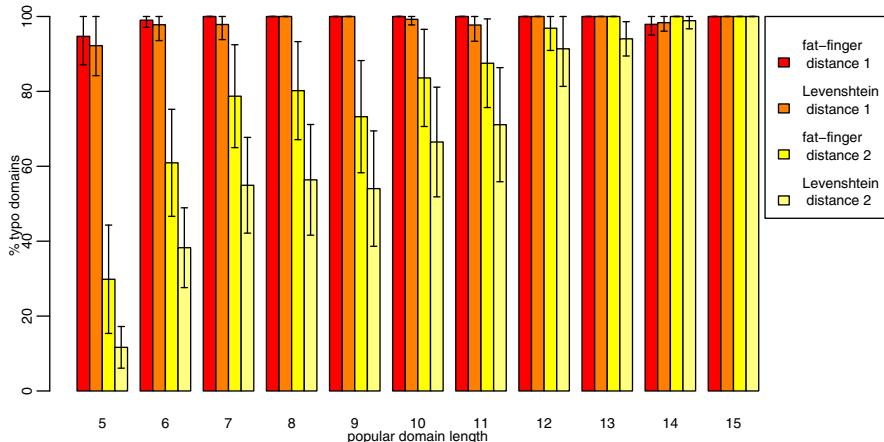


Fig. 2. Typosquatting classification accuracy

and end of the strings, respectively. These appendages help recognize frequent mistakes arising out of omitting a ‘.’ when typing a URL.

We also devised a new measure of string distance useful for keyboard misspellings, called *fat-finger distance*: the minimum number of insertions, deletions, substitutions or transpositions *using letters adjacent on a QWERTY keyboard* to transform one string into another. For example, `facebojk` has a fat-finger distance of one from `facebook`, since ‘j’ is next to ‘k’ on a standard keyboard.

To identify typosquatting domains, we enumerated all strings with a Damerau-Levenshtein (and fat-finger) distance of up to 2 from each of the 3 264 popular domains. This captured all plausible one- and two-letter typos of popular domains. We next intersected this set with the nearly 81 million registered .com domains (according to the .com zone file). This process yielded 1 910 738 registered .com domains as *candidate* typo domains of the 3 264 popular domains.

We manually checked a sample of 2 195 domains randomly selected from the list of 1.9 million candidates. To form this sample, we selected candidate domains targeting popular domains of length 5–15, allowing variations of Levenshtein distances of 1 and 2 and with fat-finger distances of 1 and 2. Figure 2 plots the results with 95% confidence intervals. (In particular, we have 95% confidence that the true number of .com domains up to a Levenshtein-Damerau distance of two from the popular domain lie between the numbers posted in the table. Of course, our analysis omits typo domains of distance ≥ 3 and also typos in other top-level domains such as .cm.) No matter the length of the popular domain, typo domains within Levenshtein or fat-finger distance 1 of popular domains were overwhelmingly confirmed as typos. When we consider typos of distance 2 from popular domains, false positives become more frequent. However, domains within fat-finger distance 2 of popular domains are more likely to be typos than domains within only Levenshtien distance of 2. Furthermore, for increasingly lengthy popular domains, it is increasingly likely that domains with a fat-finger or Levenshtein distance of 2 are in fact typo domains.

Table 1. Selected domains highly targeted by typosquatting

popular site	candidate typo domains	point estimate typo domains	95% confidence interval
google.com	5 731	2 537	(1 728, 3 252)
youtube.com	3 616	2 069	(1 589, 2 534)
myspace.com	3 482	1 960	(1 457, 2 440)
freecreditreport.com	1 904	1 904	(1 904, 1 904)
hotels.com	4 465	1 865	(1 207, 2 442)
total for 3 264 domains	1 910 738	937 918	(710 872, 1 236 924)

With false positive estimates from our manual checks, we estimated the number of .com typo domains targeting the popular sites we identified. To do so, we added up the number of candidate typos matching each popular site; then we adjusted each candidate's weight based on our confidence in its accuracy in light of the typographical distance between the typo domain and the popular domain. By this methodology, we estimate that approximately 938 000 typo domains target variations of the 3 264 popular domains we studied. On average, each popular site is targeted by 281 typo domains, but some sites attract more typosquatting than others. Table 1 lists the sites that are most targeted by typosquatting. Topping the list is google.com, for which we found an estimated 2 537 typo domains. That said, as we show in Section 4.1, Google also supports typo domains by providing both technical assistance and advertisement payments.

3.2 Crawling Typosquatting Websites

We cannot easily visit all 938 000 typo domains without also visiting many sites that are not typos. Because we wish to learn more about only typosquatting websites, we decided to crawl only a subset of the typo domains where we know the vast majority are in fact typos. To that end, we developed a crawler to visit the 284 914 typo domains where the Damerau-Levenshtein distance between typo and popular domains is at most one for popular domains between five and nine characters in length, and a distance of up to two for popular domains at least ten characters long. Consequently, the totals discussed in the subsequent sections should be interpreted as a sample of the larger typosquatting population.

The crawler explored each typo site and its links to determine how a given site is being used, and the crawler recorded all HTML and headers that it received. The results of our crawl are presented in Sections 4.1 and 4.2.

We designed the crawler to avoid burdening websites or advertisers. The crawler follows three randomly-selected links on each page, up to (at most) a depth of three links. Since indiscriminately following pay-per-click links and redirects could yield unwarranted cost to advertisers and unearned revenues to squatters, the crawler only invokes a link after comparing that link to a list of known ad servers. If a link references a known advertising domain, the site is marked as containing advertisements, the link is recorded, and the crawler

Table 2. How typo domains are used

classification	typo domains	%
pay-per-click ads	74 024	79.4
Google	53 364	57.2
Yahoo!/Overture	19 145	20.5
Ask.com	555	0.6
Miva	541	0.6
Enhance	297	0.3
domain redirection/link	19 227	20.6
self-registration	4 133	4.4
affiliate marketing	10 215	11.0
redirect or link to competing site	4 879	5.2
blocked	124 211	—
unclassified	70 729	—

proceeds no further at that site. The same logic is used whenever the crawler encounters a HTTP redirect. The crawler assures that at most one of its threads visits a single server (at a single IP address) at a time.

Although our crawler could not retrieve and classify all the typo domains it identified, we believe the classified domains provide appropriate insight into the usage of the other domains. For one, 131 637 of the 194 940 blocked or unclassified domains share the same IP address and name server with domains where we confirmed the presence of PPC ads.

Our crawler found that 124 211 domains blocked its efforts at inspection. Some servers host tens of thousands of typo domains. Despite our crawler's best efforts, such servers may notice after a machine from a single IP address tries to visit several links on thousands of domains. In manual tests, we confirmed that a few of these domains were truly unreachable. But often these "blocked" domains reside on IP addresses that had previously responded as expected, and often these "blocked" domains loaded as expected when tested from another IP address. We therefore conclude that at least some typosquatting hosts recognized our crawler's examinations and took steps to prevent our analysis.

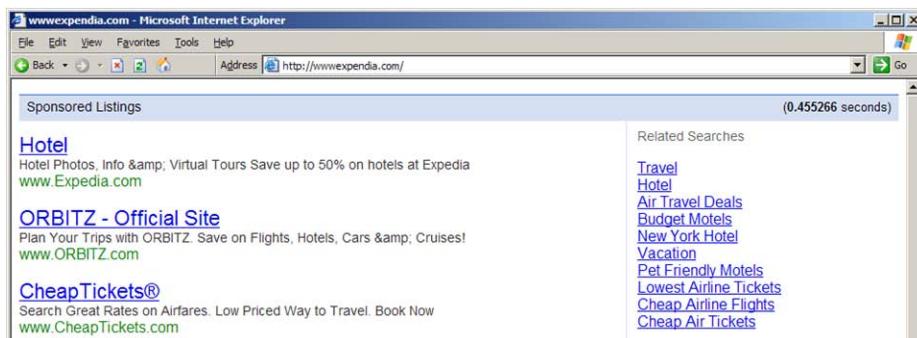
In addition to sites that affirmatively blocked our analysis, our crawler was unable to classify a further 70 729 domains. Many of these domains included JavaScript links, which our crawler could not reliably follow.

4 How Typosquatting Domains Are Used

From crawling typosquatting pages, we confirmed two main uses for traffic diverted to typo domains: placing pay-per-click ads and redirecting to other (often competing) domains. We discuss our findings for each strategy in turn.

4.1 Squatter Strategy 1: Pay-Per-Click Ads

By far the most common use for typo domains is displaying pay-per-click advertisements. Of the typo domains our crawler could classify, Table 2 reports



self-advertising rate	popular sites	examples
≥75%	18	papajohns (90%), saksfifthavenue (88%)
50%≤...<75%	58	expedia (50%), t-mobile (70%)
25%≤...<50%	106	wellsfargo (43%), businessweek (48%)
<25%	81	findlaw (21%), tigerdirect (22%)
overall: 36%	263	

Fig. 3. `wwwexpendia.com` shows ads for `expedia.com` and competitors (top); self-advertisement prevalence for 263 popular sites buying ads (bottom)

that 80% – over 74 000 – included pay-per-click ads. Most of these websites – at least 53 364 – partnered with Google to sell ad space to advertisers, select which ads to display, track clicks, and collect payments, among other functions. Google’s prevalence in part reflects Google’s large market share in pay-per-click advertising, and Google further benefits from its development of an advertising service dedicated to placing ads onto parked domains.² Next-largest after Google is Yahoo; we found Yahoo ads on at least 19 145 typo domains. We detected three additional PPC ad providers also being used, but with dramatically lower prevalence, as detailed in Table 2.

Figure 3 (top) shows PPC ads on `wwwexpendia.com`. The top advertisement promotes `expedia.com`, the same domain misspelled in the user’s request. The Expedia ad appears because Expedia pays Google to advertise on websites with “relevant” content, and Google’s algorithm select `wwwexpendia.com` as a suitable place for those ads. Consequently, Expedia pays Google whenever a user misspells Expedia and clicks the sponsored link to Expedia [4]. Meanwhile, immediately below Expedia are advertisements for competitors Orbitz and CheapTickets. Had Expedia chosen not to pay Google to place ads on parked domains, Google would have shown links only to competing sites.

We found self-advertisements on typo domains targeting 263 popular sites that bought ads (Figure 3 (bottom)). Sometimes, nearly all typo domains included ads to the popular site (e.g., 90% for typos of `papajohns.com`). For others, self-advertising occurred less often (e.g., 22% for `tigerdirect.com`). Overall, we saw ads corresponding to the popular site on 36% of typo domains.

² See Google AdSense for Domains, <http://www.google.com/domainpark>.

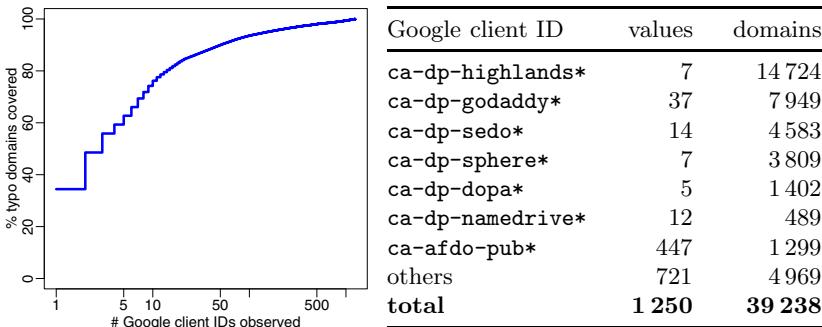


Fig. 4. Advertising client IDs matching typo domains

While stopping so many typo domains may seem like a Sisyphean task, we found considerable concentration upon closer inspection of PPC ad links. An ad provider needs to know who to pay for a given advertisement placement, so a partner ID is passed as a parameter in an ad's click URL. For example, in the link <http://domains.googlesyndication.com/apps/domainpark/results.cgi?client=ca-dp-mborin&...> on cartoonntewrok.com, the `client` parameter is set to `ca-dp-mborin` – indicating Google will pay the corresponding partner if a user clicks that ad link. On other domains, these parameters appear in HTML entity encoding, in redirects, and/or in JavaScript variables.

We found partner ID codes for 74% of typo domains showing Google ads. While 1 250 different codes were found, some turned up disproportionately often. Figure 4 (left) plots the cumulative distribution of typo domains by partner ID (note the logarithmic x-axis). The top 5 partner IDs cover 63% of the Google typo domains we explored, and the top 10 cover 76%. The most frequent partner ID was `ca-dp-highlands19_3ph_xml`, appearing in ad links on 13 542 typo domains. The table in Figure 4 lists specific partner ID we observed particularly frequently on typo domains. Large domainers and traffic aggregators often have recognizable Google IDs, such as `ca-dp-godaddy` (GoDaddy), `ca-dp-namedrive` (NameDrive), `ca-dp-sedo` (Sedo) and `ca-dp-namesphere` (NameSphere).

Unfortunately, we could not identify partners from Yahoo ads, since the ads use a single parameter `xargs` presenting a lengthy obfuscated string apparently combining ad destination, partner, and more. While we cannot determine which Yahoo partner receives credit for a given placement, we can still demonstrate high concentration among Yahoo partners. For example, one typosquatter passed Yahoo PPC links as a parameter within redirect URLs with the distinctive parameter `provider` set to 1200. This same pattern is found on 10 446 typo domains, nearly all using the same name server and IP address.

4.2 Squatter Strategy 2: Redirection and Linked Domains

Rather than showing pay-per-click ads, other typo domains redirect or link to predetermined destination domains. We saw three practices in this vein:

(i) self-registrations/defensive registrations, (ii) affiliate marketing and (iii) redirect or link to competing site.

Self-registrations / defensive registrations. In some instances, a company will “self-register” misspellings of its key domains. Often, requests for these typo domains redirect a user to the company’s main site, where the user likely intended to go. Through self-registrations, a company can avoid unwarranted marketing expense, such as paying for ads on typo sites, as shown in Figure 3. Self-registrations also help users reach their intended destinations without extra clicks or delays. We found 4 133 typo domains that match this profile, in that they share the same name servers as the popular sites of which they are variants.

Affiliate marketing. Through merchants’ affiliate marketing programs, some typosquatters send users to the sites users intended to visit – but charge the merchants a fee for providing these referrals.

In affiliate marketing, advertisers pay for referrals on a performance basis: Send a user to Dell and Dell will pay a commission of 2% or more. Because affiliate merchants generally only pay when a user makes a purchase, many merchants fail to supervise their affiliates’ specific promotional methods. Few affiliate merchants affirmatively allow typosquatting, and most disallow it when it comes to their attention. But to date, few merchants have uncovered affiliates engaged in typosquatting. (In a rare exception, Lands’ End sued several squatters who registered typosquatting domains and redirected resulting traffic to Lands’ End affiliate links [7].)

We saw 10 215 typosquatting domains that linked or redirected to the corresponding popular site, where the name server used by the squatting site and the popular site differed. We checked all redirections from the misspelled domain name, looking for redirections to the popular site via an affiliate marketing network. We confirmed 2,697 domains redirected to affiliate marketing networks, including 905 typo domains promoting Commission Junction merchants, 652 promoting LinkShare merchants, and 290 promoting Performics (Google Affiliate Network) merchants. Another 4 629 redirected to the legitimate domain, either as a result of defensive registration or for directly managing affiliates (e.g., `bookihng.com` redirects to `booking.com/?aid=311266;label=11-booking-promo`).

Redirects or links to competing site. When users attempt to visit a popular site, some typosquatters instead forward the users to a competing site – often in the same industry, but a notch less popular. For example, `pict.com` is a relatively little-known document sharing site (Alexa rank 8 581 as of Aug 27, 2009). But `pict.com` is redirected to by typos of 128 competing, more popular sharing sites – 24 typos of `depositfiles.com` (Alexa rank 167), 22 typos of `picoodle.com` (Alexa rank 5 040), 18 typos of `sharebee.com` (Alexa rank 1 673), and more. These redirects take users directly to `pict.com` with no link codes or partner IDs of any kind – suggesting that `pict.com` itself registered these domains and that, in any event, `pict.com` is probably not paying partners for this traffic. Similarly, we found 156 typo domains that are variations of `yellowpages.com`, which all redirect to the website `yellowpagestheworld.com`.

Table 3. Example domains linked to by typo variations of competing domains

yellowpagesoftheworld.com: 158 typo domains
yellowpages.com: yellopasges, yeollwpages, yelkowpages & 153 more
whitepages.com: whigtepages & whitepagecom
bet365.com: 367 typo domains
sportsbook.com: saportsbook, sxportsbook, sportszbook & 325 more
betclic.com: betclico, betclicm, betclicj & 7 more
fulltiltpoker.com: fulltilt6poker, fuylltiltpoker, fulltiltpoke4r & 5 more
pict.com: 128 typo domains
depositfiles.com: dopsktfiles, depositfimes, depositciles & 21 more
picoodle.com: picoodke, picoodme, piciodle & 19 more
sharebee.com: shaerbee, shafebee, shatebee & 15 more
movietheatertickets.biz: 85 typo domains
movietickets.com: movietikits, mpvietickets, mvietickets & 19 more
rottentomatoes.com: rottentomaos, rottentmoatoes, rotentomatoe & 10 more
fandango.com: fandsango, fandnango, faneango & 9 more
total: 75 beneficiary domains on 4 879 typo domains targeting 668 competing popular sites

We developed a simple heuristic to identify typo domains linking to competing domains. First, we group typo domains that all link to the same *beneficiary domain* (e.g., `pict.com`, a domain benefiting from this group of typo domains). Next, we consider only those beneficiary domains that are linked by typo domains targeting a *small number* of popular sites. By focusing on beneficiary domains receiving traffic from typos on a small number of popular sites, we identify beneficiary domains that are targeting typosquatting on specific popular sites (typically, in the same sector), rather than aggregating typo traffic more generally. Through testing, we adjusted the parameters, and we elected to focus on beneficiary domains linked by at least 75 typo domains that target no more than 40 popular sites. Using this criteria, we identified 75 beneficiary domains that are linked from 4 879 distinct typo domains, which collectively target 668 competing popular sites.

Table 3 lists selected beneficiary domains identified using our heuristic. Notably, every beneficiary domain is linked by typos in the same category: typos of popular casino websites link to `bet365.com`, popular movie sites link to `movietheatertickets.biz`, and so on. This trend is consistent for all beneficiary domains. It is not always clear whether the beneficiary domain directly registered and configured the typo domains; affiliate marketing and similar relationships can motivate partners to register typo domains.

5 Do Pay-Per-Click Ads Promote Typosquatting?

Table 1 reveals that some popular sites are targeted by typosquatting far more than others. Why? We initially hypothesized that typosquatting disproportionately afflicts domains that are difficult to spell. To check, we regressed number

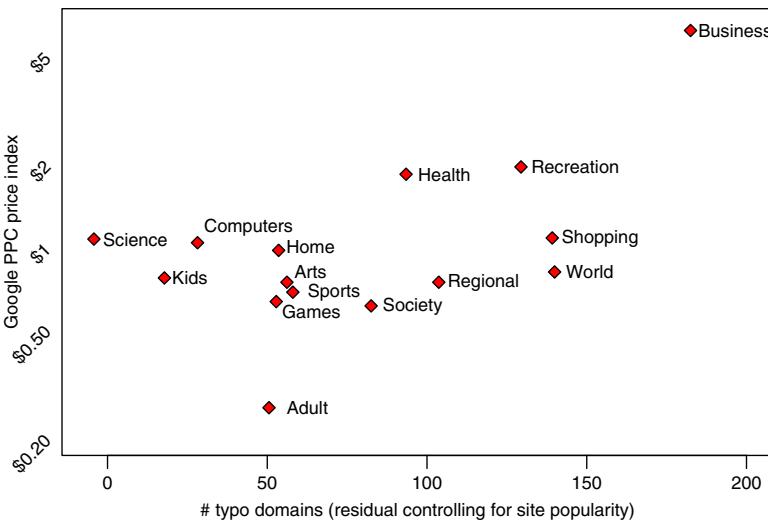


Fig. 5. Scatter plot comparing typosquatting incidence to amount paid out by pay-per-click ads

of typos on popular sites spelling difficulty (with controls for various measures of popularity of the popular site). We found no effect of spelling difficulty as measured by number of double letters or presence of adjacent i/e tuples – perhaps reflecting that these top popular sites have limited variation in spelling difficulty (as measured by these proxies).

However, we do find significant differences across website categories. To assign popular sites to categories, we used Alexa's listings of the top 500 websites for 15 different categories (e.g., Kids and Teens, Business, News). 1 075 of the popular sites we studied also appeared in one or more of Alexa's top 500 categories. In a regression controlling for each popular site's popularity and number of category listings, we included a fixed effect for each category, and we noted the coefficient associated with each category variable. These coefficients form the x coordinates in Figure 5. For example, the average popular site Alexa places in "Shopping" is targeted by 143 more typo domains than the average popular site Alexa places in "Science."

Because most typo domains are funded by pay-per-click ads, we examined patterns in PPC pricing across Alexa categories. For each popular site Alexa listed in each category, we extracted META keywords, and we identified the ten most frequent keywords in each category. Using the Google Traffic Estimator, we obtained minimum and maximum PPC price estimates for each frequent keyword. We formed a *Google PPC price index* for each category, given by the average of 1) the median of the minimum PPC prices for keywords in that category, and 2) the median of the maximum PPC prices in that category.

Combining Alexa's categorizations with our PPC price index yields the result shown in Figure 5. Notice the positive association: In categories with higher PPC prices, parkers registered more typosquatting domains. We interpret this

relationship as evidence that high PPC prices spur typosquatting registrations in the corresponding categories.

6 Estimating Visitors and Advertising Costs

It is difficult to know precisely how many people visit typo sites. However, even a rough approximation helps confirm the impact on consumers and advertisers. Using the estimates in preceding sections plus public site traffic data, in this section we form an estimate of the number of visitors reaching typo sites, as well as the fees advertisers pay to Google, the ad platform which most frequently funds typo sites.

For site traffic data, we look to Alexa, which estimates the popularity of selected websites. For a sufficiently popular site, even the site's typosquatting misspellings receive enough traffic for Alexa to estimate their popularity. However, less popular sites receive too little traffic at their typosquatting variants for Alexa to report a rank for those typo sites. We therefore begin our analysis by considering Alexa's estimates of the number of daily visitors browsing close typos of the 50 most popular .com websites. On average, visitors to a site's typo domains total 0.7% of visits to the genuine site. Extrapolating with this percentage to consider all 3264 popular sites studied in this paper, we estimate that typo domains collectively receive at least 22.1 million daily visitors. If these typo domains were treated as a single website, that site would be ranked by Alexa as the 36th most popular website in the world.

Expanding to the top 100 000 sites, and retaining the 0.7% estimated ratio of typosquatting visitors per popular site, we estimate that typo domains collectively receive at least 68.2 million daily visitors. If these typo domains were treated as a single website, that site would be ranked by Alexa as the 10th most popular website in the world. It would be more popular, in unique daily visitors, than twitter.com, myspace.com, or amazon.com!

How much do advertisers pay for this traffic? Table 2 reports that 57% of typo domains include Google pay-per-click ads. But prices and click-through rates vary across Google partners, and to our knowledge Google has never publicly reported its revenues from domain parking sites. To estimate Google's charges, we turn to Forbes coverage of a Trefis analyst report³ based on Google's recent SEC filings, concluding that Google's revenue per search is 3.5 cents. Meanwhile, Google's AdSense for Domains case study⁴ indicates that Google's domain parking prices are comparable to other Google prices, letting us use Google's search prices to estimate prices on typosquatting sites. Combining these factors, and extrapolating across the top 100 000 sites with the other values estimated above, we estimate that Google's revenue from typosquatting on the top 100 000 sites is \$497 million per year. In fact, comparing domain parking sites to ordinary

³ [http://blogs.forbes.com/greatspeculations/2010/01/29/
google-spending-less-to-make-more/](http://blogs.forbes.com/greatspeculations/2010/01/29/google-spending-less-to-make-more/)

⁴ <http://www.google.com/adwords/casestudies/EfficientFrontierAFDCaseStudy.pdf>

search results, we expect that the parking sites (including typo sites) have a higher click-through rate (because they typically show only ads, and no other links) and a higher conversion rate (Google's case study suggests that twice the conversion rate of search). If so, advertisers' costs for typosquatting placements could easily exceed our estimate by a factor of two or more.

7 Countering Typosquatting

7.1 Existing Efforts to Regulate Typosquatting

The rise of typosquatting in the 1990's prompted a series of regulations intended to put a check on abusive domain registrations. Initially, domain registrations were challenged primarily under trademark law, common law, and the arbitration procedures specified under domain registration agreements. For example, in arbitration arising out of Princeton Review's 1994 registration of `kaplan.com`, a panel held that Princeton Review had obtained the domain in bad faith with the objective of confusing consumers and harming Kaplan's reputation, and the panel ordered that the domain be transferred to Kaplan pursuant to the registration agreement Princeton Review had accepted upon registering the domain. In the subsequent *MTV Networks v. Curry* [8], a federal court noted similarities between domain names and mnemonic telephone numbers, suggesting that existing trademark law could apply to domain names.

After half a decade chasing cybersquatters, repeat plaintiffs offered three major complaints. First, it was increasingly burdensome to pursue many infringing domains; tens of thousands of dollars of attorney time to resolve each dispute compares unfavorably to tens of dollars for squatters to register new domains. In response, ICANN developed the Uniform Domain-Name Dispute-Resolution Policy (UDRP). For a relatively small filing fee of \$1 300 to \$4 000, complainants could seek electronic adjudication of an allegedly-infringing domains.

Second, plaintiffs faced cybersquatters who failed to disclose their true names and addresses, making a traditional lawsuit hard. The 1998 Anti-cybersquatting Consumer Protection Act (ACPA) (15 USC §1125(d)) offered an alternative, allowing a plaintiff to sue a domain *in rem* – suing the domain itself, rather than the domain's registrant. Domains were found to be at the location of the relevant registrar, registry, or other domain name authority. A plaintiff could petition a court in that jurisdiction for transfer or cancellation of a disputed domain.

Finally, plaintiffs worried that cybersquatters faced skewed incentives that invited infringements. Previously, after registering an infringing domain, a typosquatter could profit from its use until a court or arbitrator ordered the domain transferred or canceled. Cybersquatters therefore faced little real downside – at most, the forfeiture of the initial registration fee and litigation costs (minimal if the cybersquatter ignored litigation). The ACPA added the threat of significant statutory damages – \$1 000 to \$100 000 per domain name (15 USC §1117(d)). The threat of such damages were to deter would-be cybersquatters.

Private plaintiffs pursued these new mechanisms to put a check on cybersquatting. Between 1999 and August 2009, complainants invoked the UDRP

Table 4. Name servers with the most typosquatted domains

<i>Name servers > 100 000 domains</i>			<i>Name servers > 1 000 domains</i>		
name server	% typo	typos	name server	% typo	typos
dnsnameserver.org	4.75	19 217	moniker.com	61.65	910
trellian.com	4.47	11 962	ipmanagerinc.net	55.63	787
hitfarm.com	3.76	17 073	citizenhawk.net	31.88	1 766
dsredirection.com	3.60	59 845	dexner.com	18.85	375
linkz.com	2.98	3 765	aphost.com	17.96	4 244
fastpark.net	2.77	7 715	freeredirection.net	17.94	1 438
above.com	2.77	16 691	ehostinginc.com	17.89	181
sedoparking.com	2.51	35 216	nnw.net	17.10	250
parked.com	2.48	13 993	onlinednsservice.net	15.09	2 844
bottom 5	:	:	plus 97 name servers above 5% typo domains		
ipowerweb.net	0.32	569			
ipowerdns.com	0.30	522			
123-reg.co.uk	0.26	860			
abac.com	0.14	248			
vpweb.com	0.12	127			

arbitration procedure more than 45 000 times, reclaiming domains in over 85% of disputes [1]. Meanwhile, some companies pursued ACPA claims in court. Neiman Marcus filed lawsuits against typosquatters including Dotster, Name.com, and Spot Domains. Verizon sued Chinese registrar OnlineNIC, which ignored the proceedings and suffered a \$33 million default judgment for 633 typo domains of Verizon marks [10]. Microsoft sued OnlineNIC, Maltuzi, and others, and sent hundreds of subpoenas to identify typosquatters. Meanwhile, as early as 2005, Microsoft Research documented 8 923 typo domains (Internet-wide, not just for Microsoft marks) and noted how many typo domains showed PPC ads [12].

Despite thousands of complaints against typosquatting, the problem remains. 45 000 UDRP complaints represents less than 5% of the currently active typosquatting sites we found. Remarkably, even vigilant companies remain highly targeted. Months after its widely-reported judgment, Verizon still suffers at least 767 typo domains on its [verizonwireless.com](#) and [verizon.com](#) domains. For Neiman Marcus, we still see 65 typo domains, and for Microsoft 437. It seems the current approach of individual trademark holders pursuing individual squatters has not been effective in preventing or discouraging typosquatting by others. Therefore, we next consider methods to influence companies that distinctively benefit from typosquatting: domain aggregators and advertising platforms.

7.2 Identifying Servers That Distinctively Host Typo Domains

Large domainers typically host their domains on a single set of name servers. By comparing the incidence of typo domains across name servers, we assess which name servers host disproportionately many typo domains.

Our analysis found 937 918 typo domains out of 80 988 864 .com domains; consequently, any name server with over 1.16% typo domains is above average. Table 4 (left) shows the incidence of typosquatting at large name servers. Many large parking companies identified in Section 4.1 have disproportionately many

typo domains: 2.5% of domains resolved by `sedoparking.com` are typos, over twice the rate on the web as a whole. On smaller name servers, typo domains can be even more frequent. Table 4 (right) considers name servers hosting at least 1 000 names. Topping the list is `moniker.com`, with 62% typo domains.

At the same time, other name servers feature disproportionately infrequent typo domains. For example, the bottom of Table 4 (left) shows large name servers with typo domains as infrequent as 0.12%, one tenth the Internet-wide average.

7.3 The Role and Responsibility of Ad Platforms

We pause for an important disclosure: One of the authors (Edelman) is co-counsel in litigation seeking to hold Google liable for using typosquatting domains to display advertising [11]. However, we now write not as lawyers but as engineer and economist seeking to address typosquatting in the most efficient way possible.

As shown in Section 3.1 and Table 2, of the typo domains we successfully crawled, nearly 80% showed pay-per-click advertisements that came from the ad platforms operated by the web’s top search engines, principally Google and (to a significantly lesser extent) Yahoo. Because ad platforms are the primary or sole source of revenue for these typo domains, we believe ad platforms are well-positioned to substantially reduce typosquatting. Among other responses, ad platforms could select partners more carefully, select only partners with a demonstrated record of avoiding typosquatting, and/or sever ties to partners who are found to engage in typosquatting. Furthermore, ad platforms could require that new partners showing ads on many domains post a bond that is forfeited upon typosquatting, or deduct penalties from payments to any partners found to engage in typosquatting. To the best of our knowledge, ad platforms have taken none of these steps.

Ad platforms typically claim that a website or trademark owner targeted by typosquatting should address its complaint directly to the typosquatter, not to the ad platform that pays the typosquatter. For example, Google’s AdSense for Domains complaint page argues that “Google is not in any way involved with the selection or registration of these domain names, and is not in a position to arbitrate trademark disputes between the registrants, our partners, and trademark owners. Accordingly, we encourage trademark owners to resolve their disputes directly with the registrants or registrars.”⁵ By stepping out of disputes between sites and typosquatters, ad platforms’ preferred approach simplifies disputes (to entail two parties rather than three) and, of course, limits ad platforms’ potential liability.

Despite the simplification resulting from ad platforms’ preferred approach, we see multiple problems with ad platforms disclaiming all responsibility for the typosquatting they fund. For one, our analysis confirms that payments from ad platforms are the sole force behind most typosquatting registrations. Furthermore, ad platforms are least-cost avoiders – able to prevent typosquatting with

⁵ <http://adwords.google.com/support/aw/bin/answer.py?answer=50003&topic=26>

less effort than any other party. In particular, thanks to the semantic analysis capabilities and spelling correction skills search engines gained through their principal businesses, ad platforms are well equipped to identify typosquatting registrations. (Consider Google’s well-known and strikingly accurate “Did you mean?” function.) Indeed, search engines already receive information about the domains users visit (necessary to target ads accordingly). It would be straightforward to compare these requests to a list of top trademarks, and disallow parking ads from appearing on domains that are misspellings of popular sites.

The dynamics of the typosquatting business give ad platforms a particularly powerful opportunity to undermine typosquatting. Suppose a site owner pursued a few large typosquatters. The associated typo domains would tend to scatter to numerous smaller typosquatters who could not be identified, located, or pursued cost-effectively (as has already happened to Microsoft, Verizon and others). In contrast, ad platforms enjoy unique positions of authority, buttressed by their relationships with advertisers. Consequently, ad platforms can authoritatively undermine typosquatting, in a way that no individual site owner can.

8 Conclusions

We are struck by the scale of the problem of typosquatting – at least many hundreds of thousands of typo domains, and probably millions – despite substantial public and private efforts to discourage such registrations. Yet with such strong economics supporting typosquatting – payments from Google and others – perhaps it is no surprise that typosquatting is as prevalent as ever.

We suspect typosquatting will continue so long as advertisers and ad networks continue to fuel and fund these practices. But let no one suggest identifying typo domains is impossible: The overwhelming majority of typos are easy to recognize, by hand or using straightforward automation. At the same time, with typo domains highly concentrated at a few large domainers and ad platforms, intermediaries could significantly discourage the registration and use of typo domains if they were so inclined.

References

1. Brand Owners Could Have Prevented \$220 Million. In: Domain Name Recovery By Spending \$1 Million. Corporation Service Company (August 24, 2009)
2. Damerau, F.J.: A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM* 7(3), 171–176 (1964)
3. Edelman, B.: Domains Reregistered for Distribution of Unrelated Content: A Case Study of Tina’s Free Live Webcam (2002), <http://cyber.law.harvard.edu/people/edelman/renewals/>
4. Edelman, B.: How Google and Its Partners Inflate Measured Conversion Rates and Increase Advertisers’ Costs (2009), <http://www.benedelman.org/news/051309-1.html>
5. Edelman, B.: Large-Scale Registration of Domains with Typographical Errors (2003), <http://cyber.law.harvard.edu/people/edelman/typo-domains/>

6. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics Doklady (1966)
7. Lands' End, Inc. v. Eric Remy, et al, W.D.Wis (2006)
8. MTV Networks v. Curry, 867 F.Supp. 202. SDNY (1994)
9. The Neiman Marcus Group Inc., et al., v. Dotster Inc., et al, W.D.Wa (2006)
10. Perez, M.: Verizon wins \$33 Million In Cybersquatting Case. Information Week (December 30, 2008)
11. Vulcan Golf, LLC, et al., v. Google, Inc., et al. N.D.Ill. Case No 1:2007cv03371
12. Wang, Y., Beck, D., Wang, J., Verbowski, C., Daniels, B.: Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting. In: 2nd USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI) (July 2006)

A Learning-Based Approach to Reactive Security

Adam Barth¹, Benjamin I.P. Rubinstein¹, Mukund Sundararajan³,
John C. Mitchell⁴, Dawn Song¹, and Peter L. Bartlett^{1,2}

¹ Computer Science Division

² Department of Statistics, UC Berkeley

³ Google Inc., Mountain View, CA

⁴ Department of Computer Science, Stanford University

Abstract. Despite the conventional wisdom that proactive security is superior to reactive security, we show that reactive security can be competitive with proactive security as long as the reactive defender learns from past attacks instead of myopically overreacting to the last attack. Our game-theoretic model follows common practice in the security literature by making worst-case assumptions about the attacker: we grant the attacker complete knowledge of the defender’s strategy and do not require the attacker to act rationally. In this model, we bound the competitive ratio between a reactive defense algorithm (which is inspired by online learning theory) and the best fixed proactive defense. Additionally, we show that, unlike proactive defenses, this reactive strategy is robust to a lack of information about the attacker’s incentives and knowledge.

1 Introduction

Many enterprises employ a Chief Information Security Officer (CISO) to manage the enterprise’s information security risks. Typically, an enterprise has many more security vulnerabilities than it can realistically repair. Instead of declaring the enterprise “insecure” until every last vulnerability is plugged, CISOs typically perform a cost-benefit analysis to identify which risks to address, but what constitutes an effective CISO strategy? The conventional wisdom [28,21] is that CISOs ought to adopt a “forward-looking” proactive approach to mitigating security risk by examining the enterprise for vulnerabilities that might be exploited in the future. Advocates of proactive security often equate reactive security with myopic bug-chasing and consider it ineffective. We establish sufficient conditions for when reacting *strategically* to attacks is as effective in discouraging attackers.

We study the efficacy of reactive strategies in an economic model of the CISO’s security cost-benefit trade-offs. Unlike previously proposed economic models of security (see Section 7), we do not assume the attacker acts according to a fixed probability distribution. Instead, we consider a game-theoretic model with a strategic attacker who responds to the defender’s strategy. As is standard in the security literature, we make worst-case assumptions about the attacker. For example, we grant the attacker complete knowledge of the defender’s strategy and do not require the attacker to act rationally. Further, we make conservative

assumptions about the reactive defender's knowledge and do not assume the defender knows all the vulnerabilities in the system or the attacker's incentives. However, we do assume that the defender can observe the attacker's past actions, for example via an intrusion detection system or user metrics [4].

In our model, we find that two properties are sufficient for a reactive strategy to perform as well as the best proactive strategies. First, no single attack is catastrophic, meaning the defender can survive a number of attacks. This is consistent with situations where intrusions (that, say, steal credit card numbers) are regrettable but not business-ending. Second, the defender's budget is *liquid*, meaning the defender can re-allocate resources without penalty. For example, a CISO can reassign members of the security team from managing firewall rules to improving database access controls at relatively low switching costs.

Because our model abstracts many vulnerabilities into a single graph edge, we view the act of defense as increasing the attacker's *cost* for mounting an attack instead of preventing the attack (e.g., by patching a single bug). By making this assumption, we choose not to study the tactical patch-by-patch interaction of the attacker and defender. Instead, we model enterprise security at a more abstract level appropriate for the CISO. For example, the CISO might allocate a portion of his or her budget to engage a consultancy, such as WhiteHat or iSEC Partners, to find and fix cross-site scripting in a particular web application or to require that employees use SecurID tokens during authentication. We make the technical assumption that attacker costs are linearly dependent on defense investments locally. This assumption does not reflect patch-by-patch interaction, which would be better represented by a step function (with the step placed at the cost to deploy the patch). Instead, this assumption reflects the CISO's higher-level viewpoint where the staircase of summed step functions fades into a slope.

We evaluate the defender's strategy by measuring the attacker's cumulative return-on-investment, the *return-on-attack* (ROA), which has been proposed previously [8]. By studying this metric, we focus on defenders who seek to "cut off the attacker's oxygen," that is to reduce the attacker's incentives for attacking the enterprise. We do not distinguish between "successful" and "unsuccessful" attacks. Instead, we compare the payoff the attacker receives from his or her nefarious deeds with the cost of performing said deeds. We imagine that sufficiently disincentivized attackers will seek alternatives, such as attacking a different organization or starting a legitimate business.

In our main result, we show sufficient conditions for a learning-based reactive strategy to be competitive with the best fixed proactive defense in the sense that the competitive ratio between the reactive ROA and the proactive ROA is at most $1 + \epsilon$, for all $\epsilon > 0$, provided the game lasts sufficiently many rounds (at least $\Omega(1/\epsilon)$). To prove our theorems, we draw on techniques from the online learning literature. We extend these techniques to the case where the learner does not know all the game matrix rows *a priori*, letting us analyze situations where the defender does not know all the vulnerabilities in advance. Although our main results are in a graph-based model with a single attacker, our results generalize to a model based on Horn clauses with multiple attackers. Our results

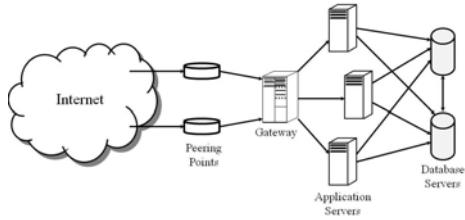


Fig. 1. An attack graph representing an enterprise data center

are also robust to switching from ROA to attacker profit and to allowing the proactive defender to revise the defense allocation a fixed number of times.

Although myopic bug chasing is most likely an ineffective reactive strategy, we find that in some situations a *strategic* reactive strategy is as effective as the optimal fixed proactive defense. In fact, we find that the natural strategy of gradually reinforcing attacked edges by shifting budget from unattacked edges “learns” the attacker’s incentives and constructs an effective defense. Such a strategic reactive strategy is both easier to implement than a proactive strategy—because it does not presume that the defender knows the attacker’s intent and capabilities—and is less wasteful than a proactive strategy because the defender does not expend budget on attacks that do not actually occur. Based on our results, we encourage CISOs to question the assumption that proactive risk management is inherently superior to reactive risk management.

Organization. Section 2 formalizes our model. Section 3 shows that perimeter defense and defense-in-depth arise naturally in our model. Section 4 presents our main results bounding the competitive ratio of reactive versus proactive defense strategies. Section 5 outlines scenarios in which reactive security out-performs proactive security. Section 6 generalizes our results to Horn clauses and multiple attackers. Section 7 relates related work. Section 8 concludes.

2 Formal Model

In this section, we present a game-theoretic model of attack and defense. Unlike traditional bug-level attack graphs, our model is meant to capture a managerial perspective on enterprise security. The model is somewhat general in the sense that attack graphs can represent a number of concrete situations, including a network (see Figure 1), components in a complex software system [9], or an Internet Fraud “Battlefield” [13].

System. We model a system using a directed graph (V, E) , which defines the game between an attacker and a defender. Each vertex $v \in V$ in the graph represents a state of the system. Each edge $e \in E$ represents a state transition the attacker can induce. For example, a vertex might represent whether a particular machine in a network has been compromised by an attacker. An edge from one machine to another might represent that an attacker who has compromised the

first machine might be able to compromise the second machine because the two are connected by a network. Alternatively, the vertices might represent different components in a software system. An edge might represent that an attacker sending input to the first component can send input to the second.

In attacking the system, the attacker selects a path in the graph that begins with a designated *start vertex* s . Our results hold in more general models (e.g., based on Horn clauses), but we defer discussing such generalizations until Section 6. We think of the attack as driving the system through the series of state transitions indicated by the edges included in the path. In the networking example in Figure 1, an attacker might first compromise a front-end server and then leverage the server’s connectivity to the back-end database server to steal credit card numbers from the database.

Incentives and Rewards. Attackers respond to incentives. For example, attackers compromise machines and form botnets because they make money from spam [20] or rent the botnet to others [32]. Other attackers steal credit card numbers because credit card numbers have monetary value [10]. We model the attacker’s incentives by attaching a non-negative *reward* to each vertex. These rewards are the utility the attacker derives from driving the system into the state represented by the vertex. For example, compromising the database server might have a sizable reward because the database server contains easily monetizable credit card numbers. We assume the start vertex has zero reward, forcing the attacker to undertake some action before earning utility. Whenever the attacker mounts an attack, the attacker receives a *payoff* equal to the sum of the rewards of the vertices visited in the attack path: $\text{payoff}(a) = \sum_{v \in a} \text{reward}(v)$. In the example from Figure 1, if an attacker compromises both a front-end server and the database server, the attacker receives both rewards.

Attack Surface and Cost. The defender has a fixed *defense budget* $B > 0$, which the defender can divide among the edges in the graph according to a *defense allocation* d : for all $e \in E$, $d(e) \geq 0$ and $\sum_{e \in E} d(e) \leq B$.

The defender’s allocation of budget to various edges corresponds to the decisions made by the Chief Information Security Officer (CISO) about where to allocate the enterprise’s security resources. For example, the CISO might allocate organizational headcount to fuzzing enterprise web applications for XSS vulnerabilities. These kinds of investments are continuous in the sense that the CISO can allocate 1/4 of a full-time employee to worrying about XSS. We denote the set of feasible allocations of budget B on edge set E by $\mathcal{D}_{B,E}$.

By defending an edge, the defender makes it more difficult for the attacker to use that edge in an attack. Each unit of budget the defender allocates to an edge raises the cost that the attacker must pay to use that edge in an attack. Each edge has an *attack surface* [19] w that represents the difficulty in defending against that state transition. For example, a server that runs both Apache and Sendmail has a larger attack surface than one that runs only Apache because defending the first server is more difficult than the second. Formally, the attacker must pay the following *cost* to traverse the edge: $\text{cost}(a, d) = \sum_{e \in a} d(e)/w(e)$. Allocating defense budget to an edge does not “reduce” an edge’s attack surface.

For example, consider defending a hallway with bricks. The wider the hallway (the larger the attack surface), the more bricks (budget allocation) required to build a wall of a certain height (the cost to the attacker).

In this formulation, the function mapping the defender’s budget allocation to attacker cost is linear, preventing the defender from ever fully defending an edge. Our use of a linear function reflects a level of abstraction more appropriate to a CISO who can never fully defend assets, which we justify by observing that the rate of vulnerability discovery in a particular piece of software is roughly constant [29]. At a lower level of detail, we might replace this function with a step function, indicating that the defender can “patch” a vulnerability by allocating a threshold amount of budget.

Objective. To evaluate defense strategies, we measure the attacker’s incentive for attacking using the *return-on-attack* (ROA) [8], which we define as follows:

$$\text{ROA}(a, d) = \frac{\text{payoff}(a)}{\text{cost}(a, d)}$$

We use this metric for evaluating defense strategy because we believe that if the defender lowers the ROA sufficiently, the attacker will be discouraged from attacking the system and will find other uses for his or her capital or industry. For example, the attacker might decide to attack another system. Analogous results hold if we quantify the attacker’s incentives in terms of profit (e.g., with $\text{profit}(a, d) = \text{payoff}(a) - \text{cost}(a, d)$), but we focus on ROA for simplicity.

A purely rational attacker will mount attacks that maximize ROA. However, a real attacker might not maximize ROA. For example, the attacker might not have complete knowledge of the system or its defense. We strengthen our results by considering all attacks, not just those that maximize ROA.

Proactive Security. We evaluate our learning-based reactive approach by comparing it against a *proactive* approach to risk management in which the defender carefully examines the system and constructs a defense in order to fend off future attacks. We strengthen this benchmark by providing the proactive defender complete knowledge about the system, but we require that the defender commit to a fixed strategy. To strengthen our results, we state our main result in terms of *all* such proactive defenders. In particular, this class of defenders includes the *rational proactive defender* who employs a defense allocation that minimizes the maximum ROA the attacker can extract from the system: $\text{argmin}_d \max_a \text{ROA}(a, d)$.

3 Case Studies

In this section, we describe instances of our model to build the reader’s intuition. These examples illustrate that some familiar security concepts, including perimeter defense and defense in depth, arise naturally as optimal defenses in our model. These defenses can be constructed either by rational proactive attackers or converged to by a learning-based reactive defense.



Fig. 2. Attack graph representing a simplified data center network

Perimeter Defense. Consider a system in which the attacker’s reward is non-zero at exactly one vertex, t . For example, in a medical system, the attacker’s reward for obtaining electronic medical records might well dominate the value of other attack targets such as employees’ vacation calendars. In such a system, a rational attacker will select the minimum-cost path from the start vertex s to the valuable vertex t . The optimal defense limits the attacker’s ROA by maximizing the cost of the minimum s - t path. The algorithm for constructing this defense is straightforward [7]:

1. Let C be the minimum weight s - t cut in (V, E, w) .
2. Select the following defense:

$$d(e) = \begin{cases} Bw(e)/Z & \text{if } e \in C \\ 0 & \text{otherwise} \end{cases}, \quad \text{where } Z = \sum_{e \in C} w(e).$$

Notice that this algorithm constructs a *perimeter defense*: the defender allocates the entire defense budget to a single cut in the graph. Essentially, the defender spreads the defense budget over the attack surface of the cut. By choosing the minimum-weight cut, the defender is choosing to defend the smallest attack surface that separates the start vertex from the target vertex. Real defenders use similar perimeter defenses, for example, when they install a firewall at the boundary between their organization and the Internet because the network’s perimeter is much smaller than its interior.

Defense in Depth. Many experts in security practice recommend that defenders employ defense in depth. Defense in depth rises naturally in our model as an optimal defense for some systems. Consider, for example, the system depicted in Figure 2. This attack graph is a simplified version of the data center network depicted in Figure 1. Although the attacker receives the largest reward for compromising the back-end database server, the attacker also receives some reward for compromising the front-end web server. Moreover, the front-end web server has a larger attack surface than the back-end database server because the front-end server exposes a more complex interface (an entire enterprise web application), whereas the database server exposes only a simple SQL interface. Allocating defense budget to the left-most edge represents trying to protect sensitive database information with a complex web application firewall instead of database access control lists (i.e., possible, but economically inefficient).

The optimal defense against a rational attacker is to allocate half of the defense budget to the left-most edge and half of the budget to the right-most edge, limiting the attacker to a ROA of unity. Shifting the entire budget to the

right-most edge (i.e., defending only the database) is disastrous because the attacker will simply attack the front-end at zero cost, achieving an unbounded ROA. Shifting the entire budget to the left-most edge is also problematic because the attacker will attack the database (achieving an ROA of 5).

4 Reactive Security

To analyze reactive security, we model the attacker and defender as playing an iterative game, alternating moves. First, the defender selects a defense, and then the attacker selects an attack. We present a learning-based reactive defense strategy that is oblivious to vertex rewards and to edges that have not yet been used in attacks. We prove a theorem bounding the competitive ratio between this reactive strategy and the best proactive defense via a series of reductions to results from the online learning theory literature. Other applications of this literature include managing stock portfolios [26], playing zero-sum games [12], and boosting other machine learning heuristics [11]. Although we provide a few technical extensions, our main contribution comes from applying results from online learning to risk management.

Repeated Game. We formalize the repeated game between the defender and the attacker as follows. In each round t from 1 to T :

1. The defender chooses defense allocation $d_t(e)$ over the edges $e \in E$.
2. The attacker chooses an attack path a_t in G .
3. The path a_t and attack surfaces $\{w(e) : e \in a_t\}$ are revealed to the defender.
4. The attacker pays cost(a_t, d_t) and gains payoff(a_t).

In each round, we let the attacker choose the attack path after the defender commits to the defense allocation because the defender’s budget allocation is not a secret (in the sense of a cryptographic key). Following the “no security through obscurity” principle, we make the conservative assumption that the attacker can accurately determine the defender’s budget allocation.

Defender Knowledge. Unlike proactive defenders, reactive defenders do not know all of the vulnerabilities that exist in the system in advance. (If defenders had complete knowledge of vulnerabilities, conferences such as Black Hat Briefings would serve little purpose.) Instead, we reveal an edge (and its attack surface) to the defender after the attacker uses the edge in an attack. For example, the defender might monitor the system and learn how the attacker attacked the system by doing a post-mortem analysis of intrusion logs. Formally, we define a *reactive defense strategy* to be a function from attack sequences $\{a_i\}$ and the subsystem induced by the edges contained in $\bigcup_i a_i$ to defense allocations such that $d(e) = 0$ if edge $e \notin \bigcup_i a_i$. Notice that this requires the defender’s strategy to be oblivious to the system beyond the edges used by the attacker.

Algorithm 1. A reactive defense strategy for hidden edges.

- Initialize $E_0 = \emptyset$
- For each round $t \in \{2, \dots, T\}$
 - Let $E_{t-1} = E_{t-2} \cup E(a_{t-1})$
 - For each $e \in E_{t-1}$, let

$$S_{t-1}(e) = \begin{cases} S_{t-2}(e) + M(e, a_{t-1}) & \text{if } e \in E_{t-2} \\ M(e, a_{t-1}) & \text{otherwise.} \end{cases}$$

$$\tilde{P}_t(e) = \beta_{t-1}^{S_{t-1}(e)}$$

$$P_t(e) = \frac{\tilde{P}_t(e)}{\sum_{e' \in E_t} \tilde{P}_t(e')} ,$$

where $M(e, a) = -\mathbf{1}[e \in a]/w(e)$ is a matrix with $|E|$ rows and a column for each attack.

Algorithm. Algorithm 1 is a reactive defense strategy based on the multiplicative update learning algorithm [6,12]. The algorithm reinforces edges on the attack path multiplicatively, taking the attack surface into account by allocating more budget to easier-to-defend edges. When new edges are revealed, the algorithm re-allocates budget uniformly from the already-revealed edges to the newly revealed edges. We state the algorithm in terms of a normalized defense allocation $P_t(e) = d_t(e)/B$. Notice that this algorithm is oblivious to unattacked edges and the attacker's reward for visiting each vertex. An appropriate setting for the algorithm parameters $\beta_t \in [0, 1]$ will be described below.

The algorithm begins without any knowledge of the graph whatsoever, and so allocates no defense budget to the system. Upon the t^{th} attack on the system, the algorithm updates E_t to be the set of edges revealed up to this point, and updates $S_t(e)$ to be a weight count of the number of times e has been used in an attack thus far. For each edge that has ever been revealed, the defense allocation $P_{t+1}(e)$ is chosen to be $\beta_t^{S_t(e)}$ normalized to sum to unity over all edges $e \in E_t$. In this way, any edge attacked in round t will have its defense allocation reinforced.

The parameter β controls how aggressively the defender reallocates defense budget to recently attacked edges. If β is infinitesimal, the defender will move the entire defense budget to the edge on the most recent attack path with the smallest attack surface. If β is enormous, the defender will not be very agile and, instead, leave the defense budget in the initial allocation. For an appropriate value of β , the algorithm will converge to the optimal defense strategy. For instance, the min cut in the example from Section 3.

Theorems. To compare this reactive defense strategy to all proactive defense strategies, we use the notion of *regret* from online learning theory. The following is an additive regret bound relating the attacker's profit under reactive and proactive defense strategies.

Theorem 1. *The average attacker profit against Algorithm 1 converges to the average attacker profit against the best proactive defense. Formally, if defense allocations $\{d_t\}_{t=1}^T$ are output by Algorithm 1 with parameter sequence $\beta_s = \left(1 + \sqrt{2 \log |E_s| / (s+1)}\right)^{-1}$ on any system $(V, E, w, \text{reward}, s)$ revealed online and any attack sequence $\{a_t\}_{t=1}^T$, then*

$$\frac{1}{T} \sum_{t=1}^T \text{profit}(a_t, d_t) - \frac{1}{T} \sum_{t=1}^T \text{profit}(a_t, d^*) \leq B \sqrt{\frac{\log |E|}{2T}} + \frac{B(\log |E| + \overline{w^{-1}})}{T} ,$$

for all proactive defense strategies $d^* \in \mathcal{D}_{B,E}$ where $\overline{w^{-1}} = |E|^{-1} \sum_{e \in E} w(e)^{-1}$, the mean of the surface reciprocals.

Remark 2. *We can interpret Theorem 1 as establishing sufficient conditions under which a reactive defense strategy is within an additive constant of the best proactive defense strategy. Instead of carefully analyzing the system to construct the best proactive defense, the defender need only react to attacks in a principled manner to achieve almost the same quality of defense in terms of attacker profit.*

Reactive defense strategies can also be competitive with proactive defense strategies when we consider an attacker motivated by return on attack (ROA). The ROA formulation is appealing because (unlike with profit) the objective function does not require measuring attacker cost and defender budget in the same units. The next result considers the competitive ratio between the ROA for a reactive defense strategy and the ROA for the best proactive defense strategy.

Theorem 3. *The ROA against Algorithm 1 converges to the ROA against best proactive defense. Formally, consider the cumulative ROA:*

$$\text{ROA} (\{a_t\}_{t=1}^T, \{d_t\}_{t=1}^T) = \frac{\sum_{t=1}^T \text{payoff}(a_t)}{\sum_{t=1}^T \text{cost}(a_t, d_t)}$$

(We abuse notation slightly and use singleton arguments to represent the corresponding constant sequence.) If defense allocations $\{d_t\}_{t=1}^T$ are output by Algorithm 1 with parameters $\beta_s = \left(1 + \sqrt{2 \log |E_s| / (s+1)}\right)^{-1}$ on any system $(V, E, w, \text{reward}, s)$ revealed online, such that $|E| > 1$, and any attack sequence $\{a_t\}_{t=1}^T$, then for all $\alpha > 0$ and proactive defense strategies $d^* \in \mathcal{D}_{B,E}$

$$\frac{\text{ROA} (\{a_t\}_{t=1}^T, \{d_t\}_{t=1}^T)}{\text{ROA} (\{a_t\}_{t=1}^T, d^*)} \leq 1 + \alpha ,$$

provided T is sufficiently large.¹

Remark 4. *Notice that the reactive defender can use the same algorithm regardless of whether the attacker is motivated by profit or by ROA. As discussed in Section 5 the optimal proactive defense is not similarly robust.*

¹ To wit: $T \geq \left(\frac{13}{\sqrt{2}} (1 + \alpha^{-1}) \left(\sum_{e \in \text{inc}(s)} w(e)\right)\right)^2 \log |E|$.

We present proofs of these theorems in the full version [3]. We first prove the theorems in the simpler setting where the defender knows the entire graph. Second, we remove the hypothesis that the defender knows the edges in advance.

Lower Bounds. In the full version [3], we use a two-vertex, two-edge graph to establish a lower bound on the competitive ratio of the ROA for all reactive strategies. The lower bound shows that the analysis of Algorithm 1 is tight and that Algorithm 1 is optimal given the information available to the algorithm. The proof gives an example where the best proactive defense (slightly) out-performs every reactive strategy, suggesting the benchmark is not unreasonably weak.

5 Advantages of Reactivity

In this section, we examine some situations in which a reactive defender outperforms a proactive defender. Proactive defenses hinge on the defender’s model of the attacker’s incentives. If the defender’s model is inaccurate, the defender will construct a proactive defense that is far from optimal. By contrast, a reactive defender need not reason about the attacker’s incentives directly. Instead, the reactive defender learns these incentives by observing the attacker in action.

Learning Rewards. One way to model inaccuracies in the defender’s estimates of the attacker’s incentives is to hide the attacker’s rewards from the defender. Without knowledge of the payoffs, a proactive defender has difficulty limiting the attacker’s ROA. Consider, for example, the star system whose edges have equal attack surfaces, as depicted in Figure 3. Without knowledge of the attacker’s rewards, a proactive defender has little choice but to allocate the defense budget equally to each edge (because the edges are indistinguishable). However, if the attacker’s reward is concentrated at a single vertex, the competitive ratio for attacker’s ROA (compared to the rational proactive defense) is the number of leaf vertices. (We can, of course, make the ratio worse by adding more vertices.) By contrast, the reactive algorithm we analyze in Section 4 is competitive with the rational proactive defense because the reactive algorithm effectively learns the rewards by observing which attacks the attacker chooses.

Robustness to Objective. Another way to model inaccuracies in the defender’s estimates of the attacker’s incentives is to assume the defender mistakes which of profit and ROA actually matter to the attacker. The defense constructed by a rational proactive defender depends crucially on whether the attacker’s actual incentives are based on profit or based on ROA, whereas the reactive algorithm we analyze in Section 4 is robust to this variation. In particular, consider the system depicted in Figure 4, and assume the defender has a budget of 9. If the defender believes the attacker is motivated by profit, the rational proactive defense is to allocate the entire defense budget to the right-most edge (making the profit 1 on both edges). However, this defense is disastrous when viewed in terms of ROA because the ROA for the left edge is infinite (as opposed to near unity when the proactive defender optimizes for ROA).

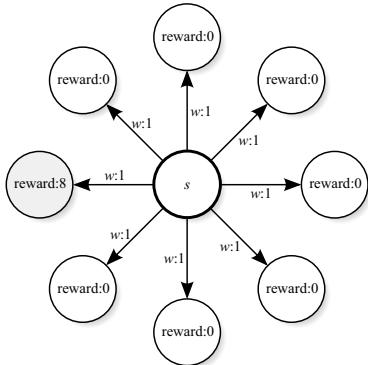


Fig. 3. Star-shaped attack graph with rewards concentrated in an unknown vertex

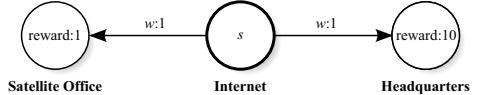


Fig. 4. An attack graph that separates the minimax strategies optimizing ROA and attacker profit

Catachresis. The defense constructed by the rational proactive defender is optimized for a rational attacker. If the attacker is not perfectly rational, there is room for out-performing the rational proactive defense. There are a number of situations in which the attacker might not mount “optimal” attacks:

- The attacker might not have complete knowledge of the attack graph. Consider, for example, a software vendor who discovers five equally severe vulnerabilities in one of their products via fuzzing. According to proactive security, the defender ought to dedicate equal resources to repairing these five vulnerabilities. However, a reactive defender might dedicate more resources to fixing a vulnerability actually exploited by attackers in the wild. We can model these situations by making the attacker oblivious to some edges.
- The attacker might not have complete knowledge of the defense allocation. For example, an attacker attempting to invade a corporate network might target computers in human resources without realizing that attacking the customer relationship management database in sales has a higher return-on-attack because the database is lightly defended.

By observing attacks, the reactive strategy learns a defense tuned for the *actual* attacker, causing the attacker to receive a lower ROA.

6 Generalizations

Horn Clauses. Thus far, we have presented our results using a graph-based system model. Our results extend, however, to a more general system model based on Horn clauses. Datalog programs, which are based on Horn clauses, have been used in previous work to represent vulnerability-level attack graphs [27]. A Horn clause is a statement in propositional logic of the form $p_1 \wedge p_2 \wedge \dots \wedge p_n \rightarrow q$. The propositions p_1, p_2, \dots, p_n are called the *antecedents*, and q is called the

consequent. The set of antecedents might be empty, in which case the clause simply asserts the consequent. Notice that Horn clauses are negation-free. In some sense, a Horn clause represents an edge in a hypergraph where multiple pre-conditions are required before taking a certain state transition.

In the Horn model, a system consists of a set of Horn clauses, an attack surface for each clause, and a reward for each proposition. The defender allocates defense budget among the Horn clauses. To mount an attack, the attacker selects a *valid proof*: an ordered list of rules such that each antecedent appears as a consequent of a rule earlier in the list. For a given proof Π ,

$$\text{cost}(\Pi, d) = \sum_{c \in \Pi} d(c)/w(e) \quad \text{payoff}(\Pi) = \sum_{p \in [\Pi]} \text{reward}(p) ,$$

where $[\Pi]$ is the set of propositions proved by Π (i.e., those propositions that appear as consequents in Π). Profit and ROA are computed as before.

Our results generalize to this model directly. Essentially, we need only replace each instance of the word “edge” with “Horn clause” and “path” with “valid proof.” For example, the rows of the matrix M used throughout the proof become the Horn clauses, and the columns become the valid proofs (which are numerous, but no matter). The entries of the matrix become $M(c, \Pi) = 1/w(c)$, analogous to the graph case. The one non-obvious substitution is $\text{inc}(s)$, which becomes the set of clauses that lack antecedents.

Multiple Attackers. We have focused on a security game between a *single* attacker and a defender. In practice, a security system might be attacked by several uncoordinated attackers, each with different information and different objectives. Fortunately, we can show that a model with multiple attackers is mathematically equivalent to a model with a single attacker with a randomized strategy: Use the set of attacks, one per attacker, to define a distribution over edges where the probability of an edge is linearly proportional to the number of attacks which use the edge. This precludes the interpretation of an attack as an s -rooted path, but our proofs do not rely upon this interpretation and our results hold in such a model with appropriate modifications.

Adaptive Proactive Defenders. A simple application of an online learning result [18], omitted due to space constraints, modifies our regret bounds for a proactive defender who re-allocates budget a fixed number of times. In this model, our results remain qualitatively the same.

7 Related Work

Anderson [1] and Varian [31] informally discuss (via anecdotes) how the design of information security must take incentives into account. August and Tunca [2] compare various ways to incentivize users to patch their systems in a setting where the users are more susceptible to attacks if their neighbors do not patch.

Gordon and Loeb [15] and Hausken [17] analyze the costs and benefits of security in an economic model (with non-strategic attackers) where the probability

of a successful exploit is a function of the defense investment. They use this model to compute the optimal level of investment. Varian [30] studies various (single-shot) security games and identifies how much agents invest in security at equilibrium. Grossklags [16] extends this model by letting agents self-insure.

Miura et al. [24] study externalities that appear due to users having the same password across various websites and discuss pareto-improving security investments. Miura and Bamboz [25] rank vulnerabilities according to a random-attacker model. Skybox and RedSeal offer practical systems that help enterprises prioritize vulnerabilities based on a random-attacker model. Kumar et al. [22] investigate optimal security architectures for a multi-division enterprise, taking into account losses due to lack of availability and confidentiality. None of the above papers explicitly model a truly adversarial attacker.

Fultz [14] generalizes [16] by modeling attackers explicitly. Cavusoglu et al. [5] highlight the importance of using a game-theoretic model over a decision theoretic model due to the presence of adversarial attackers. However, these models look at idealized settings that are not generically applicable. Lye and Wing [23] study the Nash equilibrium of a single-shot game between an attacker and a defender that models a particular enterprise security scenario. Arguably this model is most similar to ours in terms of abstraction level. However, calculating the Nash equilibrium requires detailed knowledge of the adversary's incentives, which as discussed in the introduction, might not be readily available to the defender. Moreover, their game contains multiple equilibria, weakening their prescriptions.

8 Conclusions

Many security experts equate reactive security with myopic bug-chasing and ignore principled reactive strategies when they recommend adopting a proactive approach to risk management. In this paper, we establish sufficient conditions for a learning-based reactive strategy to be competitive with the best fixed proactive defense. Additionally, we show that reactive defenders can out-perform proactive defenders when the proactive defender defends against attacks that never actually occur. Although our model is an abstraction of the complex interplay between attackers and defenders, our results support the following practical advice for CISOs making security investments:

- Employ monitoring tools that let you detect and analyze attacks against your enterprise. These tools help focus your efforts on thwarting real attacks.
- Make your security organization more agile. For example, build a rigorous testing lab that lets you roll out security patches quickly once you detect that attackers are exploiting these vulnerabilities.
- When determining how to expend your security budget, avoid overreacting to the most recent attack. Instead, consider all previous attacks, but discount the importance of past attacks exponentially.

In some situations, proactive security can out-perform reactive security. For example, reactive approaches are ill-suited for defending against catastrophic

attacks because there is no “next round” in which the defender can use information learned from the attack. We hope our results will lead to a productive discussion of the limitations of our model and the validity of our conclusions.

Instead of assuming that proactive security is always superior to reactive security, we invite the reader to consider when a reactive approach might be appropriate. For the parts of an enterprise where the defender’s budget is liquid and there are no catastrophic losses, a carefully constructed reactive strategy can be as effective as the best proactive defense in the worst case and significantly better in the best case.

Acknowledgments. We would like to thank Elie Bursztein, Eu-Jin Goh, and Matt Finifter for their thoughtful comments and helpful feedback. We gratefully acknowledge the support of the NSF through the TRUST Science and Technology Center and grants DMS-0707060, CCF-0424422, 0311808, 0448452, and 0627511, and the support of the AFOSR through the MURI Program, and the support of the Siebel Scholars Foundation.

References

1. Anderson, R.: Why information security is hard—An economic perspective. In: 17th Annual Computer Security Applications Conference, pp. 358–365 (2001)
2. August, T., Tunca, T.I.: Network software security and user incentives. *Management Science* 52(11), 1703–1720 (2006)
3. Barth, A., Rubinstein, B.I.P., Sundararajan, M., Mitchell, J.C., Song, D., Bartlett, P.L.: A learning-based approach to reactive security (2009), <http://arxiv.org/abs/0912.1155>
4. Beard, C.: Introducing Test Pilot (March 2008), <http://labs.mozilla.com/2008/03/introducing-test-pilot/>
5. Cavusoglu, H., Raghunathan, S., Yue, W.: Decision-theoretic and game-theoretic approaches to IT security investment. *Journal of Management Information Systems* 25(2), 281–304 (2008)
6. Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D.P., Schapire, R.E., Warmuth, M.K.: How to use expert advice. *Journal of the Association for Computing Machinery* 44(3), 427–485 (1997)
7. Chakrabarty, D., Mehta, A., Vazirani, V.V.: Design is as easy as optimization. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 477–488. Springer, Heidelberg (2006)
8. Cremonini, M.: Evaluating information security investments from attackers perspective: the return-on-attack (ROA). In: Fourth Workshop on the Economics of Information Security (2005)
9. Fisher, D.: Multi-process architecture (July 2008), <http://dev.chromium.org/developers/design-documents/multi-process-architecture>
10. Franklin, J., Paxson, V., Perrig, A., Savage, S.: An inquiry into the nature and causes of the wealth of internet miscreants. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, pp. 375–388. ACM, New York (2007)
11. Freund, Y., Schapire, R.: A short introduction to boosting. *Journal of the Japanese Society for Artificial Intelligence* 14(5), 771–780 (1999)

12. Freund, Y., Schapire, R.E.: Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29, 79–103 (1999)
13. Friedberg, J.: Internet fraud battlefield (April 2007), http://www.ftc.gov/bcp/workshops/proofpositive/Battlefield_Overview.pdf
14. Fultz, N., Grossklags, J. (eds.): Blue versus Red: Towards a model of distributed security attacks. Proceedings of the Thirteenth International Conference Financial Cryptography and Data Security (February 2009)
15. Gordon, L.A., Loeb, M.P.: The economics of information security investment. *ACM Transactions on Information and System Security* 5(4), 438–457 (2002)
16. Grossklags, J., Christin, N., Chuang, J.: Secure or insure?: A game-theoretic analysis of information security games. In: Proceeding of the 17th International Conference on World Wide Web, pp. 209–218. ACM, New York (2008)
17. Hausken, K.: Returns to information security investment: The effect of alternative information security breach functions on optimal investment and sensitivity to vulnerability. *Information Systems Frontiers* 8(5), 338–349 (2006)
18. Herbster, M., Warmuth, M.K.: Tracking the best expert. *Machine Learning* 32(2), 151–178 (1998)
19. Howard, M.: Attack surface: Mitigate security risks by minimizing the code you expose to untrusted users. *MSDN Magazine* (November 2004), <http://msdn.microsoft.com/en-us/magazine/cc163882.aspx>
20. Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G.M., Paxson, V., Savage, S.: Spamaalytics: An empirical analysis of spam marketing conversion. In: Proceedings of the 2008 ACM Conference on Computer and Communications Security, pp. 3–14. ACM, New York (2008)
21. Kark, K., Penn, J., Dill, A.: 2008 CISO priorities: The right objectives but the wrong focus. *Le Magazine de la Sécurité Informatique* (April 2009)
22. Kumar, V., Telang, R., Mukhopadhyay, T.: Optimal information security architecture for the enterprise, <http://ssrn.com/abstract=1086690>
23. Lye, K.W., Wing, J.M.: Game strategies in network security. In: Proceedings of the Foundations of Computer Security Workshop, pp. 13–22 (2002)
24. Miura-Ko, R.A., Yolken, B., Mitchell, J., Bambos, N.: Security decision-making among interdependent organizations. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium, pp. 66–80. IEEE Computer Society, Washington (2008)
25. Miura-Ko, R., Bambos, N.: SecureRank: A risk-based vulnerability management scheme for computing infrastructures. In: Proceedings of IEEE International Conference on Communications, pp. 1455–1460 (June 2007)
26. Ordentlich, E., Cover, T.M.: The cost of achieving the best portfolio in hindsight. *Mathematics of Operations Research* 23(4), 960–982 (1998)
27. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 336–345 (2006)
28. Pironti, J.P.: Key elements of an information security program. *Information Systems Control Journal* 1 (2005)
29. Rescorla, E.: Is finding security holes a good idea? *IEEE Security and Privacy* 3(1), 14–19 (2005)
30. Varian, H.: System reliability and free riding (2001)
31. Varian, H.R.: Managing online security risks, June 1. *New York Times* (2000)
32. Warner, B.: Home PCs rented out in sabotage-for-hire racket. *Reuters* (July 2004)

Embedded SFE: Offloading Server and Network Using Hardware Tokens

Kimmo Järvinen¹, Vladimir Kolesnikov²,
Ahmad-Reza Sadeghi³, and Thomas Schneider³

¹ Dep. of Information and Comp. Science, Aalto University, Finland
`kimmo.jarvinen@tkk.fi`*

² Alcatel-Lucent Bell Laboratories, Murray Hill, NJ 07974, USA
`kolesnikov@research.bell-labs.com`

³ Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
`{ahmad.sadeghi, thomas.schneider}@trust.rub.de`**

Abstract. We consider Secure Function Evaluation (SFE) in the client-server setting where the server issues a secure token to the client. The token is not trusted by the client and is *not* a trusted third party.

We show how to take advantage of the token to drastically reduce the communication complexity of SFE and computation load of the server.

Our main contribution is the detailed consideration of design decisions, optimizations, and trade-offs, associated with the setting and its strict hardware requirements for practical deployment. In particular, we model the token as a computationally weak device with small constant-size memory and limit communication between client and server.

We consider semi-honest, covert, and malicious adversaries. We show the feasibility of our protocols based on a FPGA implementation.

1 Introduction

Secure and efficient evaluation of arbitrary functions on private inputs has been subject of cryptographic research for decades. In particular, the following scenario appears in a variety of practical applications: a service provider (server \mathcal{S}) and user (client \mathcal{C}) wish to compute a function f on their respective private data, without incurring the expense of a trusted third party. This can be solved interactively using Secure Function Evaluation (SFE) protocols, for example using the very efficient garbled circuit (GC) approach [26]. However, GC protocols potentially require a large amount of data to be transferred between \mathcal{S} and \mathcal{C} . This is because f needs to be encrypted (garbled) as \tilde{f} and transferred from \mathcal{S} to \mathcal{C} . In fact, the communication complexity of GC-based SFE protocols is dominated by the size of the GC, which can reach Megabytes or Gigabytes even for relatively small and simple functions (e.g., the GC for AES has size 0.5 MBytes [23]). Further, if security against more powerful adversaries is required, the use

* Supported by EU FP7 project CACE.

** Supported by EU FP6 project SPEED, EU FP7 project CACE and ECRYPT II.

of the standard cut-and-choose technique implies transfer of multiple GCs. (For covert adversaries, the transfer of only one GC is sufficient [7].)

While transmission of this large amount of data is possible for exceptional occurrences, in most cases, the network will not be able to sustain the resulting traffic. This holds especially for larger-scale deployment of secure computations, e.g., by banks or service providers, with a large number of customers. Additional obstacles include energy consumption required to transmit/receive the data, and the resulting reduced battery life in mobile clients, such as smartphones.¹

Further, computational load on \mathcal{S} (computing f) is also a significant problem, especially in the case of large-scale deployment of SFE.

Our setting, goals and approach. Motivated by the possibility of large-scale and decentralized SFE deployment we aim to remove the expensive communication requirement, and to shift some of \mathcal{S} 's computation to \mathcal{C} . To this end, we note that in SFE protocols, and, in particular, in GC, the role of the server can be split between two entities, with the introduction of a new entity – secure token \mathcal{T} , which is placed in client \mathcal{C} 's possession, but executes \mathcal{S} 's code thus offloading \mathcal{S} . Further, it is possible to eliminate most of the communication between \mathcal{C} and \mathcal{S} and replace this with local communication between \mathcal{C} and \mathcal{T} . A number of technical issues arises in this setting, which we address in this work.

More specifically, we discuss and analyze hardware-supported SFE, where the service provider \mathcal{S} issues a *secure* (see below) hardware token \mathcal{T} to \mathcal{C} . \mathcal{C} communicates locally with \mathcal{T} , and remotely with \mathcal{S} . There is no direct channel between \mathcal{T} and \mathcal{S} , but of course \mathcal{C} can pass (and potentially interfere with) messages between \mathcal{T} and \mathcal{S} . \mathcal{T} is created by \mathcal{S} , so \mathcal{S} trusts \mathcal{T} ; however, as \mathcal{C} does not trust \mathcal{S} , she also does not trust the token \mathcal{T} to behave honestly.²

Attack model. We consider all three standard types of adversaries: semi-honest (follows protocol but analyzes the transcript), malicious (arbitrary behavior, cheating is always caught), and covert (cheating is caught with a certain deterrence probability, e.g., 1/2).

Hardware assumption. We assume \mathcal{T} is tamper-proof or tamper-resistant. We argue that this assumption is reasonable. Indeed, while every token can likely be broken into, given sufficient resources, we are motivated by the scenarios where the payoff of the break is far below the cost of the break. This holds for relatively low-value transactions such as cell phone or TV service, where the potential benefit of the attack (e.g., free TV for one user) is not worth the investment of thousands or tens of thousands of dollars to break into the card. For higher-value applications one could raise the cost of the attack by using a high-end token \mathcal{T} , e.g., a smart card certified at FIPS 140-2, level 3 or 4.

¹ In some cases, the impact can be mitigated by creating and transferring GCs in the precomputation phase. However, this is not fully satisfactory. Firstly, even more data needs to be transferred since demand cannot be perfectly predicted. Further, this creates other problems, such as requiring large long-term storage on client devices.

² Note, if \mathcal{C} in fact trusts \mathcal{T} to behave honestly, then there exists a trivial solution, where \mathcal{C} would let \mathcal{T} compute the function on her inputs [11].

Hardware restrictions. As we assume the token to be produced in large quantities, we try to minimize its costs (e.g., chip surface) and make the assumptions on it as weak as possible. In particular our token requires only restricted computational capabilities (no public-key operations) and small constant secure RAM. We consider \mathcal{T} with and without small constant secure non-volatile storage.

Envisioned Applications. As mentioned above, we aim to bring SFE closer to a large-scale deployment. The need to minimize communication forces us to rely on tokens, the issuance of which requires certain logistical capabilities. Therefore, we believe client-server applications are most likely to be the early adopters of SFE. Further, the natural trust model (semi-honest or covert server and malicious client) allow for efficient GC protocols. Finally, many client-server application scenarios naturally include financial and other transactions which involve sensitive, in particular privacy-sensitive, information.

Today, many service providers already distribute trusted tokens to their users. Examples include SIM cards in users' cell phones, and smart cards in users' TV set-top boxes. Bank- and credit cards often contain embedded secure chips. Special purpose (e.g., diagnostic) medical devices, which need to be monitored and controlled, are often issued to users at home. In these settings, it is natural to use the existing infrastructure to enhance the security and privacy guarantees of the offered products, and to offer new products previously impossible due to privacy violation concerns. We expand this discussion and consider other applications, such as privacy protection in targeted advertisement, content delivery, and remote medical diagnostics, in the full version [13].

1.1 Our Contributions and Outline

Our main contribution is architecture design, implementation, a number of optimizations, and detailed analysis of two-party SFE aided by a server-issued low-cost tamper-proof token. The communication complexity of our protocols is linear in the size of the input, and is independent of the size of the evaluated functionality. Further, most of the work of \mathcal{S} can be offloaded to \mathcal{T} .

We use GC techniques of [15] and offer no-cost XOR gates. We rely on cheap hardware – the token \mathcal{T} only executes symmetric-key operations (e.g., SHA and AES). \mathcal{T} has small constant-size RAM (much smaller than the size of the circuit), but we do not resort to implementing expensive secure external RAM.

We provide two solutions; in one, \mathcal{T} keeps state in secure non-volatile storage (a monotonic counter), while in the other, \mathcal{T} maintains no long-term state.

We consider semi-honest, covert [7], and malicious [16] adversaries; our corresponding communication improvements are shown in Table 1.

Outline. We start with outlining our model and architecture in §3. We describe the protocols for both stateful and stateless \mathcal{T} , and state the security claim in §4. In §5, we discuss technical details of our FPGA prototype implementation, present timings and measurements, and show practicality of our solution.

Table 1. Communication between server \mathcal{S} and client \mathcal{C} for secure evaluation of function f with n inputs, statistical security parameter s , and deterrence probability $1 - 1/r$

Security	Previous Work	This Work
semi-honest	[26] $\mathcal{O}(f + n)$	$\mathcal{O}(n)$
covert	[7] $\mathcal{O}(f + sn + r)$	$\mathcal{O}(sn + r)$
malicious	[16] $\mathcal{O}(s f + s^2n)$	$\mathcal{O}(s^2n)$

1.2 Related Work

Related work on using tokens for secure computations falls in the following three categories, summarized in Table 2.

Table 2. Secure Protocols using Hardware Tokens. Columns denote the number of tokens, who trusts the token(s), if token(s) are stateful or stateless, and perform public-key operations. Properties more desired for practical applications in bold font.

Type	Reference	Functionality	# Tokens	Trusted by	Stateful	PK ops
A)	[10]	UC commitment	2	both	yes	yes
	[14,4]	UC commitment	2	issuer	yes	yes
	[3]	UC commitment	2	issuer	no	yes
	[19]	UC commitment	1	issuer	yes	no
B)	[9]	Set Intersection, ODBS	1	both	yes	no
	[8]	Non-Interact. OT	1	both	yes	yes
	[25]	Verif. Enc., Fair Exch.	1	both	yes	yes
C)	[6]	SFE	2	both	yes	yes
	[11]	SFE	1	both	yes	yes
	This Work	SFE	1	issuer	yes / no	no

A) *Setup assumptions for the universal composability (UC) framework.* As shown in [1], UC SFE protocols can be constructed from UC commitments. In turn, UC commitments can be constructed from signature cards trusted by both parties [10], or from tamper-proof tokens created and trusted only by the issuing party [14,19,3,4]. Here, [3] consider stateless tokens, and [19] require only one party to issue a token. This line of research mainly addresses the feasibility of UC computation based on tamper-proof hardware and relies on expensive primitives such as generic zero-knowledge proofs. Our protocols are far more practical.

B) *Efficiency Improvements for Specific Functionalities.* Efficient protocols with a tamper-proof token trusted by both players have been proposed for specific functionalities such as set intersection and oblivious database search (ODBS) [9], non-interactive oblivious transfer (OT) [8], and verifiable encryption and fair exchange [25]. In contrast, we solve the general SFE problem.

C) *Efficiency Improvements for Arbitrary Functionalities.* Clearly, SFE is efficient if aided by a trusted third party (TTP), who simply computes the function. SFE aided by hardware TTP was considered, e.g., in [6,11]. In contrast, we do not use TTP; our token is only trusted by its issuer.

2 Preliminaries

Notation. We denote symmetric security parameter by t (e.g., $t = 128$), and pseudo-random function (PRF) keyed with k and evaluated on x by $\text{PRF}_k(x)$. PRF can be instantiated with a block cipher, e.g., AES, or a cryptographic hash function H , e.g., SHA-256, which we model as a Random Oracle (RO). AES is preferable if PRF is run repeatedly with same k as AES’s key schedule amortizes. Message authentication code (MAC) keyed with k and evaluated on message m is denoted by $\text{MAC}_k(m)$. We use a MAC that does not need to store the entire message, but can operate “online” on small blocks, e.g., AES-CMAC [24].

2.1 Garbled Circuits (GC)

Yao’s Garbled Circuit approach [26] is the most efficient method for secure evaluation of a boolean circuit C . We summarize its ideas in the following. The circuit *constructor* (server \mathcal{S}) creates a *garbled circuit* \tilde{C} : for each wire w_i of the circuit, he randomly chooses two garblings $\tilde{w}_i^0, \tilde{w}_i^1$, where \tilde{w}_i^j is the *garbled value* of w_i ’s value j . (Note: \tilde{w}_i^j does not reveal j .) Further, for each gate G_i , \mathcal{S} creates a *garbled table* \tilde{T}_i with the following property: given a set of garbled values of G_i ’s inputs, \tilde{T}_i allows to recover the garbled value of the corresponding G_i ’s output, but nothing else. \mathcal{S} sends these garbled tables, called *garbled circuit* \tilde{C} to the *evaluator* (client \mathcal{C}). Additionally, \mathcal{C} obliviously obtains the *garbled inputs* \tilde{w}_i corresponding to inputs of both parties: the garbled inputs \tilde{y} corresponding to the inputs y of \mathcal{S} are sent directly and \tilde{x} are obtained with a parallel 1-out-of-2 oblivious transfer (OT) protocol [20]. Now, \mathcal{C} can evaluate the garbled circuit \tilde{C} on the garbled inputs to obtain the *garbled outputs* by evaluating \tilde{C} gate by gate, using the garbled tables \tilde{T}_i . Finally, \mathcal{C} determines the plain values corresponding to the obtained garbled output values using an output translation table received by \mathcal{S} . Correctness of GC follows from the way garbled tables \tilde{T}_i are constructed.

Improved Garbled Circuit with free XOR [15]. An efficient method for creating garbled circuits which allows “free” evaluation of XOR gates was presented in [15]. More specifically, a garbled XOR gate has no garbled table (*no communication*) and its evaluation consists of XORing the its garbled input values (*negligible computation*) – details below. The other gates, called *non-XOR gates*, are evaluated as in Yao’s GC construction [26] with a *point-and-permute technique* (as used in [18]): The garbled values $\tilde{w}_i = \langle k_i, \pi_i \rangle \in \{0, 1\}^t$ consist of a symmetric key $k_i \in \{0, 1\}^t$ and a random permutation bit $\pi_i \in \{0, 1\}$ (recall, t is the symmetric security parameter). The entries of the garbled table are permuted such that the permutation bits π_i of a gate’s garbled input wires can be used as index into the garbled table to directly point to the entry to be decrypted. After decrypting this entry using the garbled input wires’ t -bit keys k_i , evaluator obtains the garbled output value of the gate. The encryption is done with the symmetric encryption function $\text{Enc}_{k_1, \dots, k_d}^s(m)$, where d is the number of inputs of the gate and s is a unique identifier used once. Enc can be instantiated with $m \oplus H(k_1 || \dots || k_d || s)$, where H is a RO. We note that the RO assumption can be

avoided or weakened at small additional computation cost – see full version [13]. The main observation of [15] is, that the constructor \mathcal{S} chooses a global key difference $\Delta \in_R \{0, 1\}^t$ which remains unknown to evaluator \mathcal{C} and relates the garbled values as $k_i^0 = k_i^1 \oplus \Delta$. Clearly, the usage of such garbled values allows for *free evaluation of XOR gates* with input wires w_1, w_2 and output wire w_3 by computing $\tilde{w}_3 = \tilde{w}_1 \oplus \tilde{w}_2$ (no communication and negligible computation).

3 Architecture, System and Trust Model

We present in detail our setting, players, and hardware and trust assumptions.

As shown in Fig. 1, there are three parties – client \mathcal{C} , server \mathcal{S} and tamper-resistant token \mathcal{T} , issued and trusted by \mathcal{S} . Our goal is to let \mathcal{C} and \mathcal{S} securely evaluate a public function f on their respective private inputs x and y .

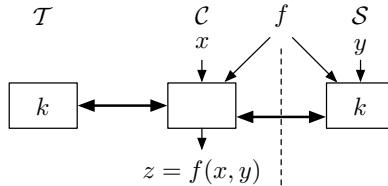


Fig. 1. Model Overview

Communication. $\mathcal{C} \leftrightarrow \mathcal{S}$: We view this as an expensive channel. Communication $\mathcal{C} \leftrightarrow \mathcal{S}$ flows over the Internet, and may include a wireless or cellular link. This implies small link bandwidth and power consumption concerns of mobile devices. We wish to minimize the utilization of this channel.

$\mathcal{T} \leftrightarrow \mathcal{C}$: As \mathcal{T} is held locally by \mathcal{C} , this is a cheap channel (both in terms of bandwidth and power consumption), suitable for transmission of data linear in the size of f , or even greater.

$\mathcal{T} \leftrightarrow \mathcal{S}$: There is no direct channel between \mathcal{T} and \mathcal{S} , but, of course, \mathcal{C} can pass (and potentially interfere with) messages between \mathcal{T} and \mathcal{S} .

Trust. $\mathcal{C} \leftrightarrow \mathcal{S}$: As in the standard SFE scenario, \mathcal{C} and \mathcal{S} don't trust each other. We address semi-honest, covert, and malicious \mathcal{C} and \mathcal{S} .

$\mathcal{S} \leftrightarrow \mathcal{T}$: \mathcal{T} is fully trusted by \mathcal{S} , since \mathcal{T} is tamper-resistant. \mathcal{S} and \mathcal{T} share a secret key k , used to establish a secure channel and to derive joint randomness.³

$\mathcal{T} \leftrightarrow \mathcal{C}$: \mathcal{C} does not trust \mathcal{T} , as \mathcal{T} is the agent of \mathcal{S} , and may communicate with \mathcal{S} through covert channels.

Storage, computation and execution. \mathcal{C} and \mathcal{S} are computationally strong devices which can perform both symmetric- and asymmetric-key operations.³ Both have sufficient memory, linear in the size of f . \mathcal{C} has control over \mathcal{T} , and can reset it, e.g., by interrupting its power supply. As justified in §1, \mathcal{T} is a cheap

³ If needed, \mathcal{C} 's capabilities may be enhanced by using a trusted hardware accelerator.

special purpose hardware with minimum chip surface: \mathcal{T} has circuitry only for evaluating symmetric-key primitives in hardware (no public-key or true random number generator) and has a small secure RAM. It may (§4.3) or may not (§4.4) have small non-volatile secure storage⁴, unaffected by the resets by \mathcal{C} .

4 Token-Assisted Garbled Circuit Protocols

In our presentation, we assume reader's familiarity with the GC technique, including free XORs of [15] (cf. §2.1), and concentrate on the aspects specific to the token setting. We start with a high-level description of our protocol. Then, in §4.2 - §4.4, we present the technical details of our construction – efficient circuit representation, and GC generation by stateful and stateless tokens.

4.1 Protocols Overview, Security Intuition and Claim

Our constructions are a natural (but technically involved) modification of standard GC protocols, so as to split the actions of the server into two parts – now executed by \mathcal{S} and \mathcal{T} – while maintaining provable security. We offload most of the work (notably, GC generation and output) to \mathcal{T} , thus achieving important communication savings, and partially offloading \mathcal{S} 's computation to \mathcal{T} .

We start our discussion with the solution in the semi-honest model. However, our modification of the basic GC is secure against malicious actions, and our protocols are easily and efficiently extendible to covert and malicious settings.

At the high level, our protocols work as shown in Fig. 2: \mathcal{C} obtains the garbled inputs \tilde{x}, \tilde{y} from \mathcal{S} , and the garbled circuit \tilde{f} corresponding to the function f from \mathcal{T} . Then, \mathcal{C} evaluates \tilde{f} on \tilde{x}, \tilde{y} and obtains the result $z = f(x, y)$.

It is easy to see that the introduction of \mathcal{T} and offloading to it some of the computation does not strengthen \mathcal{S} , and thus does not bring security concerns for \mathcal{C} (as compared to standard two-party GC). On the other hand, separating

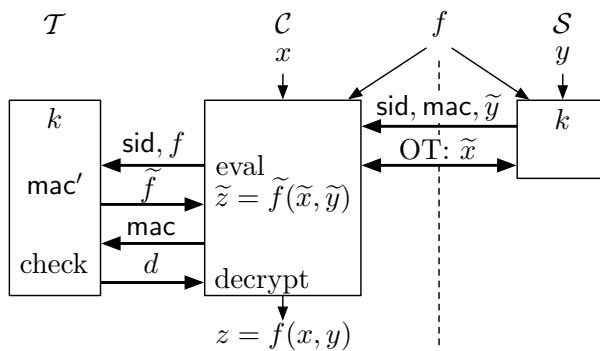


Fig. 2. Protocols Overview

⁴ \mathcal{T} 's key k is a fixed part of its circuit, and is kept even without non-volatile storage.

the states of \mathcal{S} and \mathcal{T} , placing \mathcal{C} in control of their communication, and \mathcal{C} 's ability to reset \mathcal{T} introduces attack opportunities for \mathcal{C} . We show how to address these issues with the proper synchronization and checks performed by \mathcal{S} and \mathcal{T} .

Our main tool is the use of a unique session id sid for each GC evaluation. From sid and the shared secret key k , \mathcal{S} and \mathcal{T} securely derive a session key K , which is then used to derive the randomness used in GC generation. Jumping ahead (details in §4.3), we note that sid uniqueness is easily achieved if \mathcal{T} is stateful simply by setting sid equal to the value of the strictly monotonic session counter ctr maintained by \mathcal{T} . However, if \mathcal{T} is stateless, \mathcal{C} can always replay \mathcal{S} 's messages. In §4.4 we show how to ensure that replays do not help \mathcal{C} .

Since \mathcal{S} and \mathcal{T} derive the same randomness for each session, the (same) garbled circuit \tilde{f} can be generated by \mathcal{T} . Unfortunately, the weak \mathcal{T} cannot store the entire f . Instead, \mathcal{C} provides the circuit corresponding to function f gate-by-gate to \mathcal{T} , and obtains the corresponding garbled gate of \tilde{f} . The garbled gate can immediately be evaluated by \mathcal{C} and needs not to be stored. \mathcal{C} is prevented from providing a wrong f to \mathcal{T} , as follows. First, \mathcal{S} issues a MAC of f , e.g., $\text{mac} = \text{MAC}_k(\text{sid}, f)$, where f is the agreed circuit representation of the evaluated function (cf. §4.2). Further, \mathcal{T} computes its version of the above MAC, mac' , as it answers \mathcal{C} 's queries in computing \tilde{f} . Finally, \mathcal{T} reveals the decryption information d that allows \mathcal{C} to decrypt the output wires only if \mathcal{C} provides the matching mac .

Garbled Inputs. The garbled input \tilde{y} of \mathcal{S} can be computed by \mathcal{S} and sent to \mathcal{C} , requiring $|y| \cdot t$ bits communication, where t is the security parameter. Alternatively, if \mathcal{T} is stateful, \mathcal{S} can establish a secure channel with \mathcal{T} , e.g., based on session key K , send y over the channel, and have \mathcal{T} output \tilde{y} to \mathcal{C} . This achieves the optimal communication between \mathcal{S} and \mathcal{C} of $|y|$ bits.

The garbling \tilde{x} of \mathcal{C} 's input can be transferred from \mathcal{S} to \mathcal{C} with a parallel OT protocol which requires $\mathcal{O}(|x|t)$ bits of communication. Alternatively, the efficient OT extension of [12] which reduces many OTs to a small number of OTs (depending on security parameter t) can be adopted to our token-based scenario as described in the full version [13, §C]. This reduces the communication between \mathcal{S} and \mathcal{C} to $\mathcal{O}(t^2)$ which is independent of the size of the input x .

Extension to Covert and Malicious Parties. Standard GC protocols for covert [7] or malicious [16] adversaries rely on the following cut-and-choose technique. \mathcal{S} creates multiple GCs \tilde{C}_i , deterministically derived from random seeds s_i , and commits to each, e.g., by sending \tilde{C}_i or $\mathsf{H}(\tilde{C}_i)$ to \mathcal{C} . In covert case, \mathcal{C} asks \mathcal{S} to open all but one garbled circuit I by revealing the corresponding $s_{i \neq I}$. For all opened circuits, \mathcal{C} computes \tilde{C}_i and checks that they match the commitments. The malicious case is similar, but \mathcal{C} asks \mathcal{S} to open half of the circuits, evaluates the remaining ones and chooses the majority of their results.

These protocols similarly benefit from our token-based separation of the server into \mathcal{S} and \mathcal{T} . As in the semi-honest protocol, the GC generation can be naturally offloaded to \mathcal{T} , achieving corresponding computation and communication relief on the server and network resources. GC correctness verification is achieved by requesting \mathcal{S} to reveal the generator seeds $s_{i \neq I}$. (Of course, these

“opened” circuits are not evaluated.) Note that requirements on \mathcal{T} are the same as in the semi-honest setting. Further, in both covert and malicious cases, the communication between \mathcal{C} and \mathcal{S} is independent of the size of f . The resulting communication complexity of these protocols is summarized in Table 1.

Security Claim. For the lack of space, in this work we present our protocols implicitly, by describing the modifications to the base protocols of [15]. We informally argue the security of the modifications as they are described. Formal proofs can be naturally built from proofs of [15] and our security arguments. At the very high level, security against \mathcal{S}/\mathcal{T} follows from the underlying GC protocols, since \mathcal{S} is not stronger here than in the two-party SFE setting. The additional power of \mathcal{C} to control the channel between \mathcal{S} and stateful \mathcal{T} is negated by establishing a secure channel (§4.3). \mathcal{C} ’s power to reset stateless \mathcal{T} is addressed by ensuring that by replaying old messages \mathcal{C} gets either what he already knows, or completely unrelated data (§4.4).

Theorem 1. *Assuming \mathcal{T} is tamper-proof, protocols described throughout this section are secure in the semi-honest, covert, and malicious models respectively.*

4.2 Circuit Representation

We now describe our circuit representation format. Our criteria are compactness, the ability to accommodate free XOR gates of [15], and ability of \mathcal{T} to process the encoding “online”, i.e., with small constant memory. Recall, our \mathcal{T} operates in request-response fashion. \mathcal{C} incrementally, gate-by-gate, “feeds” the circuit description to \mathcal{T} which responds with the corresponding garbled tables.

We consider circuits with two-input boolean gates. We note that our techniques can be naturally generalized to general circuits.

Our format is derived from standard representations, such as that of Fairplay [18], with the necessary changes to support our requirements. For readability, we describe the format using a simple example circuit shown in Fig. 3. This circuit computes $z_1 = x_1 \wedge (y_1 \oplus y_2)$, where x_1 is the input bit of \mathcal{C} and y_1, y_2 are two input bits of \mathcal{S} . The corresponding circuit representation shown on the right is composed from the description of the inputs, gates, and outputs as follows.

Inputs and wires: The wires w_i of the circuit are labeled with their index $i = \{0, 1, \dots\}$ (wires with a larger fan-out are viewed as a single wire). The first X wires are associated with the input of \mathcal{C} , the following Y wires are associated with the input of \mathcal{S} , and the internal wires are labeled in topological order starting

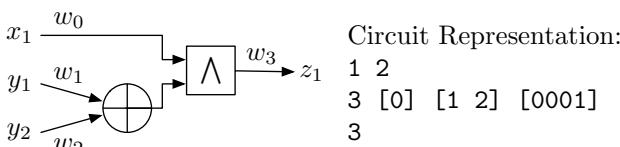


Fig. 3. Example for Circuit Representation

from index $X + Y$ (output wires of XOR gates are not labeled, as XOR gates are incorporated into their successor gates as described in the next paragraph). The first line of the circuit description specifies X and Y (Fig. 3: $X = 1, Y = 2$).

Gates are labeled with the index of their outgoing wire; each gate description specifies its input wires. XOR gates do not have gate tables and are omitted from the description. Rather, non-XOR gates, instead of pointing to two input wires, include two input wire *lists*. If the input list contains more than one wire, these wire values are to be XORed to obtain the corresponding gate input. Gate's description concludes with its truth table. In Fig. 3, the second line describes the AND gate, which has index 3, and inputs w_0 and $w_1 \oplus w_2$.

Outputs: The circuit description concludes with Z lines which contain the indices of the Z output wires (Fig. 3: the only ($Z = 1$) output wire is w_3).

Large XOR sub-circuits. In this representation, XOR gates with fan-out > 1 occur multiple times in the description of their successor gates. In the worst case, this results in a quadratic increase of the circuit description. To avoid this cost, we insert an identity gate after each XOR gate with a large fan-out.

4.3 GC Creation with Stateful Token Using Secure Counter

The main idea of our small-RAM-footprint GC generation is having \mathcal{T} generate garbled tables “on the fly”. This is possible, since each garbled table can be generated only given the garblings of input and output wires. In our implementation, we pseudorandomly derive the wire garbling from the session key and wire index. The rest of this section contains relevant details.

Session Initialization. SFE proceeds in sessions, where one session is used to securely evaluate a function once. \mathcal{T} has a secure monotonic session counter ctr which is (irreversibly) incremented at the beginning of each session. The session id sid is set to the incremented state of ctr . (We omit the discussion of synchronization of ctr between \mathcal{T} and \mathcal{S} which may happen due to communication and other errors.) Then, the session key is computed by \mathcal{S} and \mathcal{T} as $K = \text{PRF}_k(\text{sid})$ and subsequently used to provide fresh randomness to create the GC.

As required by the construction of [15] (cf. §2.1), the two garbled values of the same wire differ by a global difference offset Δ . This offset is derived from K at session initialization and kept in RAM throughout the session.

Subsequently, garbled wire values w_i are derived on-the-fly from K as

$$\tilde{w}_i^0 = \text{PRF}_K(i), \quad \tilde{w}_i^1 = \tilde{w}_i^0 \oplus \Delta. \quad (1)$$

Garbled Gates. \mathcal{T} receives the description of the circuit, line by line, in the format described in §4.2, and generates and outputs to \mathcal{C} corresponding garbled gates, using only small constant memory. \mathcal{T} first verifies that the gate with the same label had not been processed before. (Otherwise, by submitting multiple gate tables for the same gate, \mathcal{C} may learn the real wire values). This is achieved by keeping the monotonically increasing processed gate counter gctr , verifying that gate's label $\text{glabel} > \text{gctr}$, and setting $\text{gctr} = \text{glabel}$. \mathcal{T} then derives and stores garblings of the gate's input and output wires according to (1). (For input lists, the wire's garbling \tilde{w}^0 is computed as the XOR of garblings of the listed wires,

and \tilde{w}^1 is set to $\tilde{w}^0 \oplus \Delta$. Note that this requires constant RAM.) Finally, based on these garblings, gate's garbled table is computed and output to \mathcal{C} .

Garbled Outputs. Recall, \mathcal{T} must verify circuit correctness by checking `mac` generated by \mathcal{S} . Thus, \mathcal{T} does not release the output decryption tables to \mathcal{C} until after the successful check. At the same time, the check is not complete until the entire circuit had been fed to \mathcal{T} . To avoid having \mathcal{T} store the output decryption tables or involving \mathcal{S} at this stage, \mathcal{T} simply encrypts the output tables using a fresh key K' , and outputs the key only upon a successful MAC verification.

4.4 GC Creation with Stateless Token (no Counter)

As discussed above, while non-volatile secure storage (the counter `ctr`) is essential in our protocol of §4.3, in some cases, it may be desired to avoid its cost. We now discuss the protocol amendments required to maintain security of SFE with the support of a token whose state can be reset by, e.g., a power interruption.

First, we observe that \mathcal{S} is still able to maintain state, and choose unique counters. However, \mathcal{T} can no longer be assured that `sid` claimed by \mathcal{C} is indeed fresh. Further, \mathcal{T} does not have a source of independent randomness, and thus cannot establish a secure channel with \mathcal{S} , e.g., by running a key exchange.

We begin with briefly describing a replay vulnerability of our protocol of §4.3, when \mathcal{T} is executed with same `sid`. First, \mathcal{C} properly executes SFE. Second time he runs \mathcal{T} with the same `sid`, but feeds \mathcal{T} an incorrect circuit, receiving valid garbled tables for each of the gates, generated for the *same* wire garblings. Now, even though \mathcal{T} will not accept `mac` and will not decrypt the output wires, \mathcal{C} had already received them in the first execution. It is easy to see that \mathcal{C} “wins”.

Our solution is to ensure that \mathcal{C} does not benefit from replaying \mathcal{T} with the same `sid`. To achieve this, we require that each wire garblings are derived from the (hash of the) entire gate description (i.e., id, truth table, and list of inputs), as described below. If \mathcal{C} replays and gives a different gate description, she will not be able to relate the produced garbled table with a previous output of \mathcal{T} .

We associate with each wire w_i a (revealed to \mathcal{C}) hash value h_i . For input wires, h_i is the empty string. For each other wire i , h_i is derived (e.g., via Random Oracle) from the description of the gate i (which includes index, truth table, and list of inputs; cf. §4.2) that emits that wire: $h_i = H(\langle \text{gate_description} \rangle)$. The garbled value of wire w_i now depends on its hash value h_i : $\tilde{w}_i^0 = \text{PRF}_K(h_i)$ and $\tilde{w}_i^1 = \tilde{w}_i^0 \oplus \Delta$. Finally, to enable the computation of the garbled tables, \mathcal{C} must feed back to \mathcal{T} the hashes h_i of the input wires, and receive from \mathcal{T} and keep for future use the hash of the output wire. As noted above, \mathcal{C} 's attempts to feed incorrect values result in the output of garbled tables that are unrelated to previous outputs of \mathcal{T} , and thus do not help \mathcal{C} .

5 Proof-of-Concept Implementation

We have designed a proof-of-concept implementation to show the practicability of our token-assisted GC protocols of §4. In the following we describe our

architecture for the stateful token case of §4.3. Extension to the stateless case is straightforward. We instantiate PRF with AES-128, H and \mathbf{H} with SHA-256, and MAC with AES-CMAC.

5.1 Architecture

Fig. 4 depicts the high level architecture of our design consisting of a two-stage pipeline and a MAC core. Stage 1 of the pipeline creates the garbled input and output values of a gate using an AES core and stage 2 computes the garbled table with a SHA core. The two-stage pipeline increases performance as two gates can be processed concurrently. The MAC core computes the authentication message mac' of the circuit provided by \mathcal{C} (cf. §4.1).

Design Principle. To achieve maximum speed with minimum hardware resources, we followed a general guideline exploiting parallelism as long as it can be done without using several instances of the same algorithm. For example, we opted to compute the four entries of a garbled table with a single SHA core instead of using four parallel SHA cores which would have increased performance, but only with a significant increase in area. As the only exception, we included a separate

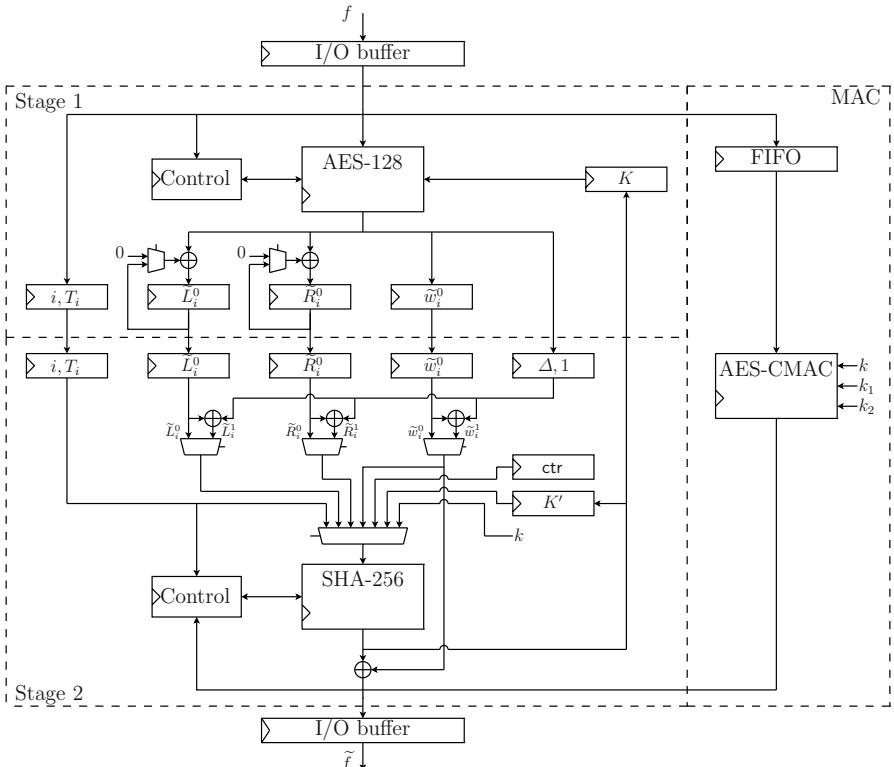


Fig. 4. Simplified architectural diagram of our proof-of-concept implementation. Selections of multiplexers and write enables of registers are set by additional control logics.

MAC core rather than reusing the AES core of stage 1, because it would have severely complicated the control of the pipeline.

Description of Operation. In the session initialization (cf. §4.4), the SHA core in stage 2 derives session key K and output encryption key K' from key k and current counter value $\text{ctr} = \text{sid}$ which is used as key for the AES core. Then, the key difference Δ is derived with the AES core and stored in a register. The circuit is provided gate-by-gate into the input buffer in the format described in §4.2 (gate table denoted by T_i in Fig. 4). Stage 1 starts to process a gate by deriving the garbled output value \tilde{w}_i^0 . Then, the two garbled inputs of the gate ($\tilde{L}_i^0, \tilde{R}_i^1$ in Fig. 4) are derived by XORing the garblings listed in the input wire lists one-by-one (see §4.3). When all garblings are derived they are forwarded to stage 2 and stage 1 processes the next gate. Stage 2 computes the garbled table and the encrypted output decryption tables and writes them into the output buffer. The MAC core operates independently from the two-stage pipeline.

5.2 Prototype Implementation

We implemented our architecture in VHDL.

Implementation Details. For the AES core we chose an iterative design of AES-128 with a latency of 10 clock cycles per encryption. The core includes an online key scheduling unit. The S-boxes are implemented as suggested in [2]; otherwise, the core is a straightforward implementation of the standard [21]. The SHA core implements SHA-256 with a latency of 67 clock cycles per 512-bit block. The core is a straightforward iterative design of the standard [22]. The MAC core includes an AES core implemented as above; otherwise the core is a straightforward implementation of AES-CMAC [24]. As the subkeys, k_1 and k_2 , depend only on the key k they were precomputed and hardwired in the design.

FPGAs. We compiled the VHDL code for a low-end FPGA, the Altera Cyclone II EP2C20F484C7 FPGA, with Quartus II, version 8.1 (2008). We emphasize that this FPGA is for prototyping only, as it lacks secure embedded non-volatile memory for storing ctr (e.g., the Xilinx Spartan-3AN FPGAs has integrated Flash memory for this). The resulting area and memory requirements are listed in Table 3. The design occupies 60% of logic cells and 21% of memory blocks available on the device and runs at 66 MHz (the critical path for clock frequency is in the AES core). These results show that the design is, indeed, feasible for a low-cost implementation, for example, with low-cost FPGAs which, in turn, is mandatory for the practicability of the token-based scheme.

Smart Cards. In particular, we note that the requirements are sufficiently low also for contemporary smart card technologies, because AES-128 and SHA-256 require only about 3,400 and 11,000 gates, respectively [5]. As our protocol requires no public-key operations on the token, small smart cards are sufficient.

Performance. We determined the latency of our implementation with ModelSim, version 6.3g (2008). Overall, the latency is given as $\#clock_cycles = 158G_1 + 312G_2 + 154O + 150$, where G_1, G_2 is the number of 1-input gates respectively

Table 3. Results on an Altera Cyclone II FPGA (the hierarchy is as shown in Fig. 4)

Entity	Stage 1	Stage 2	MAC	IO	Total
Area (Logic cells)	3317 (30 %)	4539 (40 %)	3059 (27 %)	263 (3 %)	11231 (60 %)
Memory (M4K)	0 (0 %)	8 (73 %)	1 (9 %)	2 (18 %)	11 (21 %)

2-input gates and O is the number of outputs, assuming that each gate has at most 21 inputs in its input lists (if more, stage 2 needs to wait for stage 1) and that data I/O does not introduce additional delays.

Example 1. Our implementation generates a GC for 16-bit comparison ($G_1 = 0$, $G_2 = 16$, $O = 1$) in 5,296 clock cycles ($\approx 80 \mu\text{s}$ with 66 MHz clock). In Software, this takes roughly 0.5 s on an Intel Core 2 6420 at 2.13 GHz [17].

Example 2. Generating a GC for AES-128 encryption ($G_1 = 12614$, $G_2 = 11334$, $O = 128$) takes 5,549,082 clock cycles ($\approx 84 \text{ ms}$ with 66 MHz clock). In Software, this takes approximately 1 s on an Intel Core 2 Duo at 3.0 GHz [23].

Acknowledgements. We would like to thank Wilko Henecka for preparing test circuits, and Ivan Damgård and reviewers of FC'10 for their helpful comments.

References

1. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC 2002, pp. 494–503 (2002)
2. 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)
3. 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)
4. 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)
5. Feldhofer, M., Wolkerstorfer, J.: Strong crypto for RFID tags — a comparison of low-power hardware implementations. In: IEEE Symp. Circuits and Systems (ISCAS 2007), pp. 1839–1842 (2007)
6. 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., Sабelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 34–48. Springer, Heidelberg (2006)
7. Goyal, V., Mohassel, P., Smith, A.: Efficient two party and multi party computation against covert adversaries. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 289–306. Springer, Heidelberg (2008)
8. Gunupudi, V., Tate, S.: Generalized non-interactive oblivious transfer using count-limited objects with applications to secure mobile agents. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 98–112. Springer, Heidelberg (2008)
9. Hazay, C., Lindell, Y.: Constructions of truly practical secure protocols using standard smartcards. In: Proc. ACM CCS, pp. 491–500. ACM, New York (2008)

10. Hofheinz, D., Müller-Quade, J., Unruh, D.: Universally composable zero-knowledge arguments and commitments from signature cards. In: Central European Conference on Cryptology (MoraviaCrypt 2005) (2005)
11. Iliev, A., Smith, S.: More efficient secure function evaluation using tiny trusted third parties. Technical Report TR2005-551, Dartmouth College, Computer Science, Hanover, NH (July 2005)
12. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
13. Järvinen, K., Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Embedded SFE: Offloading server and network using hardware tokens. In: Cryptology ePrint Archive, Report 2009/591 (2009), <http://eprint.iacr.org/>
14. 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)
15. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórssón, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
16. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
17. Lindell, Y., Pinkas, B., Smart, N.: Implementing two-party computation efficiently with security against malicious adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
18. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: USENIX Security Symposium 2004. USENIX Association (2004)
19. 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)
20. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), pp. 448–457. SIAM, Philadelphia (2001)
21. NIST, U.S. National Institute of Standards and Technology. Federal Information Processing Standards (FIPS 197). Advanced Encryption Standard (AES) (November 2001), <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
22. NIST, U.S. National Institute of Standards and Technology. Federal Information Processing Standards (FIPS 180-2). Announcing the Secure Hash Standard (August 2002), <http://csrc.nist.gov/publications/fips/fips180-2/fips-180-2.pdf>
23. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
24. Song, J.H., Poovendran, R., Lee, J., Iwata, T.: The AES-CMAC Algorithm. RFC 4493 (Informational) (June 2006), <http://tools.ietf.org/html/rfc4493>
25. 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)
26. Yao, A.C.: How to generate and exchange secrets. In: IEEE Symposium on Foundations of Computer Science (FOCS 1986), pp. 162–167. IEEE, Los Alamitos (1986)

The Phish-Market Protocol: Securely Sharing Attack Data between Competitors

Tal Moran and Tyler Moore

Center for Research on Computation & Society, Harvard University
`{talm,tmoore}@seas.harvard.edu`

Abstract. A key way in which banks mitigate the effects of phishing is to remove fraudulent websites or suspend abusive domain names. This ‘take-down’ is often subcontracted to specialist firms. Prior work has shown that these take-down companies refuse to share ‘feeds’ of phishing website URLs with each other, and consequently, many phishing websites are not removed because the firm with the take-down contract remains unaware of their existence. The take-down companies are reticent to exchange feeds, fearing that competitors with less comprehensive lists might ‘free-ride’ off their efforts by not investing resources to find new websites, as well as use the feeds to poach clients. In this paper, we propose the Phish-Market protocol, which enables companies with less comprehensive feeds to learn about websites impersonating their own clients that are held by other firms. The protocol is designed so that the contributing firm is compensated only for those websites affecting its competitor’s clients and only those previously unknown to the receiving firm. Crucially, the protocol does not reveal to the contributing firm which URLs are needed by the receiver, as this is viewed as sensitive information by take-down firms. Using complete lists of phishing URLs obtained from two large take-down companies, our elliptic-curve-based implementation added a negligible average 5 second delay to securely share URLs.

1 Introduction

Phishing is the criminal activity of enticing people into visiting websites that impersonate genuine bank¹ websites, and to dupe them into revealing passwords and other credentials to carry out fraudulent activities. One of the key countermeasures to phishing is the prompt removal of the imitation bank websites. Removal may be achieved by erasing the web pages from the hosting machine, or by contacting a registrar to suspend a domain name from the DNS so the fraudulent host can no longer be resolved.

¹ Although a wide range of companies have been subject to phishing attacks, the vast majority are financial institutions; for simplicity, we use the term ‘banks’ for firms being attacked.

Although some banks deal with phishing website removal exclusively ‘in-house’, most hire specialist ‘take-down’ companies to carry out the task. Take-down companies — typically divisions of brand-protection firms or information security service providers — perform two key services for banks. First, they are good at getting phishing websites removed quickly, having developed relationships with ISPs and registrars across the globe and deployed multi-lingual teams at 24x7 operations centers. Second, they collect a more timely and comprehensive listing of phishing URLs than banks normally gather.

Most take-down companies view their URL feeds as a key competitive advantage over banks and other take-down providers. However, recent work has shown that the feeds compiled by take-down companies suffer from large gaps in coverage that significantly prolong the time taken to remove phishing websites. Moore and Clayton examined six months of aggregated URL feeds from many sources, including two major take-down companies [12]. They found that up to 40% of the phishing websites impersonating banks hired by take-down companies were known to others but not by the company with the take-down contract. Another 29% of websites were discovered by the responsible take-down company only after others had identified the sites. By measuring the substantially longer lifetimes of these missed websites, Moore and Clayton estimated that at least \$330 million per year is being put at risk by the failure to share proprietary feeds of URLs for just the two companies they studied.

But is sharing the answer, and, if so, then how should an effective sharing mechanism be designed? Moore and Clayton appealed to the security industry’s sense of responsibility and argued that URL feeds should be shared freely between take-down companies and banks, pointing to the precedent of sharing in the anti-virus industry. However, there are some reasonable objections to a sharing free-for-all. First, competition between take-down companies may drive investment into better techniques for identifying new phishing websites faster, and mandated sharing might undermine the incentive to innovate. Unsurprisingly, most take-down companies would rather see banks purchase the services of several take-down providers to overcome gaps in coverage.

In this paper, we describe the Phish-Market protocol, which addresses the competitive concerns of take-down companies so that widespread sharing can take place. To bolster the incentive to share, our protocol enables sharing of URLs where the net contributors are compensated without revealing the sensitive details of what is shared to competitors. At a high level, the Phish-Market protocol does the following:

1. shares only those URLs that the receiving party wants (i.e., the banks the receiving party works for);
2. does not reveal to the providing party which URLs are given to the receiving party;
3. securely tallies the number of URLs given to the receiving party;
4. does not count URLs the receiving party already has.

Timing is critical when it comes to distributing URL feeds — the longer a phishing website remains online, the more customer credentials may be at risk. While

in theory generic multiparty computation protocols can be used to implement this mechanism, in practice they are very inefficient and would introduce significant delays in processing the many thousands of phishing websites. In contrast, our custom protocol is extremely efficient (and still provably secure).

To demonstrate the feasibility of our mechanism, we have implemented an elliptic-curve-based version of the protocol in Java. Using the feeds from two take-down companies during the first two weeks of April 2009, we tested protocol performance in a real-world scenario. We found that our sharing protocol introduces an average delay of 5 seconds to the processing and transmission per phishing URL. In exchange for this very short delay, information on new phishing websites is exchanged between take-down companies so that the overall lifetime of phishing websites may be halved [12] while crediting the contributing firm.

2 The Phish-Market Protocol

We now describe the Phish-Market protocol, where companies with more comprehensive feeds of phishing URLs are compensated for sharing with those who learn most from sharing. The protocol deals with a number of constraints in order to satisfy the exchanging parties without relying on a trusted third party. While we formalize the security properties guaranteed in Section 2.2, it is helpful to first mention the requirements affecting the protocol’s design. In particular, each company is only interested in a subset of their competitors’ feeds, namely those URLs that affect their own customers. As an added complexity, take-down companies keep their list of client banks secret from competitors. Hence, we need a way to share only those URLs that the other party is interested in, without revealing which URLs are being shared. Note that our mechanism does not directly compensate contributors; instead, it tallies the total number of useful URLs exchanged in a way that cannot be manipulated by either party.

An Optimal Ideal-World Protocol. We describe the task our protocol performs by first explaining how it could be done if we used a trusted third party (TTP) — someone who was entirely trusted by both the contributor (or *Seller*) and the receiver (or *Buyer*). To share data in this ideal scenario, both the Buyer and the Seller would send the data to the TTP; the Buyer’s data consists of the URLs she already knows and her list of client banks, while the Seller’s data consists of the URLs he is attempting to sell and their classification (i.e., which bank each URL is attempting to impersonate). The TTP could then send the Buyer only those URLs that both impersonate her clients and that she did not already know. The TTP would send the Seller the number of URLs sent to the Buyer. This number would then be used to compute the compensation owed to the Seller. Since the TTP only sends the new “interesting” URLs to the Buyer, she will not learn anything about URLs she was not interested in (and would not have to pay for them). On the other hand, the TTP sends the Seller only the number of URLs sold, not the URLs themselves. Consequently, the Seller will not gain additional information about the Buyer’s client list.

Our protocol is intended to provide this functionality, maintaining its privacy properties, but without requiring a third party. Using powerful results from theoretical cryptography, it is known how to convert any task that can be performed with the aid of a TTP to one that does not require third parties. However, these techniques are usually inefficient. In our case, even the most efficient implementations of general techniques (such as the Fairplay system [10]) would be orders of magnitude too slow for practical use.

We give an efficient protocol for executing a single ‘transaction’ of the following form: the Seller first sends a ‘tag’ to the Buyer. The tag can be, for example, the name of the bank associated with the URL to be sold. The Buyer uses the tag to decide whether or not she is interested in learning the corresponding URL. She also commits in advance to the set of URLs she already knows. If the Buyer was interested in the tag and did not already know the corresponding URL, the Seller receives a ‘payment’. Otherwise, the Seller receives a ‘counterfeit payment’ (the Seller should not be able to tell whether or not a payment is counterfeit — otherwise he would be able to tell whether or not the Buyer was interested in the URL, and thus discover the Buyer’s client list).

At the end of some previously agreed period (or number of transactions), the Buyer reveals to the Seller how many ‘real’ payments were sent, and proves that this is indeed the case (without revealing which of the payments were real). In practice, we envision each pair of take-down companies executing the basic protocol in both directions: when one of the companies acquires a new URL, it would execute the protocol as the Seller, with the other company playing the Buyer. When the second company acquires a new URL, it would execute an instance of the protocol in the other direction, with the first party as Buyer and the second as Seller.

Note that, even when using a trusted third party, some attacks are still possible. For example, there is no guarantee that the URLs sold will be useful or correctly tagged. A malicious Seller could send random strings instead of URLs, forcing the Buyer to ‘pay’ for garbage URLs (since they would not appear in the Buyer’s database). A malicious Seller can also attack the Buyer’s privacy: if he uses the same tag for all the URLs in a certain period, the Seller can tell whether or not the Buyer is interested in the tag by whether or not a payment was made at the end of the period.

Since these attacks can be carried out in the ideal world, any protocol implementing this type of exchange is also vulnerable. For the situations in which we anticipate our protocol will be used, however, there are mitigating strategies. First, the Buyer can evaluate the URLs she learns and set the price she is willing to pay for each URL based on the quality of URLs she received in the past. If she determines that the Seller is providing low-quality URLs, the Buyer can request a lower dollar price per URL or refuse to do business with that Seller in the future. This would mitigate the “garbage URL” attack. Defending against the privacy breach attack is harder — the payment will always leak some information about which tags the Buyer is interested in. We can help the Buyer detect this type of attack by compromising a little on the Seller’s privacy: if we give the

Buyer all the tags the Seller uses (without the corresponding URLs), the Buyer can verify that no set of tags is overly represented.

Finally, in a two-party protocol, unlike a protocol that uses a trusted third party, each side can decide to abort the protocol prematurely. This affects the security of our protocol if the Buyer decides to abort after learning a URL but before making the payment. However, the same problem exists in many remote transactions (e.g., when purchasing physical goods over the phone, the seller can refuse to send the goods after receiving payment). The same legal frameworks can be used to handle a refusal to pay in this case.

Below, we describe the protocol as well as the precise security guarantees we make.

2.1 Protocol Overview

Payment Commitments. Before we describe the protocol itself, we must clarify what we mean by ‘real’ and ‘counterfeit’ payments. Our protocol uses cryptographic commitments as payment tokens. Loosely speaking, a commitment to a value x can be thought of as a public-key encryption of x , for which only the Buyer knows the secret key; the Seller can’t tell what x is from the commitment, but the Buyer can ‘open’ a commitment and prove to the Seller that the commitment is to a specific value. In our protocol, a ‘real’ payment is a cryptographic commitment to the number 1, while a ‘counterfeit’ payment is a commitment to the number 0.

The payment commitments used by the protocol have a special property that allows them to be efficiently aggregated, even in encrypted form (they are *homomorphic*). Thus, the Seller can take the ‘payments’ from multiple executions of the basic protocol and compute a commitment to the total payment (the number of URLs actually ‘sold’).

The Buyer will eventually open the aggregated commitment. At this point, the Seller will learn only the total number of ‘real’ payments received (and not which individual payments were real). This value can be used as the basis for a monetary transaction between the two parties.

Protocol Construction. One of the more difficult challenges to solve efficiently is that the Buyer should not have to pay for URLs she already knew, while simultaneously protecting the privacy of the Buyer’s client list. The known techniques for general secure computation of a function require an expensive public-key operation for each input (or even each bit of the input). In our case, the input would have to include the set of previously known URLs, which may be very large: A typical take-down company could learn an excess of 10 000 URLs per month, making existing systems impractical.

To solve this problem, we let the Buyer perform the database search locally, after learning the URL. If she discovers the URL in the database, she must then prove to the Seller that the URL existed in the database before the start of the transaction. However, this proof cannot use the URL itself, since that would reveal to the Seller that the Buyer was interested in it (thus exposing one of

the Buyer's clients). The main idea behind the protocol is to split the proof into two:

1. The first proof is a 'proof of payment'. The payment in this case is a commitment to the value 1; the proof of payment proves that the Buyer can open the commitment she sent to the value 1.
2. The second proof is a 'proof of previous knowledge'. This proof convinces the Seller that the Buyer knew the URL before the start of the protocol.

The essence of the protocol is that we allow the Buyer to 'fake' a proof if she knows a corresponding secret key. The protocol is set up so that the Buyer initially knows a single secret key: she can fake the first proof or the second proof, but not both. Once the Buyer learns the tag, she must make a choice: she can either learn the corresponding URL, or learn the second secret key (but not both). Thus, if she chooses not to learn the URL, the Buyer can send a counterfeit payment (a commitment to 0), and fake both proofs. If she chooses to learn the URL and did not already know it, she is forced to fake the second proof, and therefore cannot fake the first (so she must send a real payment). The proofs we use are *Zero-Knowledge* (ZK) proofs: the Seller learns nothing from the proof except the validity of its statement. This protects the privacy of the Buyer (the Seller cannot tell whether or not the Buyer was interested in the URL or whether she previously knew it).

Fig. 1 shows a graphical overview of the protocol. We split the second proof into the boxes labeled *ZK Proof #2* and *Proof #3* in the figure. Before the protocol begins, the Buyer sends the Seller a commitment to her set of previously known URLs. ZK Proof #2 proves the Buyer holds a commitment for the URL (this part can be faked using a secret key). Proof #3 proves the Buyer knew the commitment before the protocol began (this part cannot be faked; however, if the Buyer faked ZK Proof #2 she can choose an arbitrary commitment and prove she knew that). The reason for the split is that Proof #3 can be performed very efficiently, while Proof #2 requires public-key type operations. The numbers on the left and right-hand sides of the figure reference the corresponding lines in the full protocol listing (on the left these are the lines in Prot. 1a, and on the right in Protocols 1b and 2).

To simplify the presentation, the protocol in Fig. 1 omits two steps present in the full protocol:

1. The Buyer must prove that the payment is valid (either a commitment to 0 or a commitment to 1). Otherwise, if the Buyer fakes the first proof she could send a commitment to a negative number instead of a zero commitment (in which case the aggregate commitment would be opened to a lower value than the actual payment due).
2. The use of a Merkle tree as a set commitment (Proof #3) is not completely secure if the same Merkle tree is used for multiple transactions. This is because every execution of the protocol requires the Buyer to reveal a path from some leaf in the tree to the root. If the Seller sees the same leaf twice, he will learn that in at least one of the transactions the Buyer was using

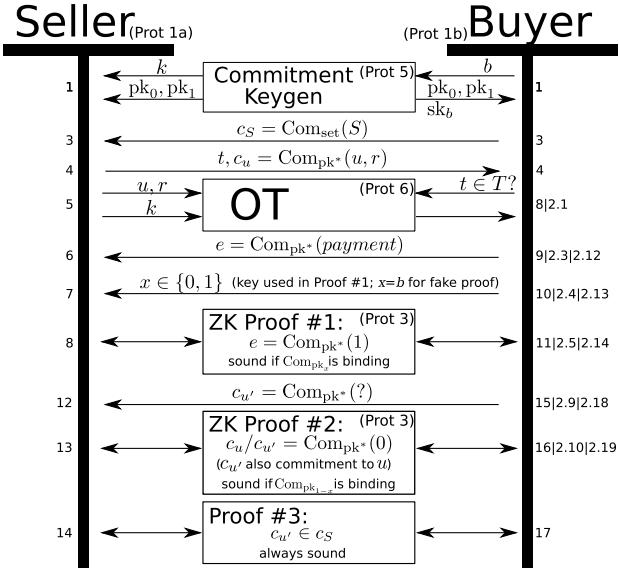


Fig. 1. Simplified Phish-Market protocol overview

a “fake”. To prevent this attack, the Buyer must make sure the tree also contains “chaff” commitments. When a fake commitment is needed, the Buyer uses one of the chaff commitments. The Buyer makes sure to use each chaff commitment at most once.

2.2 Security Properties

Unlike errors in most computer algorithms, protocols with faulty security may perform flawlessly — often by definition a failure in security is one that is undetected. Thus, an important part of the specification for any secure protocol is a formal definition of its security properties and an analysis of the conditions under which they are guaranteed.

We make separate security guarantees for the Seller and for the Buyer in each transaction (execution of the basic protocol).

Buyer’s Security. The Buyer in our protocol has as input a set of tags in which she is interested, T (e.g., the list of banks she has as clients) and a set of previously known URLs, U . The security guarantee for the Buyer is that a malicious Seller does not learn anything about T or U , beyond what he can deduce from the payment amount. This is important because competitors naturally do not wish to reveal weaknesses (in terms of gaps in URL coverage). On the other hand, the Seller does not want to reveal URLs to the Buyer that the Buyer is unaware of without compensation. Finally, the Buyer does not want to reveal its client list to the Seller.

More formally:

Theorem 1 (Buyer’s Security). *For any two sets of inputs (T_0, U_0) and (T_1, U_1) , such that $|U_0| = |U_1|$, the Seller’s view of a protocol execution when the Buyer is given input (T_0, U_0) is statistically indistinguishable from its view when the Buyer is given input (T_1, U_1) .*

Note that the Seller’s view of the protocol does *not* include the opening of the aggregate payment: the Seller will obviously gain some information about T and U from the payment amount — what the theorem implies is that this is *all* the Seller learns.

Due to limited space, we defer the proof this theorem to the full version of the paper.

Seller’s Security. Essentially, the Seller needs to ensure that he is being justly compensated for each URL that the Buyer learns from him. We define the security of the Seller by formally comparing our protocol to an ‘ideal world’ in which there exists a completely trusted third party (the ‘ideal Phish-Market functionality’). The protocol in the ideal world is much simpler than that in the real world, hence its security guarantees are easier to understand intuitively. We prove our protocol’s security by showing that any attacks by the Buyer on the protocol in the real world (without the trusted third party) can be performed in the ideal world as well. Hence, our intuitions for the ideal world must hold for the real world too (this is the ideal/real simulation paradigm).

Below, we describe the ideal-world protocol for a single transaction. In both the real and the ideal world, the Buyer’s inputs consist of T , a set of tags in which the Buyer is interested, and U , a set of previously known URLs. The Seller’s inputs consist of a tag t and a URL u . The output of the protocol, on the Buyer’s side, is the tag t , and optionally the URL u (if the Buyer was interested in it). On the Seller’s side, the output is a payment commitment. We denote $\text{Com}_{\text{pk}^*}(x)$ a commitment to a value x .²

The protocol in the ideal world proceeds as follows:

- 1: The ideal functionality waits for the Buyer to send U and the Seller to send t, u .
- 2: The functionality then sends t to the Buyer and waits for the Buyer to respond.
- 3: **if** The Buyer responds with 0 (she’s interested in t) **then**
- 4: The functionality sends u to the Buyer.
- 5: **if** $u \notin U$ **then**
- 6: The functionality sends $e = \text{Com}_{\text{pk}^*}(1)$ to the Seller.
- 7: **else** // $u \in U$
- 8: The functionality sends $e = \text{Com}_{\text{pk}^*}(0)$ to the Seller.
 (The functionality will allow a corrupt Buyer to send $e = \text{Com}_{\text{pk}^*}(1)$ in this case as well)

² For clarity, we’re ignoring the fact that the commitments are randomized — the commitment function is actually $\text{Com}_{\text{pk}^*}(x, r)$, where r is the commitment’s randomizer).

```

9: else // The Buyer is not interested in  $t$ 
10:   The functionality sends  $e = \text{Com}_{\text{pk}^*}(0)$  to the Seller.

```

We allow a corrupt party to abort the computation at any point, in which case the other party will receive a special \perp symbol from the ideal functionality (this corresponds to a cheating party being detected). This ideal-world protocol is very similar to the optimal ideal-world protocol described in the beginning of this section. However, in this protocol the ideal party always sends the tag to the Buyer, and if the Buyer is interested, always sends the URL to the Buyer (rather than only sending those URLs that were both interesting and not previously known). This extra ‘information leakage’ (compared to the optimal protocol) is the result of allowing the Buyer to perform the database lookup on her own.

Formally, the Seller’s security is defined as follows:

Theorem 2 (Seller’s Security). *For any set of inputs to the Buyer and Seller, and for every (probabilistic polynomial-time) adversary \mathcal{A} that corrupts the Buyer in the real world, there exists a simulator \mathcal{S} who corrupts the Buyer in the ideal world such that the outputs of both parties in the ideal world (the ideal-world Seller and \mathcal{S}) are computationally indistinguishable from the outputs of both parties in the real world (the real-world Seller and \mathcal{A}), under the assumption that the underlying cryptographic primitives are secure.*

Due to limited space, we defer the proof this theorem to the full version of the paper.

Side-Channel Attacks. As with every ‘provably secure’ system, the proof of security only holds as long as certain assumptions are met. For example, it may be possible to break the security of the protocol if the parties receive information outside the ‘legitimate’ channels specified by the protocol (these unanticipated information channels are called *side channels*).

The Phish-Market protocol is potentially vulnerable to a timing side-channel attack: the Seller can measure the time it takes the Buyer to complete a transaction. If this time depends on whether or not she was interested in the tag, or on whether or not she already knew the URL, the Seller will gain information about the Buyer’s client list or coverage rate. This particular attack can be foiled with relatively little effort by adding artificial delays to the code to ensure all code paths on the Buyer’s side take the same time³. Of course, as in the case of any secure protocol, the Phish-Market protocol may be vulnerable to other side-channel attacks that we did not anticipate.

2.3 Formal Protocol Definition

We give a full protocol listing (in pseudocode) below. We describe separately the pseudocode for the Sellers’ and Buyers’ sides of the protocol. To make the protocol listing easier to read, we divide it into a number of smaller subprotocols

³ Note that the delays are not random noise — the delay on each code path must be computed so that the total time taken by the Buyer does not depend on her input.

(called as subroutines from the top-level protocol, Prot. 1). Prot. 1a specifies the top-level protocol run by the Seller and Prot. 1b that run by the Buyer. Prot. 2 is called by the Buyer when she is interested in the tag sent by the Seller (the Seller’s side of the protocol looks the same whether or not the Buyer was interested). Prot. 3 is used to prove knowledge of a given commitment value; this protocol has three “sides”: Prot. 3a is the Seller’s view of the proof (verification), Prot. 3b is the Buyer’s view when performing a “real” proof, and Prot. 3c is the Buyer’s view when performing a “fake” proof (using the trapdoor key).

Pedersen commitments are based on the hardness of discrete log in a group \mathcal{G} of prime order p (plaintext values are in the group \mathbb{Z}_p). The commitment public-key is a pair of generators $g, h \in \mathcal{G}$. The commitment is binding iff it is hard to compute the discrete log of g with respect to h . Throughout the protocol, we denote a Pedersen commitment under public-key pk to a value x and with randomizer r by $\text{Com}_{\text{pk}}(x, r)$.

A party that knows $\text{sk} = \log_g h$ can open a commitment to any value (thus, this value acts as a trapdoor for the commitment). The Phish-Market protocol requires a key-generation protocol with special properties: it must generate a pair of commitment-keys such that the Buyer knows only one of the corresponding trapdoors, but the Seller does not know which (the Seller is allowed to learn both). This is easily done using standard techniques; we leave the details to the full version.

We assume the two parties have previously agreed on a Pedersen public-key pk^* that is binding to both parties (i.e., neither party knows its trapdoor key).

3 Performance Evaluation

3.1 Theoretical Efficiency

The advantage of this protocol over a generic secure-computation is its efficiency. We measure efficiency in terms of both computation and communication overhead. In Section 3.2, we describe our implementation of the protocol, which is instantiated using Pedersen commitments and the Naor-Pinkas OT protocol. To get a theoretical estimate of the protocol’s efficiency we count the most expensive operations — those that dominate the protocol’s overall cost.

Exponentiations are the most expensive computation required, while the main communications requirement is the exchange of group elements and hashes. For each URL transmitted, the Seller must compute 34 exponentiations, while transmitting 10 group elements and 2 hashes to the Buyer. Meanwhile, the Buyer’s computation load is a bit lighter but the communications requirements are slightly higher. The Buyer computes just 24 exponentiations, in addition to sending 39 group elements and $\log |U| + 1$ hashes to the Seller. The complete costs, broken down according to each protocol component, are given in Table 2.

Protocol 1a. Phish-Market Protocol: Seller

Input: Commitment public-key, pk^* , such that Buyer does not know corresponding secret key.

Input: A URL, u , with tag, t

- 1: Perform Commitment Key Generation.
The Seller learns the new commitment public keys $(\text{pk}_0, \text{pk}_1)$ and the secret $k = \text{sk}_0 + \text{sk}_1$ (where $(\text{sk}_0, \text{sk}_1)$ are the corresponding trapdoors) // Learning k will allow Buyer to compute both trapdoors if given one
- 2: Perform Commitment Key Generation.
Denote the resulting commitment keys $(\text{pk}_2, \text{pk}_3)$ (discard the secret).
- 3: Wait to receive Merkle root c_U from Buyer // Root of a Merkle hash tree whose leaves are commitments to known URLs
- 4: Choose $r \in_R \mathbb{Z}_p$.
Send $(t, \text{Com}_{\text{pk}^*}(H(u), r))$ to Buyer.
- 5: Perform OT protocol as sender (Buyer as receiver) with input strings $s_0 = (u, r)$ and $s_1 = k$.
- 6: Wait to receive commitment e from Buyer. // ‘payment’ commitment
Verify that $e \in \mathcal{C}$.
- 7: Wait to receive bit b from Buyer. // Payment proof based on binding of Com_{pk_b}
- 8: Verify that $e = \text{Com}_{\text{pk}^*}(1)$ using Com_{pk_b} for coin-flipping (Prot. 3a). // Proves that Buyer can either open e to 1 or knows sk_b
- 9: Wait to receive bit $b' \in \{2, 3\}$ from Buyer. // Payment validity proof based on binding of $\text{Com}_{\text{pk}_{b'}}$
- 10: Verify that $e = \text{Com}_{\text{pk}^*}(1)$ using $\text{Com}_{\text{pk}_{b'}}$ for coin-flipping (Prot. 3a).
- 11: Verify that $e = \text{Com}_{\text{pk}^*}(0)$ using $\text{Com}_{\text{pk}_{5-b'}}$ for coin-flipping (Prot. 3a). // Together with previous step proves that Buyer can open e to either 0 or 1
- 12: Wait to receive commitment c_u from Buyer // Buyer’s ‘previously known commitment’ to u
Verify that $c_u \in \mathcal{C}$.
- 13: Let $c_{\text{test}} \leftarrow \frac{\text{Com}_{\text{pk}^*}(H(u), r)}{c_u}$.
Verify that $c_{\text{test}} = \text{Com}_{\text{pk}^*}(0)$ using $\text{Com}_{\text{pk}_{1-b}}$ for coin-flipping (Prot. 3a) // Proves that either Buyer can open c_u to $H(u)$ or that Buyer knows sk_{1-b}
- 14: Verify proof that c_u is in set committed to by c_U (e.g. verify a Merkle path from c_u to c_U).

Protocol 1b. Phish-Market Protocol: Buyer

Input: Commitment public-key, pk^*

Input: Set of commitments to known URLs: U = $\{c_{u_1} = \text{Com}_{\text{pk}^*}(H(u_1), r_1), \dots, c_{u_{|U|}} = \text{Com}_{\text{pk}^*}(H(u_{|U|}), r_{|U|})\}$

Input: Set of wanted tags, T

- 1: Perform Commitment-Generation with input bit b .
The Buyer learns the new commitment keys pk_0, pk_1 and trapdoor sk_b .
- 2: Perform Commitment-Generation with input bit $b' \in \{2, 3\}$.
Denote the resulting commitment keys pk_2, pk_3 and trapdoor $\text{sk}_{b'}$.
- 3: Generate a ‘chaff’ commitment: $c_{\text{chaff}} \in_R \mathcal{C}$.
Let $U' \leftarrow U \cup \{c_{\text{chaff}}\}$.
Send $\text{Com}_{\text{set}}(U')$ to Seller (e.g. the root of a Merkle hash tree with elements of U' as the leaves) // Commitment to set of already known URLs
- 4: Wait to receive (t, c_u) from Seller // Tag and commitment to URL
- 5: **if** $t \in T$ **then** // Buyer is interested in tag
- 6: Run Subprotocol 2
- 7: **else** // Buyer is not interested in tag
- 8: Perform OT protocol as receiver (Seller as sender) with choice bit 1.
Denote result sk_0, sk_1
- 9: Choose $r_e \in_R \mathbb{Z}_p$.
Send $e = \text{Com}_{\text{pk}^*}(0, r_e)$ to Seller // ‘Fake’ payment
- 10: Send b to Seller // Use Com_{pk_b} for payment proof
- 11: ‘Prove’ that $e = \text{Com}_{\text{pk}^*}(1)$ using Com_{pk_b} and sk_b (Prot. 3c).
- 12: Send b' to Seller // Use $\text{Com}_{\text{pk}_{b'}}$ for payment validity proof
- 13: ‘Prove’ that $e = \text{Com}_{\text{pk}^*}(1)$ using $\text{Com}_{\text{pk}_{b'}}$ and $\text{sk}_{b'}$ (Prot. 3c).
- 14: Prove that $e = \text{Com}_{\text{pk}^*}(0)$ using $\text{Com}_{\text{pk}_{5-b'}}$ and r_e (Prot. 3b).
- 15: Choose a ‘chaff’ commitment $c_u \in U$
Send c_u to Seller.
- 16: Let $c_{\text{test}} \leftarrow \frac{c_{u'}}{c_u}$.
‘Prove’ that $c_{\text{test}} = \text{Com}_{\text{pk}^*}(0)$ using $\text{Com}_{\text{pk}_{1-b}}$ and sk_{1-b} (Prot. 3c).
- 17: Prove that c_u is in set committed to by $\text{Com}_{\text{set}}(U)$ (e.g. show a Merkle path from c_u to c_U).

Protocol 2. Phish-Market Subprotocol: Buyer is Interested in t

-
- 1: Perform OT protocol as receiver (Seller as sender) with choice bit 0.
Denote result u, r'
 - 2: **if** $c_u' = \text{Com}_{\text{pk}^*}(H(u), r')$ and $\exists i : c_{u_i} \in U$ and $u_i = u$ **then** // Buyer already knows u
 - 3: Choose $r_e \in \mathcal{R} \mathbb{Z}_p$.
Send $e = \text{Com}_{\text{pk}^*}(0, r_e)$ to Seller // ‘Fake’ payment
 - 4: Send b to Seller // Use Com_{pk_b} for payment proof
 - 5: ‘Prove’ that $e = \text{Com}_{\text{pk}^*}(1)$ using Com_{pk_b} and sk_b (Prot. 3c).
 - 6: Send b' to Seller // Use $\text{Com}_{\text{pk}_{b'}}$ for payment validity proof
 - 7: ‘Prove’ that $e = \text{Com}_{\text{pk}^*}(1)$ using $\text{Com}_{\text{pk}_{b'}}$ and $\text{sk}_{b'}$ (Prot. 3c).
 - 8: Prove that $e = \text{Com}_{\text{pk}^*}(0)$ using $\text{Com}_{\text{pk}_{5-b'}}$ and r_e (Prot. 3b).
 - 9: Let $c_u = \text{Com}_{\text{pk}^*}(H(u), r)$ such that $c_u \in U$.
Send c_u to Seller.
 - 10: Let $c_{\text{test}} \leftarrow \frac{c_{u'}}{c_u} = \text{Com}_{\text{pk}^*}(0, r' - r)$.
Prove that $c_{\text{test}} = \text{Com}_{\text{pk}^*}(0)$ using $\text{Com}_{\text{pk}_{1-b}}$ and $r' - r$ (Prot. 3b).
 - 11: **else** // Buyer did not know u or Seller is cheating
 - 12: Choose $r_e \in \mathcal{R} \mathbb{Z}_p$.
Send $e = \text{Com}_{\text{pk}^*}(1, r_e)$ to Seller // ‘Real’ payment
 - 13: Send $1 - b$ to Seller // Use $\text{Com}_{\text{pk}_{1-b}}$ for payment proof
 - 14: Prove that $e = \text{Com}_{\text{pk}^*}(1)$ using $\text{Com}_{\text{pk}_{1-b}}$ and r_e (Prot. 3b).
 - 15: Send $5 - b'$ to Seller // Use $\text{Com}_{\text{pk}_{5-b'}}$ for payment validity proof
 - 16: Prove that $e = \text{Com}_{\text{pk}^*}(1)$ using $\text{Com}_{\text{pk}_{5-b'}}$ and r_e (Prot. 3b).
 - 17: ‘Prove’ that $e = \text{Com}_{\text{pk}^*}(0)$ using $\text{Com}_{\text{pk}_{b'}}$ and $\text{sk}_{b'}$ (Prot. 3c).
 - 18: Send c_{chaff} to Seller.
 - 19: Let $c_{\text{test}} \leftarrow \frac{c_{u'}}{c_{\text{chaff}}}$.
‘Prove’ that $c_{\text{test}} = \text{Com}_{\text{pk}^*}(0)$ using Com_{pk_b} and sk_b (Prot. 3c).
-

Protocol 3a. Proof of Committed Value: Seller

Input: Commitment c and claimed value x // Commitment uses public key pk^*

Input: Trapdoor Commitment public key pk // Used for coin flipping

- 1: Wait to receive c_{chal} from Buyer.
 - 2: Wait to receive (b, c_b) from Buyer.
Verify that $c_b \in C$.
 - 3: Choose $\text{chal}_1 \in \mathcal{R} \mathbb{Z}_p$
Send chal_1 to Buyer.
 - 4: Wait to receive $(\text{chal}_0, r_{\text{chal}})$ from Buyer.
Verify that $c_{\text{chal}} = \text{Com}_{\text{pk}}(\text{chal}_0, r_{\text{chal}})$.
 - 5: Wait to receive r' from Buyer.
Let $\text{chal} \leftarrow \text{chal}_0 + \text{chal}_1$.
Verify that $c^{\text{chal}} \cdot c_b = \text{Com}_{\text{pk}^*}(\text{chal} \cdot x + b, r')$.
-

3.2 Implementation Performance

We implemented an elliptic-curve (EC) based version of the protocol in Java, using the Bouncy Castle Crypto API⁴ for basic EC operations. In our implementation the Merkle hash-tree was kept entirely in memory. This is feasible even for moderately large URL lists (e.g., in one of the experiments the tree consisted of about 18 000 URLs).

Our experiments used the NIST-recommended EC curve P-256 [14] for the group underlying both the Pedersen commitments and Naor-Pinkas OT, and SHA-1 in place of a “random oracle”. Both sides of the protocol were simulated on a single server with one dual-core 2.4GHz Intel Xeon processor and 2GB of

⁴ <http://www.bouncycastle.org/>

Protocol 3b. Proof of Committed Value: Buyer

Input: Commitment $c = \text{Com}_{\text{pk}^*}(x, r_x)$, r_x and claimed value x // *Commitment uses public key pk^**

Input: Trapdoor Commitment public key pk // *Used for coin flipping*

- 1: Choose $\text{chal}_0 \in_{\mathcal{R}} \mathbb{Z}_p$ and $r_{\text{chal}} \in_{\mathcal{R}} \mathbb{Z}_p$.
Send $\text{Com}_{\text{pk}}(\text{chal}_0, r_{\text{chal}})$ to Seller.
- 2: Choose $b \in_{\mathcal{R}} \mathbb{Z}_p$ and $r_b \in_{\mathcal{R}} \mathbb{Z}_p$
Send $(b, \text{Com}_{\text{pk}^*}(b, r_b))$ to Seller
- 3: Wait to receive chal_1 from Seller.
- 4: Send $(\text{chal}_0, r_{\text{chal}})$ to Seller.
- 5: Let $\text{chal} \leftarrow \text{chal}_0 + \text{chal}_1$.
Compute r' such that $c_{\text{chal}}^{\text{chal}} \cdot \text{Com}_{\text{pk}^*}(b, r_b) = \text{Com}_{\text{pk}^*}(\text{chal} \cdot x + b, r')$. // r' can be efficiently computed using r_b and r_x .
Send r' to Seller.

Protocol 3c. Fake Proof of Committed Value: Buyer

Input: Commitment c and claimed value x // *Commitment uses public key pk^**

Input: Trapdoor Commitment public and secret keys pk, sk // *Used to fake coin flipping*

- 1: Choose $r'_{\text{chal}} \in_{\mathcal{R}} \mathbb{Z}_p$.
Let $c_{\text{chal}} \leftarrow \text{Com}_{\text{pk}}(0, r'_{\text{chal}})$.
Send c_{chal} to Seller. // Using sk and r'_{chal} , Buyer can open c_{chal} to any value
- 2: Choose $\text{chal} \in_{\mathcal{R}} \mathbb{Z}_p$, $b \in_{\mathcal{R}} \mathbb{Z}_p$ and $r' \in_{\mathcal{R}} \mathbb{Z}_p$.
Let $c_{\text{target}} \leftarrow \text{Com}_{\text{pk}^*}(\text{chal} \cdot x + b, r')$.
Let $c_b \leftarrow \frac{c_{\text{target}}}{c_{\text{chal}}}$.
Send (b, c_b) to Seller. // Buyer does not know how to open c_b
- 3: Wait to receive chal_1 from Seller.
- 4: Let $\text{chal}_0 \leftarrow \text{chal} - \text{chal}_1$.
Compute r_{chal} s.t. $c_{\text{chal}} = \text{Com}_{\text{pk}}(\text{chal}_0, r_{\text{chal}})$. // r_{chal} can be efficiently computed using sk and r'_{chal} .
Send $(\text{chal}_0, r_{\text{chal}})$ to Seller.
- 5: Send r' to Seller. // c_b and b were computed at step 2 such that $c_{\text{chal}}^{\text{chal}} \cdot c_b = \text{Com}_{\text{pk}^*}(\text{chal} \cdot x + b, r')$

memory (the protocol is CPU bound — one transaction requires less than 3kB of communication — so running both sides on one server would only cause us to overestimate the running time).

To test the protocol’s performance under real-world conditions, we used the URL feeds from two large take-down companies during the first two weeks of April 2009. We assigned one of the take-down companies to be the Seller, while making the other the Buyer (we ran experiments using both assignments). For the two-week sample period, one company found 8 582 unique URLs while the other discovered 17 721 URLs. The first company was interested in obtaining phishing URLs for 59 banks, and the second for 54 banks, according to the client lists shared with the authors.

The primary metric we use to measure the performance of our implementation is the time required to process and transmit each phishing URL from the seller to the buyer. On average, each URL faced a very acceptable delay of 5.13 seconds to complete the exchange (3.19 second median). Two main factors affect the total delay. First is the processing time required to execute the protocol. This computational time was very consistent, taking an average of 2.37 seconds, but never more than 4.02 seconds. The other, less predictable, reason for delay happens whenever many URLs are discovered around the same time. When this occurred, some URLs had to wait for other URLs to be processed, leading to a longer delay. While a multi-threaded implementation could

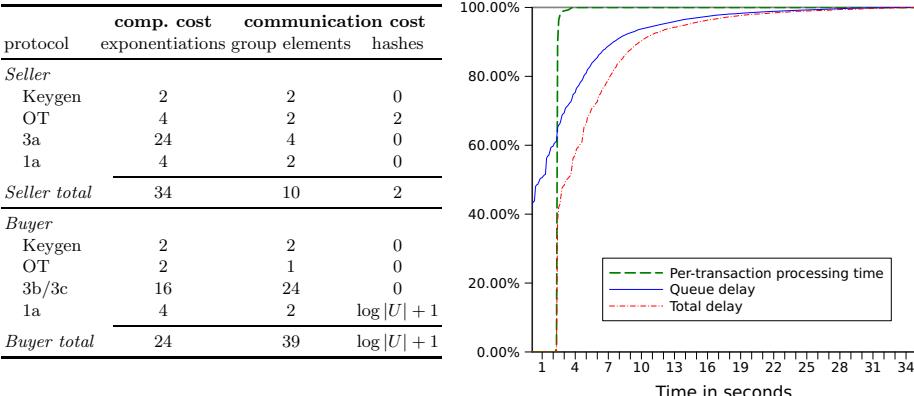


Fig. 2. Theoretical computation and communication costs of the Phish-Market protocol (left); observed cumulative distribution function of the time required to share each phishing URL (right)

minimize these ‘queue delays’ (by utilizing more CPU cores), we chose to implement the protocol using a single thread to demonstrate its feasibility even with modest hardware. Moreover, note that the protocol implementation was optimized for clarity and generality of the source code rather than speed. The average queue delay caused by waiting on other URLs to finish processing was 2.76 seconds, while the longest delay was 34.6 seconds.

To get a better feel for how the processing time varies, Figure 2 (right) plots the cumulative distribution functions for the time taken to process each URL, the time that URL spent waiting in the Seller’s queue, and the total delay between the time the URL entered the Seller’s queue and the time the Buyer received it. 48.4% of URLs were processed in under 3 seconds, yet 9.7% took more than 10 seconds. Despite the variation, no URL took more than 37 seconds to process. Given that phishing website removal requires human intervention, a 37 second delay is negligible, and certainly much better than the many days longer unknown sites currently take to be removed!

In addition to the total delay (red dash-dot line), Figure 2 (right) plots the two key components of delay. The green dash line appears nearly vertical around 2 seconds, suggesting that the per-URL processing time is very consistent. Meanwhile, the blue solid line plots the queue delay, which accounts for the stretched tail of the overall delay. Hence, if the queue delay were reduced by using multiple processors or threads, the total delay might approach the consistently shorter processing time.

4 Related Work

Sharing Attack Data. The academic work on phishing has been diverse, with a useful starting point being Jakobsson and Myers’ book [9]. However, there has been only limited examination of the take-down process employed by the banks

and specialist companies, even though it is the primary defense employed today. Moore and Clayton estimated the number and lifetimes of phishing websites and demonstrated that timely removal reduced user exposure [11]. Subsequently, they presented evidence (repeated in Section 1) showing that take-down companies do not share data on phishing websites with each other, and they calculated that website lifetimes might be halved if companies shared their URL feeds. They appealed to the greater good in advocating that take-down companies voluntarily exchange URL feeds with each other at no charge. By contrast, this paper proposes a mechanism for sharing where net contributors are compensated by net receivers of phishing URLs.

Information sharing has long been recognized as necessary for improving information security. Gordon and Ford discussed early forms of sharing in the anti-virus industry and contrasted it with sharing when disclosing vulnerabilities [8]. Some have worried that firms might free-ride off the security expenditures of other firms by only ‘consuming’ shared information (e.g., phishing feeds) and never providing any data of their own [7], while others have argued that there can also be positive economic incentives for sharing security information [5].

Cryptographic Protocols. The Phish-Market protocol is an instance of *secure multiparty computation* (MPC). MPC has been a major area of work in theoretical cryptography, and general techniques are known for securely computing any functionality [15,6,3,1].

These techniques, however, are not practical for computing functions that have large input size (e.g., an optimized implementation of Yao’s protocol for general two-party computation can take seconds to evaluate a simple function with 32-bit inputs [10]). In our case, one of the inputs to the function is a database of previously-known URLs containing thousands of entries, making general techniques completely impractical.

For many specific functionalities, efficient protocols are known. We use some of these as subroutines in our protocol. We make use of the Naor-Pinkas OT protocol [13], which is itself a more efficient version of the Bellare-Micali OT protocol [2]. We also use a generalization of the Chaum-Pedersen protocol for proving in zero-knowledge the value of a commitment [4].

5 Concluding Remarks

Security mechanisms are becoming increasingly data-driven, from identifying malware hosts to blocking spam and shutting down phishing websites. Consequently, sharing data is now essential as no single defender has a complete view of attacker behavior.

In this paper, we have devised a mechanism to make it easier for take-down companies to interact: by compensating net contributors of phishing URLs, we can bolster the incentive to share while rewarding investment into better discovery techniques. As a bonus, our protocol has the desirable property of being provably secure and allowing parties to share data without relying on a trusted third party to mediate. Crucially, the protocol is also efficient: our implementation

easily processed the phishing URLs in a two-week sample from two take-down companies while introducing average delays of 5 seconds before sharing.

Of course, to cut phishing website lifetimes in half and reduce the annual financial exposure due to phishing by several hundred million dollars, we must still convince the take-down companies that sharing is a good idea. We feel that the security guarantees our protocol provides will make it easier for companies to at least explore the idea of sharing data with their competitors. At present, many companies still cling to the view that their feed is best. Our protocol offers companies the chance to put their claims to the test while avoiding the potential for public embarrassment if they happen to find that sharing can indeed help.

References

1. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: STOC 1990, pp. 503–513. ACM Press, New York (1990)
2. Bellare, M., Micali, S.: Non-interactive oblivious transfer and applications. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 547–557. Springer, Heidelberg (1990)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for noncryptographic fault-tolerant distributed computations. In: STOC 1988, pp. 1–10. ACM Press, New York (1988)
4. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
5. Gal-Or, E., Ghose, A.: The economic incentives for sharing security information. *Information Systems Research* 16(2), 186–208 (2005)
6. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game — A completeness theorem for protocols with honest majority. In: ACM (ed.) STOC 1987, pp. 218–229. ACM Press, New York (1987)
7. Gordon, L., Loeb, M., Lucyshyn, W.: Sharing information on computer systems security: An economic analysis. *Journal of Accounting and Public Policy* 22(6), 461–485 (2003)
8. Gordon, S., Ford, R.: When worlds collide: information sharing for the security and anti-virus communities, IBM research paper (1999)
9. Jakobsson, M., Myers, S. (eds.): *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley, New York (2006)
10. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: USENIX Security Symposium, pp. 287–302 (2004)
11. Moore, T., Clayton, R.: Examining the impact of website take-down on phishing. In: Anti-Phishing Working Group eCrime Researchers Summit (APWG eCrime), pp. 1–13 (2007)
12. Moore, T., Clayton, R.: The consequence of non-cooperation in the fight against phishing. In: Anti-Phishing Working Group eCrime Researchers Summit (APWG eCrime), pp. 1–14 (2008)
13. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: SODA 2001, pp. 448–457. Society for Industrial and Applied Mathematics, Philadelphia (2001)
14. NIST. Digital signature standard (DSS). FIPS 186-2 (January 2000), <http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf>
15. Yao, A.C.-C.: How to generate and exchange secrets. In: FOCS 1986, pp. 162–167. IEEE Computer Society, Los Alamitos (1986)

Building Incentives into Tor^{*}

Tsuen-Wan “Johnny” Ngan¹, Roger Dingledine², and Dan S. Wallach³

¹ Google Inc.

² The Tor Project

³ Department of Computer Science, Rice University

Abstract. Distributed anonymous communication networks like Tor depend on volunteers to donate their resources. However, the efforts of Tor volunteers have not grown as fast as the demands on the Tor network. We explore techniques to incentivize Tor users to relay Tor traffic too; if users contribute resources to the Tor overlay, they should receive faster service in return. In our design, the central Tor directory authorities measure performance and publish a list of Tor relays that should be given higher priority when establishing circuits. Simulations of our proposed design show that conforming users receive significant improvements in performance, in some cases experiencing twice the network throughput of selfish users who do not relay traffic for the Tor network.

1 Introduction

Anonymizing networks such as Tor [16] and Mixminion [11] aim to protect users from traffic analysis on the Internet. That is, they help defeat attempts to catalog who is talking to whom, who is using which websites, and so on. These anonymity systems have a broad range of users: ordinary citizens who want to avoid being profiled for targeted advertisements, corporations who do not want to reveal information to their competitors, and law enforcement and government agencies who need to interact with the Internet without being noticed.

These systems work by bouncing traffic around a network of relays operated around the world, and strong security comes from having a large and diverse network. To this end, Tor has fostered a community of volunteer relay operators. This approach can provide sustainability (the network doesn’t shut down when the money runs out) and diversity (many different groups run relays for many different reasons), but it can also be a weakness if not enough people choose to operate relays to support the network’s traffic.

In fact, the number of Tor users keeps growing [30], while a variety of factors discourage more people from setting up relays; some want to save their bandwidth for their own use, some can’t be bothered to configure port forwarding on their firewall, and some worry about the possible consequences from running a relay. This growing user-to-relay ratio in turn hurts the service received by all users, leading to a classic “tragedy of the commons” situation [24].

* This research was funded, in part, by NSF grants CNS-0524211, CNS-0509297, and CNS-0959138. The first author did part of this work while at Rice University.

Worse, not all users are equal. While Tor was designed for web browsing, instant messaging, and other low-bandwidth communication, an increasing number of Internet users are looking for ways to anonymize high-volume communications. We did an informal measurement study by running a Tor exit relay at our institution, and we confirmed McCoy et al.'s results [32]: the median connection coming out of our relay looked like web browsing traffic, but the median *byte* looked like file-sharing traffic.

The Tor designers argued in 2005 [17] that having too much load on the Tor network should be self-correcting, since low bandwidth and poor performance would drive away users until the users that remain have acceptable performance. Instead, performance has remained bad for many users. We suggest this disparity is because different activities have different tolerance for bad performance: users of interactive applications like web browsing give up before the file-sharers, who are less sensitive to waiting hours for their work to complete.

How can we get more users to relay traffic? There are three common approaches to encouraging people to offer service in the p2p design space: building community, making it easier to run relays, and providing improved performance in exchange for service. So far Tor has focused most on the first two approaches, attracting people who believe in the need for anonymous communications to run relays. Tor now has over 1500 relays pushing over 1Gbit/s of aggregate traffic [31], but it still has not kept up with demand. On the other hand, an accounting scheme for tracking nodes' performance and rewarding nodes who perform well would seem to be at odds with preserving anonymity.

This paper shows how to strike a balance between these seemingly conflicting goals. We propose a solution where the central Tor directory authorities measure the performance of each relay and construct a list of well-behaving relays. Relays obtain this list from the authorities during normal updates. To allow relays to be treated differently, traffic from relays in the list is marked as high priority by other relays and receives better treatment along the whole circuit.

The rest of the paper is organized as follows. Sect. 2 provides background on Tor. Sect. 3 investigates exactly which behaviors we need to incentivize. Sect. 4 describes our proposed design, and Sect. 5 presents simulation results showing our design improves performance for listed relays, even as traffic from other users increases. We discuss the results in Sect. 6, and evaluate the new risks our design introduces, the most notable of which is that we end up with two anonymity sets: the group of well-behaving relays and the group of other users and relays. We review related works in Sect. 7, and conclude in Sect. 8.

2 Background

The Tor network is an overlay network of volunteers running *Tor relays* that relay TCP streams for *Tor clients*. Tor lets its users connect to Internet destinations like websites while making it hard for 1) an attacker on the client side to learn the intended destination, 2) an attacker on the destination side to learn the client's location, and 3) any small group of relays to link the client to her destinations.

To connect to a destination via Tor, the client software incrementally creates a private pathway or *circuit* of encrypted connections through several Tor relays, negotiating a separate set of encryption keys for each hop along the circuit. The circuit is extended one hop at a time, and each relay along the way knows only the immediately previous and following relay in the circuit, so no single Tor relay knows the complete path that each fixed-sized data packet (or *cell*) will take. Thus, neither an eavesdropper nor a compromised relay can see both the connection’s source and destination. Clients periodically rotate to a new circuit to complicate long-term linkability between different actions by a single user.

The client learns which relays it can use by fetching a signed list of Tor relays from the *directory authorities*. Each authority lists the available relays along with opinions on whether each relay is considered reliable, fast, and so on. Clients base their decisions on the consensus (i.e. the majority of authority opinions). Each authority’s signing key comes with the Tor software, so Tor clients can’t be tricked into using an alternate network run by an attacker. Authorities also provide a way for Tor users to synchronize their behavior; since anonymity loves company, users that make decisions based on similar information will blend together better [15]. A more detailed description of the Tor design can be found in its original design document [16] and its specifications [14].

Anonymity designs can be divided into two classes based on their goals: *high-latency* and *low-latency*. High-latency designs like Mixmaster [34] and Mixminion [11] can take hours to deliver messages, but because messages mix with each other they can withstand quite powerful attackers. These designs are not suitable for web surfing, which would be untenable with long latencies.

Tor chooses to build a practical and useful network, then try to achieve good security within these constraints. To that end, Tor doesn’t batch or reorder messages at each hop. This choice means that Tor circuits are vulnerable to *end-to-end correlation attacks*: an attacker who can measure traffic at both ends of the circuit can link them [10,28]. A variety of other anonymity-breaking attacks become possible because of Tor’s requirement to remain useful for low-latency communications [26,29,35,36,41,44].

Because Tor aims to resist *traffic analysis* attacks (attacks that try to pick the communicants out of a large set of participants) but does not aim to protect against correlation attacks (attacks that watch two suspected endpoints to confirm the link), we have some flexibility in what design changes we can propose. As long as we don’t introduce any attacks that are worse than the correlation attacks, we are still within Tor’s threat model.

3 Incentive Goals

Relayed traffic is traffic forwarded from a Tor client or Tor relay to another relay within the network. Choosing to relay traffic can provide better anonymity in some cases: an attacker who controls the user’s next hop would not be able to know whether the connection originated at the user or was relayed from somebody else. But the exact details of the potential anonymity improvement

are not well-understood even by the research community. Therefore they are hard to communicate to users, so any potential perceived gains do not outweigh the costs of setting up relaying and providing bandwidth to others.

Tor relays may also opt to serve as exit relays. *Exit traffic* is forwarded from a Tor relay to somewhere outside the Tor network, as well as return traffic from outside back into Tor. While there are theoretical anonymity improvements similar to those for relaying traffic, as well as potential legal advantages for the relay operator from not necessarily being the originator of all traffic coming from the relay's IP address [20], in practice the destination website and the user's ISP have no idea that Tor exists, and so they assume all connections are from the operator. Some ISPs tolerate abuse complaints better than others. This hassle and legal uncertainty may drive users away from running as an exit relay.

Beyond creating incentives to relay traffic inside the Tor network and to allow connections to external services, we also need to consider the *quality* of the traffic (e.g., the latency and throughput provided, and the reliability and consistency of these properties). Since Tor circuits pass over several relays, the slowest relay in the circuit has the largest impact.

4 Design

Our solution is to give a “gold star” in the directory listing to relays that provide good service to others. A gold star relay's traffic is given higher priority by other relays, i.e., they always get relayed ahead of other traffic. Furthermore, when a gold star relay receives a high priority connection from another gold star relay, it passes on the gold star status so the connection remains high priority on the next hop. All other traffic gets low priority. If a low priority node relays data through a gold star relay, the traffic is still relayed but at low priority. Traffic priority is circuit-based. Once a circuit is created, its priority remains the same during its entire lifetime.

We can leverage Tor's existing directory authorities to actively measure the performance of each individual relay [42] and only grant those with satisfactory performance the gold star status. This measurement can include bandwidth and latency of the relayed traffic for that relay. By measuring the bandwidth through the Tor network itself, the directory authorities can hide their identity and intent from the Tor relays. This method of anonymously auditing nodes' behavior is similarly used in other systems [19,38,45].

Due to variations of the network conditions and the multi-hop nature of Tor, it may take multiple measurements to get accurate results. Therefore, we use a “ k out of n ” approach, where a relay has to have satisfactory performance for k times out of the last n measurements to be eligible for gold star status. At that point, it becomes a policy issue of who gets a gold star. We assign a gold star to the fastest 7/8 of the nodes, following the current Tor design in which the slowest one-eighth of Tor relays are not used to relay traffic at all. Of course, relays may choose not to give priority to gold star traffic. But in this case, they would most likely become the slowest nodes in the measurements and would

not earn a gold star. The directory authorities can then distribute the gold star status labels along with the network information they presently distribute.

The behaviors we most need to encourage will vary based on the demands facing the Tor network. Since there are enough exit relays currently, the design and analysis in this paper focuses on incentives for relayed traffic. However, our approach of giving priority to traffic from the *most useful* relays means that we can adapt the definition of “useful” as needed: e.g. we could vary the required threshold in the measurement tests above or require new tests such as verifying that exit traffic is handled correctly. We would then only need to publish the desired policy; users desiring higher priority for their traffic would then decide whether to follow the policy. We consider exit traffic testing more in Section 6.2.

The effectiveness of this design depends on the accuracy of the measurements, which in turn depends on the measurement frequency. Frequent measurements increase our confidence, but they also place a burden on the network and limit the scalability of the measuring nodes. Snader and Borisov [46] suggest an alternate approach to learning bandwidth, where every relay reports its own observations about the other relays, and the directory authorities use the median vote. If we used this approach, gold stars could then be assigned based on having a high enough median vote; but we note that active measurements would still be needed for verifying other properties such as exit traffic.

5 Experiments

Here we show simulations of the effectiveness of our “gold star” incentive scheme against different scenarios, including varying amounts of load on the Tor network, and varying strategies taken by simulated nodes (e.g., selfish vs. cooperative).

5.1 Experimental Apparatus

We built a packet-level discrete event simulator that models a Tor overlay network. The simulator, written in Java, was executed on 64-bit AMD Opteron 252 dual processor servers with 4GB of RAM and running RedHat Enterprise Linux (kernel version 2.6.9) and Sun’s JVM, version 1.5.0.

We simulate every cell at every hop. Each node, particularly simulated Bit-Torrent clients, can easily have hundreds of outstanding cells in the network at any particular time. Simulating 20 BitTorrent clients and 2000 web clients consumes most of the available memory. To keep the client-to-relay ratio realistic, we could only simulate Tor networks with around 150 relays.

For simplicity, we assumed the upstream and downstream bandwidth for all relays is symmetric, since the forwarding rate of any relay with asymmetric bandwidth will be limited by its lower upstream throughput. We also simplify relays by assuming they take no processing time. The cooperative relays (which reflect the altruists in the current Tor network) have a bandwidth of 500KB/s. The latency between any two nodes in the network is fixed at 100 ms.

Our simulations use different numbers of simplified web and BitTorrent clients to generate background traffic. Our web traffic is based on Hernández-Campos

et al. [25]’s “Data Set 4,” collected in April 2003 [48]. Our simplified BitTorrent clients always maintain four connections, and upload and download data at the maximum speed Tor allows. They also periodically replace their slowest connection with a new one, following BitTorrent’s standard policy. We assume that the external web or BitTorrent servers have unlimited bandwidth. The different relay traffic types are:

Cooperative. These nodes use their entire 500KB/s bandwidth to satisfy the needs of their peers, and give priority to “gold star” traffic when present. (If sufficient gold star traffic is available to fill the entire pipe, regular traffic will be completely starved for service.)

Selfish. These nodes *never* relay traffic for others. They are freeloaders on the Tor system with 500KB/s of bandwidth.

Cooperative slow. These nodes follow the same policy as cooperative nodes, but with only 50KB/s of bandwidth.

Cooperative reserve. These nodes have 500KB/s bandwidth, just like cooperative nodes, but cap their relaying at 50KB/s, saving the remainder for traffic that they originate.

Adaptive. These nodes are cooperative until they get a gold star. After this, they are selfish until they lose the gold star.

All of our simulations use ten directory authorities. To assign the gold star status, every minute each directory authority randomly builds a circuit with three Tor relays and measures its bandwidth by downloading a small, known 40KB file from an external server. The bandwidth measurement is recorded and attributed to only the middle relay in the circuit. (In a genuine deployment, the entry and exit nodes would be able to determine that they were being measured by virtue of being connected to known measurement nodes and could thus change their behavior in response.) To obtain a gold star, we require Tor relays to successfully relay traffic at least two times out of the last five measurements (i.e. $k = 2$ and $n = 5$ from Section 4). A relay is defined as successful if the directory authority can receive the correct file within a reasonable amount of time.

For our results, we describe the observed network performance in terms of “download time” and “ping time.” Download time describes the time for nodes to download a 100KB file from an external server. Ping time describes the round-trip latency for that same external server. (This external server is assumed to have infinite bandwidth and introduce zero latency of its own.) Both measures are important indicators of how a Tor user might perceive Tor’s quality when web surfing. For contrast, a Tor user downloading large files will be largely insensitive to ping times, caring only about throughput.

5.2 Experiment 1: Unincentivized Tor

First, we want to understand how Tor networks behave when demand for the network’s resources exceeds its supply. We simulated 50 cooperative relays, 50 selfish relays, and 50 cooperative reserve relays with heavy background traffic (20 BitTorrent clients and 2000 web clients).

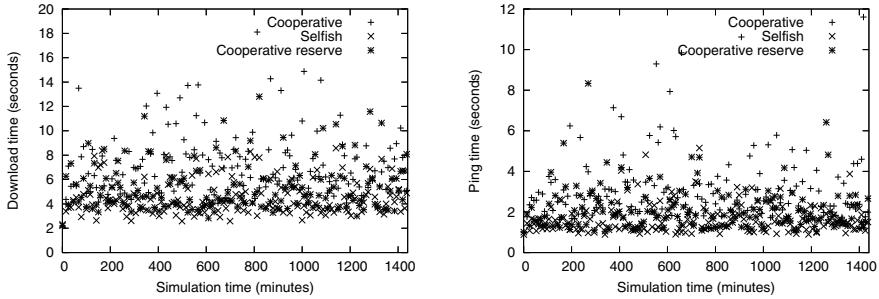


Fig. 1. Average download and ping time over time when no incentive scheme is in place and heavy traffic (20 BitTorrent clients and 2000 web clients). Both download and ping time show significant variation, regardless of relay type.

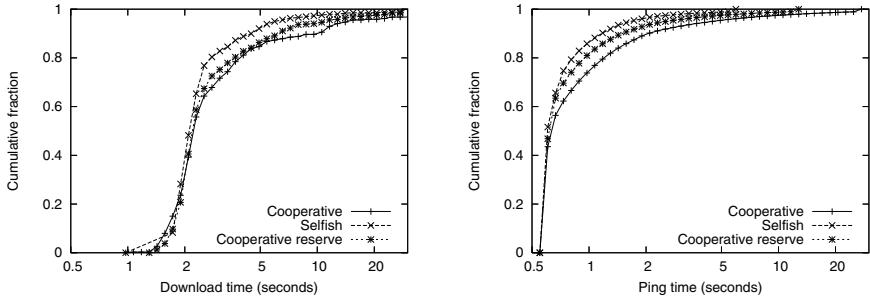


Fig. 2. Cumulative download and ping time when no incentive scheme is in place and heavy traffic (20 BitTorrent clients and 2000 web clients). Performance for all relay types is similar, although selfish relays do somewhat better in the worst case.

Figure 1 plots the *average* download and ping time for each relay type. Each plotted point is the average for 50 raw samples. Despite this, the variation among the data points suggests that the network performance is highly variable.

To get a better view of the distribution of download times and ping times, we use cumulative distribution functions (CDFs). Figure 2 represents the same data as Fig. 1, albeit without any of the data-point averaging. The *x*-axis represents download time or ping time and the *y*-axis represents the percentage of nodes who experienced that particular download or ping time *or less*.

While the ideal download time for all relay types in this experiment is 0.8 seconds (six network roundtrip hops plus transmission time), all relay types rarely achieve anywhere close to this number. Figure 2 shows that roughly 80% of the attempted downloads take more than two seconds, regardless of a node's policy. Approximately 10% of cooperative relays take longer than ten seconds. Selfish nodes, in general, do better in the worst case than cooperative nodes, but observe similar common-case performance.

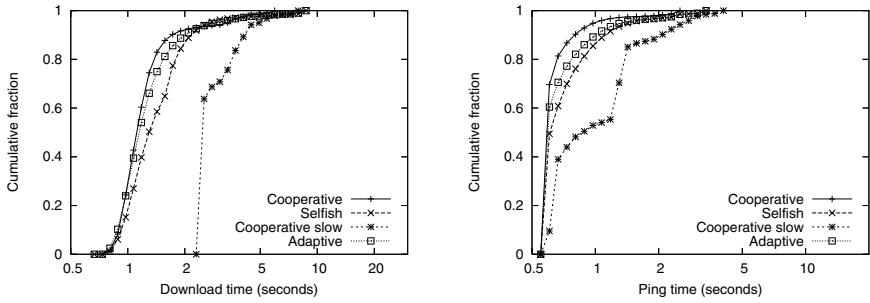


Fig. 3. Cumulative download and ping time with the gold star scheme and no background traffic.

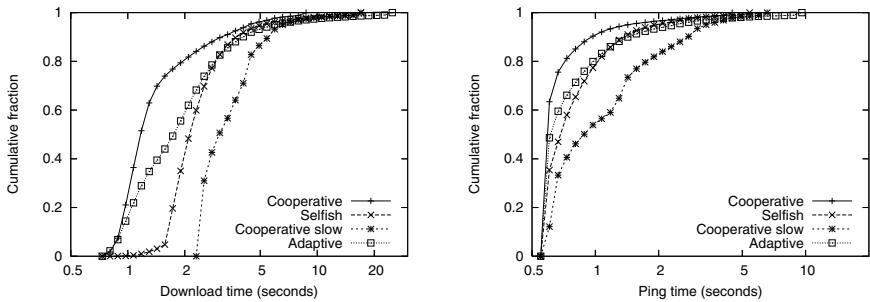


Fig. 4. Cumulative download and ping time with the gold star scheme and heavy background traffic (20 BitTorrent clients and 2000 web clients). Cooperative nodes maintain their performance, while selfish and adaptive nodes suffer.

5.3 Experiment 2: Gold Stars

Our first experiment represents the present-day situation in the Tor network and is clearly unsatisfactory. This second experiment measures the effectiveness of our “gold star” mechanism in addressing this concern. This time, our simulation consists of 40 cooperative relays, 40 selfish relays, 40 cooperative slow relays, and 40 adaptive relays. These variations, relative to the first experiment, also allow us to see whether slower cooperative nodes still get the benefits of a gold star, and whether adaptive nodes can be more effective than purely selfish nodes. Figures 3 and 4 show the cumulative download and ping time with no background traffic and heavy background traffic, respectively.

Our results are striking. Cooperative nodes maintain their performance, regardless of the level of background traffic in the overlay. When there is no background traffic, they slightly outperform the selfish and adaptive nodes, but once the traffic grows, the cooperative nodes see clear improvements. For example, under heavy background traffic, 80% of the cooperative nodes see download times under two seconds, versus roughly 2.5 seconds for the selfish and adaptive nodes.

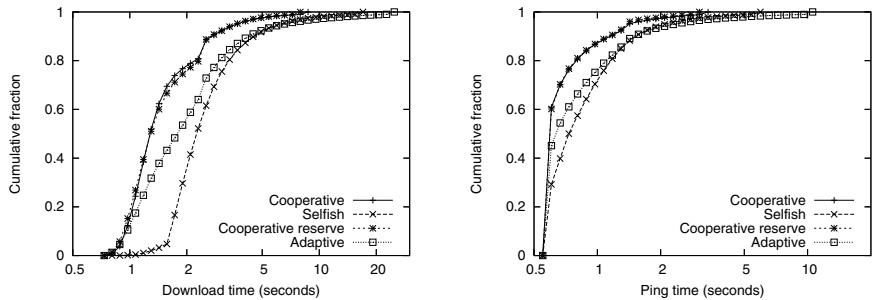


Fig. 5. Cumulative download and ping time with the gold star scheme and heavy background traffic (20 BitTorrent clients and 2000 web clients). Cooperative reserve relays, which replaced the cooperative slow relays, have similar performance to fully cooperative relays.

Our experiment also shows that the adaptive policy does not defeat the gold star mechanism. Adaptive nodes will experience better performance while they have a gold star, but their benefit only splits the difference between the cooperative and selfish policies, roughly in proportion to the effort they are spending to maintain their gold star.

Cooperative slow nodes are always relatively slow due to their limited available bandwidth. However, like their fast counterparts, they experience stable performance as the background load on the Tor network increases. This demonstrates that the gold star policy can effectively reward good behavior, regardless of a node’s available bandwidth.

We conducted a further experiment, replacing the cooperative slow nodes with cooperative reserve nodes, representing a possible rational response to the gold star mechanism. These nodes use 10% of their bandwidth for relaying and earning a gold star, reserving 90% of their bandwidth for their own needs. Figure 5 shows the results of this experiment. Both kinds of cooperative nodes observe identical distributions of bandwidth and latency. Selfish and adaptive nodes suffer as the background traffic increases. This experiment shows, unsurprisingly, that nodes need not be “fully” cooperative to gain a gold star. In an actual Tor deployment, it would become a policy matter, perhaps an adaptive process based on measuring the Tor network, to determine a suitable cutoff for granting gold stars (see Sect. 6.1).

5.4 Experiment 3: Alternating Relays

This experiment considers a variation on the adaptive strategy used in the previous experiments. Alternating nodes will toggle between the cooperative and the selfish strategies on a longer timescale—four hours per switch. This experiment uses 50 such alternating relays with 50 cooperative relays and with heavy background traffic (20 BitTorrent clients and 2000 web clients).

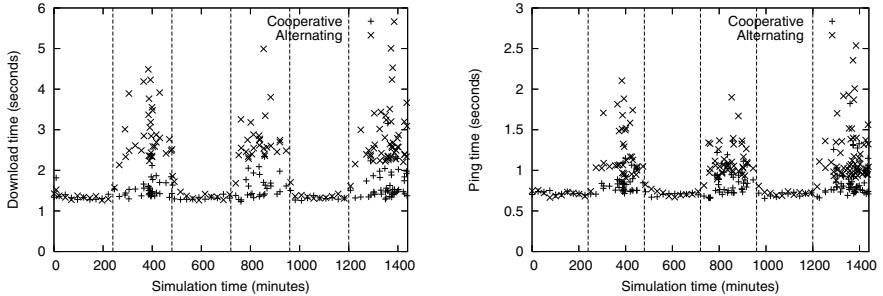


Fig. 6. Average download and ping time with relays that alternate between being cooperative and selfish. This experiment is with the gold star scheme in place and heavy background traffic (20 BitTorrent clients and 2000 web clients). Dotted lines show the times at which the alternating relays switch. The performance of alternating relays gets worse whenever they switch to being selfish, while performance for cooperative relays only suffers a little.

Figure 6 shows the average download and ping time for both relay types over time. During the periods where the alternating relays are cooperative, they receive service of a similar quality as the full-time cooperative nodes. However, once the alternating relays switch to become selfish, their download times quickly increase, representing the same quality of service that would be observed by a selfish node. Note that while the cooperative nodes do observe lower quality of service (after all, fully half of the Tor nodes stopped relaying any data), they still do much better than their selfish peers. Our gold star system robustly responds to changes in node behavior.

5.5 Experiment 4: Pair-Wise Reputation

Last, we investigated a variation on our gold star design where individual circuits are not labelled as being low or high priority. Rather, each circuit inherits its priority from the status of the previous relay. That is, a low-priority node routing traffic through a gold-star node will experience delays getting the gold-star node to accept the traffic, but the traffic will have high priority in its subsequent hops. This alternative design has significant improvements from an anonymity perspective, because traffic at a given hop does not give any hint about whether it originated from a low-priority or high-priority node. However, our experiment showed selfish nodes clearly outperforming their cooperative peers. The results are shown in Appendix A.

6 Discussion

Our experiments show that our “gold star” technique is effective at giving higher priority to users who contribute to the Tor network. Nonetheless, a variety of

questions remain about the policy that should be used for assigning gold stars and how the policy may affect the behavior of strategic Tor users.

6.1 Strategic Users

Our proposed incentive scheme is not perfectly strategy-proof, in the sense that users can earn a gold star without providing *all* of their network capacity for the use of the Tor network (as in the “cooperative reserve” policy discussed in Sect. 5.3). This balance creates a variety of possible strategic behaviors.

Provide borderline or spotty service. A relay needs to provide only the minimal amount of bandwidth necessary to gain the gold star. Of course, if every user provided this amount, Tor would still have vastly greater resources than it does today. Next, because the bandwidth policies are determined centrally, the required minimum bandwidth for obtaining a gold star could be adjusted in response to network congestion. Strategic nodes would then adjust the capacity they offer, making more bandwidth available whenever they are needed.

Only relay at strategic times. Strategic users might provide relay services only when the “local” user is away, and thus not making demands on the Tor network. Such behavior is not disincentivized by our approach, as it still provides scalable resources to the Tor network. However, any users following such behavior may be partially compromising their anonymity, as their presence or absence will be externally observable.

Share a relay among several users. Several users could share a single entry relay into the Tor network, thus inheriting its gold star benefits without providing any additional bandwidth to the Tor network. In fact, we may even want to support this design, so users can run a fast relay at a colocation facility and then reap the rewards from their slower cable-modem or DSL Tor client. To allow the client to inherit the reputation of the server, the relay could be configured to give high priority to connections from a given set of IP addresses or Tor identity keys. On the other hand, multiple users that use a shared entry point must be able to trust one another. Lacking such trust, their desire for personal anonymity would incentivize them to run individual Tor relays.

Accept traffic only from known relays. In our design the directory authorities do their measurements anonymously via Tor, so all audits will come from other listed Tor relays. Thus a strategic relay could get away with giving poor performance (or no performance at all!) to connections from IP addresses not listed in the directory. One answer is that some of the measurements should be done through unlisted relays, perhaps by gathering a large pool of volunteer Tor users to help diversify the audit sources. Another answer is to turn this vulnerability around and call it a feature—another reason that users should want to get listed as a good relay.

Forward high-priority traffic as low-priority. A relay who correctly forwards traffic can still cheat by changing the priority on incoming traffic. The

measuring authorities should build high-priority test circuits back to a trusted relay, to see if the circuit arrives with the expected high-priority status.

6.2 The Audit Arms Race

Some attacks outlined above involve relays that provide some level of service but not quite as much as we might prefer. The response in each case is a smarter or more intensive measurement algorithm so the directory authorities can more precisely distinguish uncooperative behavior.

To see why this won't be an arms race between increasingly subtle cheating and increasingly sophisticated audits, we need to examine the incentives for ordinary users. Based on informal discussions with Tor relay operators, the most challenging part of setting up a Tor relay is configuring the software, enabling port forwarding in the firewall, etc. Compared to this initial barrier, the incremental cost of providing a bit more bandwidth is low for most users. As long as our audit mechanism correctly judges whether the user relays any traffic at all, we're verifying that the user has performed the most personally costly step in setting up a relay. We expect that the diminishing returns a strategic relay gets in saving bandwidth as we progress down the arms race will limit the complexity required for the auditing mechanism.

Measuring whether a relay is forwarding traffic adequately within the network is only one step. We could also extend our auditing techniques to measure whether an exit relay is in fact correctly forwarding exit traffic. We could thus incentivize exit traffic in the same way we incentivize relay traffic.

One concern in any measurement scheme over Tor is that the anonymity of Tor hides which node in an overlay route may have been responsible for degrading the quality of service. We could potentially “charge” all of the nodes in the route, but this could lead to “collateral reputation damage” for innocent nodes. An adversary may even strategically target a node for such damage. This ability to influence reputation can assist in anonymity-breaking attacks [6,19].

6.3 Anonymity Implications

Anonymity metrics like entropy [12,43] apply to high-latency systems like Mixminion, where a global attacker aims to narrow the set of suspects to a small anonymity set (or to a probability distribution that has low entropy). With low-latency systems like Tor, most attacks either fail or reduce the entropy to zero. Thus these systems instead measure their security by the probability that a non-global attacker will be in the right positions in the network to launch an attack [47]. One key factor in this metric is the number of relays in the network.

Assuming the Tor network starts out with a small number of gold star relays, whenever a Tor relay receives a high priority cell, it knows with absolute certainty that the cell must have originated from a relay having a gold star. With so few gold star relays, the presence of high priority traffic greatly reduces the number of possible sources for that traffic. Worse, the set of users with a gold star is made public, further simplifying the attack.

We believe this tradeoff would still attract many gold star relays, though. First, altruists who don’t use Tor as a client would still be the early adopters, as predicted by Acquisti et al. [1] and as observed in the current Tor network. Low-sensitivity users would come next; many users who care more about performance than anonymity would be enticed into running Tor relays. The number of gold star nodes in the system should therefore increase over time, reducing the extent to which the presence of prioritized traffic gives away information to an attacker. We speculate that the growing anonymity set of gold star relays, along with the improved performance from being in the group getting priority traffic, would ultimately be enough to push many mainstream users into setting up relays.

We leave one major obstacle for future work: even if the anonymity set of gold-star users is large, changes in the set over time allow *intersection attacks* on anonymity. That is, if an attacker can link connections by a gold-star user (for example by tracking cookies or logins to a website), this attacker can narrow down which relay is running and has a gold star whenever such a connection occurs. One solution might be to make the gold star status persist a while after the relay stops offering service, to dampen out fluctuations in the anonymity sets. Depending on the rate of churn in the Tor network, this period might need to be a week or more, which means we then need to reevaluate the balance questions from this section. A more radical change would be to move to an ecash based service where getting high priority is less related to whether you’re running a good relay at the time [2] – but we note that even in these more complex designs where there are multiple plausible reasons for a given user to get higher priority, users that earn their priority by relaying traffic can still be attacked [33].

6.4 The Economics of Attracting More Relays

The experiments in Section 5 show that our design creates significant incentives for users to run Tor relays. As we attract more relays, the Tor network grows larger. Thus the anonymity that can be achieved increases for both the relays and the clients. As we attract more relays, the overall capacity in the network grows too. In fact, if we get a large enough network, the performance will improve compared to the currently deployed Tor network not only for the users who choose to run relays, but also for the users who don’t!

If enough users do choose to run relays that there is excess network capacity, then the observable performance difference between high priority traffic and regular traffic might be insufficient to get more relays, or even to keep all of the current relays. If such a problem were to occur, one additional possibility would be to reserve bandwidth for high-priority traffic [40], effectively throttling low-priority traffic and creating a larger incentive for users to get a gold star. The downside to such an approach, of course, is that Tor performance would “needlessly” suffer for low-priority Tor users.

This discussion of the economics of our incentive strategy leaves out many details. We should start by analyzing the various equilibria and deriving utility functions for various user classes. We leave this investigation to future work.

7 Related Work

7.1 Incentives in Anonymous Communication Networks

Real-world anonymizing networks have operated on three incentive approaches: *community support*, *payment for service*, and *government support*. (Discussion of the funding approaches for research and development of anonymity designs, while related, is outside the scope of this paper.) The Tor network right now is built on community support: a group of volunteers from around the Internet donate their resources because they want the network to exist.

Zero-Knowledge Systems' Freedom network [7] on the other hand was a commercial anonymity service. They collected money from their users, and paid commercial ISPs to relay traffic. While that particular company failed to make its business model work, the more modest Anonymizer [3] successfully operates a commercial one-hop proxy based on a similar approach. PAR [2] proposes a micropayment model where clients pay coins for each circuit, and relays can use these coins for service of their own or convert them into actual payments; however, its dependency on an ecash bank means it remains a theoretical design. Lastly, the AN.ON project's cascade-based network was directly funded by the German government as part of a research project [4]. Unfortunately, the funding ended in 2007, so they are exploring the community support approach (several of their nodes are now operated by other universities) and the pay-for-play approach (setting up commercial cascades that provide more reliable service).

Other incentive approaches have been discussed as well. Acquisti et al. [1] argued that high-needs users (people who place a high value on their anonymity) will opt to relay traffic in order to attract low-needs users — and that some level of free riding is actually beneficial because it provides cover traffic. It is unclear how well that argument transitions from the high-latency systems analyzed in that paper to low-latency systems like Tor.

7.2 Incentives in Other Peer-to-Peer Networks

Incentives for applications. Incentive schemes have been proposed for several p2p applications. BitTorrent [8], one of the pioneers, facilitates large numbers of nodes sharing the effort of downloading very large files. Every node will have acquired some subset of the file and will trade blocks with other nodes until it has the rest. Nodes will preferentially trade blocks with peers that give them better service (“tit-for-tat” trading). Scrivener [37] addresses a more general problem, where nodes are interested in sharing a larger set of smaller files.

In a storage network, nodes share spare disk capacity for applications such as distributed backup systems. Ngan et al. [38] proposed an auditing mechanism, allowing cheaters to be discovered and evicted from the system. Samsara [9] enforced fairness by requiring an equal exchange of storage space between peers and by challenging peers periodically to prove that they are actually storing the data. Tangler [50] required users to provide resources for a probation period before they are allowed to consume resources.

Reputation systems. Resource allocation and accountability problems are fundamental to p2p systems. Dingledine et al. [13] survey many schemes for tracking nodes’ reputations. In particular, if obtaining a new identity is cheap and positive reputations have value, negative reputation could be shed easily by leaving the system and rejoining with a new identity. Friedman and Resnick [21] also study the case of cheap pseudonyms, and argue that suspicion of strangers is costly. EigenTrust [27] is a distributed algorithm for nodes to securely compute global trust values based on their past performance. Blanc et al. [5] suggest a reputation system for incentivizing routing in peer-to-peer networks that uses a trusted authority to manage the reputation values for all peers, comparable to our use of directory authorities.

Trading and payments. SHARP [22] is a framework for distributed resource management, where users trade resources with trusted peers. KARMA [49] and SeAl [39] rely on auditor sets to track the resource usage of each node in the network. Golle et al. [23] considered p2p systems with micro-payments, analyzing how various user strategies reach equilibrium within a game theoretic model.

Tradeoff between anonymity and performance. If the number of gold star relays in the network is small, sending gold star traffic may result in reduced anonymity, albeit better performance. This introduces another dimension of traffic control. In our design a gold star relay is not required to send its own traffic at high priority; it may choose to send it at a low priority for better anonymity. This tradeoff is similar to the idea in Alpha-mixing [18], where the sender can use a parameter to choose between better anonymity and lower latency.

8 Conclusions

This paper proposes an incentive scheme to reward Tor users who relay traffic. Our simulations show that we can correctly identify nodes who cooperate with our desired policies, and they achieve sizable performance improvements, particularly in Tor’s current situation where the Tor network is saturated with traffic. While we reduce anonymity for cooperative (“gold star”) nodes because any high priority traffic must have originated from a gold star node, we create significant performance incentives for many users to join the Tor network as relays, which improves both performance *and* anonymity.

Once our technique is ready to be deployed on the live Tor network, both pragmatic and theoretical concerns remain. For example, we cannot predict future demand on the Tor network, nor can we predict the extent to which firewalls or ISP bandwidth policies might interfere with Tor or otherwise disincentivize users from relaying Tor traffic. We should also investigate the extent to which the centralized Tor management nodes might be able to coordinate their network measurements and agree on optimal incentivization policies as network conditions evolve.

References

1. Acquisti, A., Dingledine, R., Syverson, P.: On the economics of anonymity. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 84–102. Springer, Heidelberg (2003)
2. Androulaki, E., Raykova, M., Srivatsan, S., Stavrou, A., Bellovin, S.M.: PAR: Payment for anonymous routing. In: Borisov, N., Goldberg, I. (eds.) PETs 2008. LNCS, vol. 5134, pp. 219–236. Springer, Heidelberg (2008)
3. The Anonymizer, <http://www.anonymizer.com/>
4. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, p. 115. Springer, Heidelberg (2001)
5. Blanc, A., Liu, Y.-K., Vahdat, A.: Designing incentives for peer-to-peer routing. In: Proceedings of the 24th IEEE INFOCOM, Miami, FL (March 2005)
6. Borisov, N., Danezis, G., Mittal, P., Tabriz, P.: Denial of service or denial of security? How attacks on reliability can compromise anonymity. In: Proceedings of CCS 2007 (October 2007)
7. Boucher, P., Shostack, A., Goldberg, I.: Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc. (December 2000), http://osiris.978.org/~brianr/crypto-research/anon/www.freedom.net/products/whitepapers/Freedom_System_2_Architecture.pdf
8. Cohen, B.: Incentives build robustness in BitTorrent. In: Proceedings of the Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA (June 2003)
9. Cox, L.P., Noble, B.D.: Samsara: Honor among thieves in peer-to-peer storage. In: Proc. 19th ACM Symp. on Operating System Principles (SOSP 2003), Bolton Landing, NY (October 2003)
10. Danezis, G.: The traffic analysis of continuous-time mixes. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 35–50. Springer, Heidelberg (2005)
11. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a type III anonymous remailer protocol. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA (May 2003)
12. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)
13. Dingledine, R., Freedman, M.J., Molnar, D.: Accountability measures for peer-to-peer systems. In: Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly and Associates, Sebastopol (November 2000)
14. Dingledine, R., Mathewson, N.: Tor protocol specification, <https://www.torproject.org/svn/trunk/doc/spec/tor-spec.txt>
15. Dingledine, R., Mathewson, N.: Anonymity loves company: Usability and the network effect. In: Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006), Cambridge, UK (June 2006)
16. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of 13th USENIX Security Symposium, San Diego, CA (August 2004), Project web site, <https://www.torproject.org/>
17. Dingledine, R., Mathewson, N., Syverson, P.: Challenges in deploying low-latency anonymity. Technical Report 5540-265, Center for High Assurance Computer Systems, Naval Research Laboratory (2005)
18. Dingledine, R., Serjantov, A., Syverson, P.: Blending different latency traffic with alpha-mixing. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 245–257. Springer, Heidelberg (2006)

19. Dingledine, R., Syverson, P.: Reliable MIX cascade networks through reputation. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 253–268. Springer, Heidelberg (2003)
20. Electronic Frontier Foundation. Tor: Legal FAQ for Tor server operators, <https://www.torproject.org/eff/tor-legal-faq.html>
21. Friedman, E., Resnick, P.: The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy* 10(2), 173–199 (2001)
22. Fu, Y., Chase, J.S., Chun, B.N., Schwab, S., Vahdat, A.: SHARP: An architecture for secure resource peering. In: Proc. 19th ACM Symp. on Operating System Principles (SOSP 2003), Bolton Landing, NY (October 2003)
23. Golle, P., Leyton-Brown, K., Mironov, I., Lillibridge, M.: Incentives for sharing in peer-to-peer networks. In: Proceedings of the 3rd ACM Conference on Electronic Commerce, Tampa, FL (October 2001)
24. Hardin, G.: The tragedy of the commons. *Science* 162 (1968), Alternate location, <http://dieoff.com/page95.htm>
25. Hernández-Campos, F., Jeffay, K., Smith, F.D.: Tracking the evolution of web traffic: 1995–2003. In: Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Orlando, FL (October 2003)
26. Hopper, N., Vasserman, E.Y., Chan-Tin, E.: How much anonymity does network latency leak? In: Proceedings of the 14th ACM Conference on Computer and Communication Security, Alexandria, VA (October 2007)
27. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary (May 2003)
28. Levine, B.N., Reiter, M.K., Wang, C., Wright, M.K.: Timing attacks in low-latency mix-based systems. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 251–265. Springer, Heidelberg (2004)
29. Liberatore, M., Levine, B.N.: Inferring the Source of Encrypted HTTP Connections. In: Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, VA, pp. 255–263 (October 2006)
30. Loesing, K.: Evaluation of client requests to the directories to determine total numbers and countries of users. Technical report, The Tor Project (June 2009), <https://torproject.org/projects/metrics>
31. Loesing, K.: Measuring the Tor network from public directory information. Technical report, 2nd Hot Topics in Privacy Enhancing Technologies (HotPETs 2009), Seattle, WA, USA (August 2009)
32. McCoy, D., Bauer, K., Grunwald, D., Kohno, T., Sicker, D.: Shining light in dark places: Understanding the Tor network. In: Borisov, N., Goldberg, I. (eds.) PETs 2008. LNCS, vol. 5134, pp. 63–76. Springer, Heidelberg (2008)
33. McLachlan, J., Hopper, N.: On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2009). ACM, New York (November 2009)
34. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster protocol — version 2. IETF Internet Draft (July 2003), <http://www.abditum.com/mixmaster-spec.txt>
35. Murdoch, S.J., Danezis, G.: Low-cost traffic analysis of Tor. In: Proc. IEEE Symposium on Security and Privacy, Oakland, CA (May 2005)
36. Murdoch, S.J., Zieliński, P.: Sampled traffic analysis by Internet-exchange-level adversaries. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 167–183. Springer, Heidelberg (2007)

37. Nandi, A., Ngan, T.-W.J., Singh, A., Druschel, P., Wallach, D.S.: Scrivener: Providing incentives in cooperative content distribution systems. In: Alonso, G. (ed.) *Middleware 2005*. LNCS, vol. 3790, pp. 270–291. Springer, Heidelberg (2005)
38. Ngan, T.-W.J., Wallach, D.S., Druschel, P.: Enforcing fair sharing of peer-to-peer resources. In: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), Berkeley, CA (February 2003)
39. Ntarmos, N., Triantafillou, P.: SeAl: Managing accesses and data in peer-to-peer sharing networks. In: Proceedings of the 4th IEEE International Conference on P2P Computing, Zurich, Switzerland (August 2004)
40. Odlyzko, A.M.: Paris metro pricing for the Internet. In: ACM Conference on Electronic Commerce, pp. 140–147 (1999)
41. Øverlier, L., Syverson, P.: Locating hidden servers. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA (May 2006)
42. Perry, M.: TorFlow: Tor Network Analysis. Technical report, 2nd Hot Topics in Privacy Enhancing Technologies (HotPETs 2009), Seattle, WA, USA (August 2009)
43. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 41–53. Springer, Heidelberg (2003)
44. Shmatikov, V., Wang, M.-H.: Timing analysis in low-latency mix networks: Attacks and defenses. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 18–33. Springer, Heidelberg (2006)
45. Singh, A., Ngan, T.-W.J., Druschel, P., Wallach, D.S.: Eclipse attacks on overlay networks: Threats and defenses. In: Proceedings of IEEE INFOCOM, Barcelona, Spain (April 2006)
46. Snader, R., Borisov, N.: A tune-up for Tor: Improving security and performance in the Tor network. In: Proceedings of the Network and Distributed Security Symposium - NDSS 2008. Internet Society, San Diego (February 2008)
47. Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an Analysis of Onion Routing Security. In: Federrath, H. (ed.) *Designing Privacy Enhancing Technologies*. LNCS, vol. 2009, pp. 96–114. Springer, Heidelberg (2001)
48. The Distributed and Real-Time Systems Research Group, UNC. Data for the UNC HTTP traffic model, <http://www.cs.unc.edu/Research/dirt/proj/http-model/>
49. Vishnumurthy, V., Chandrakumar, S., Sirer, E.G.: KARMA: A secure economic framework for p2p resource sharing. In: Proceedings of the Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA (June 2003)
50. Waldman, M., Mazières, D.: Tangler: A censorship resistant publishing system based on document entanglements. In: Proceedings of the 8th ACM Conference on Computer and Communication Security (CCS 2001), Philadelphia, Pennsylvania (November 2001)

A Experiment 4: Pair-Wise Reputation

In this experiment, we investigated a variation on our gold star design, where individual circuits are not labelled as being low or high priority. In this variation, a low-priority node routing traffic through a gold-star node will experience delays getting the gold-star node to accept the traffic, but the traffic will have the gold-star priority in its subsequent hops. This alternative design has significant improvements from an anonymity perspective, because traffic at a given hop doesn’t give any hint about whether it originated from a low-priority or high-priority node. However, this design might fail from an incentives perspective, since there is less incentive for a node to earn its own gold star.

In this experiment, we again simulate a network with 40 relays for each relay type: cooperative, selfish, cooperative reserve, and adaptive. For clarity, Fig. 7 only shows the download and ping time for cooperative and selfish relays, as the performance for cooperative reserve and adaptive relays is very close to that for cooperative relays.

This experiment shows selfish nodes clearly outperforming their cooperative peers. This indicates that the gold star strategy requires a transitive property, i.e., each hop of a circuit must inherit the gold star status of the previous hop. Otherwise, selfish nodes will outperform their cooperative peers and there will be no incentive for cooperation.

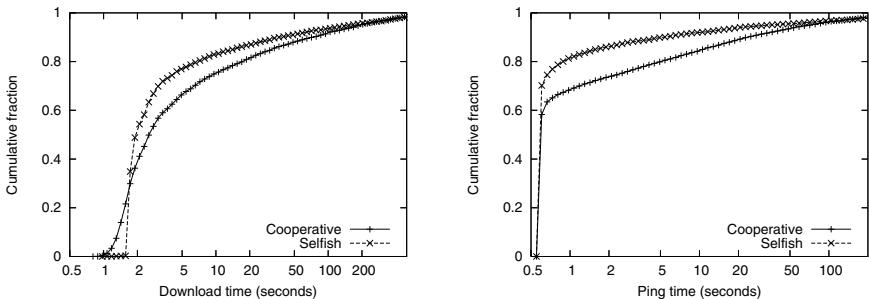


Fig. 7. Cumulative download and ping time with the pair-wise reputation design and heavy traffic (20 BitTorrent clients and 2000 web clients). Four relay types (cooperative, selfish, cooperative reserve, and adaptive) are simulated, although only the performance of the former two are shown, as the latter two behave similarly to cooperative relays.

Tree-Homomorphic Encryption and Scalable Hierarchical Secret-Ballot Elections

Aggelos Kiayias¹ and Moti Yung²

¹ University of Athens, Dept. of Informatics*, Athens, Greece
aggelos@di.uoa.gr

² Google Inc. and Computer Science, Columbia University, New York, NY, USA
moti@cs.columbia.edu

Abstract. In this work we present a new paradigm for trust and work distribution in a hierarchy of servers that aims to achieve scalability of work and trust simultaneously. The paradigm is implemented with a decryption capability which is distributed and forces a workflow along a tree structure, enforcing distribution of the workload as well as fairness and partial disclosure (privacy) properties. We call the method “tree-homomorphic” since it extends traditional homomorphic encryption and we exemplify its usage within a large scale election scheme, showing how it contributes to the properties that such a scheme needs. We note that existing design models over which e-voting schemes have been designed for, do not adapt to scale with respect to a combination of privacy and trust (fairness); thus we present a model emphasizing the scaling of privacy and fairness in parallel to the growth and distribution of the election structure. We present two instantiations of e-voting schemes that are robust, publicly verifiable, and support multiple candidate ballot casting employing tree-homomorphic encryption schemes. We extend the scheme to allow the voters in a smallest administrated election unit to employ a security mechanism that protects their privacy even if **all** authorities are corrupt.

1 Introduction

While many secure ballot election schemes have been offered in the literature, the issue of scalability of e-voting schemes is typically not addressed in a direct manner.

By a scalable design we mean the task of *organizing* an election which can be state-wide or county-wide or local, based on the same underlying mechanism which has to *adapt to scale*. In such scalable design, election officials and voters are organized to vote and to manage and tally the results, respectively. In contrast, most of secure election papers deal with a single location process (e.g. [CF85, Ben87, BY81, SK33, CFSY96, Oka97, CGS97, Sch99, KY01, DJ02, KY04, G04]). As a matter of fact it is a wide belief that these schemes can be generalized to handle large-scale elections, and as a result scalability issues are

* Research partly performed at the University of Connecticut, partly supported by NSF Awards CNS-0447808,0831304,0831306.

not dealt with explicitly in the previous work, under the assumption that the cryptographic component and the distributed computing component can be deployed in a totally disjoint manner, provided that the cryptographic component is robust, in the sense that it tolerates faulty participants (see e.g. [BFPPS01]).

As we will see this assumption is not true in its entirety and not all properties of the above schemes can adapt to scale. In the era where the distributed computing paradigm includes cloud computing and P2P overlay networks, we advocate here a cryptographic protocol that incorporates trust and load constraints into the intra-server relationships. We demonstrate its usefulness within the context of a scalable e-voting scheme (though we note that other applications are possible).

Consider the paradigm for homomorphic encryption based elections. Every voter publishes his/her vote encrypted along with a proof of “ballot validity” which ensures that the submitted ciphertext encrypts a properly encoded vote. Then using the homomorphic property of the encryption scheme the ballots are pooled together into a single ciphertext and finally a set of authorities that share the decryption privileges recover the final tally in a publicly verifiable manner. Is the above design scalable? It is tempting to say yes, however caution needs to be applied. How can the election be organized to follow some given administrative tree structure? and, further how can we withstand failures of sub-election trees? These are important questions for the applicability of an election scheme in the large scale, and as we will see, they can be dealt with satisfactorily based on the homomorphic properties of the underlying encryption function.

For sharing the load of preparing the ciphertext, ciphertexts along the tree can be multiplied (generating the encryption of their sum) helping in the tally process. This seems easy, but there are crucial properties where the standard paradigm above fails to adapt to scale: this is the privacy of the voters and the fairness of the election. Scaling the number of voters and precincts as we will see is doable based on the standard paradigm. Nevertheless privacy does not scale in the same manner. This is because the enlargement of the set of authorities that share decryption privileges both in terms of numbers as well as in geographic proximity can be quite problematic as the underlying threshold cryptographic mechanisms are quite expensive to implement compared to the remaining components of the election protocol. Simply put, an efficient implementation of a scheme for a 1,000 voters and 5 authorities would not scale well to the case of 1 million voters and 5000 authorities, since it would be extremely cumbersome to initialize a threshold cryptosystem with this number of participating authorities (due to a quadratic total complexity overhead). If votes are communicated along a tree, a hierarchical organization of the trust structure makes sense. On top of that it is not only the privacy of the voters that is at stake: the fairness of the election procedure (prevention of premature partial tally disclosure) is also conditioned on the same threshold properties out of which the election satisfies the secrecy of the ballots.

Of course one might argue that the scaling factor of the election scheme need not apply evenly to voters and authorities (and therefore, e.g., a voting scheme

with 100 million voters and 5 authorities would be acceptable). *Still we believe there are situations where exactly the opposite applies:* privacy of the voters (or at least their ability to choose privacy if they wish with minimal trust in the election system) and fairness of the election procedure seem to be fundamental ingredients of a democratic system and as such they are important to pursue as part of the design requirements. If large scale e-voting would be applied without taking into account the scalability of privacy and fairness this would have potentially disastrous long term effects. Simply put, regarding the secrecy of the ballots: when one votes using one of the existing systems he/she knows that his privacy does not depend on whether BigCorp, GiantCorp and some Governmental agency do not collaborate. He knows that he is alone in the booth and that the cost of, say, installing cameras in all booths across the country is sufficiently high and such act would require such large scale coordination that it would be detected; therefore with minimal trust in the system the voter knows that his/her choices are private. In the e-voting domain privacy is not upheld as it is in the real world elections. And ultimately it is disturbing not to be able to offer efficient e-voting systems that support comparable privacy and fairness properties as such that can be achieved by trusting traditional infrastructures.

A skepticist might further ask: what is the point of scaling the authorities in parallel to the growth of the size of the elections, since there are not so many authorities out there? after all there is only BigCorp and GiantCorp and a few others. Well, this is an ill-perceived perspective of the electronic world. In real world elections, people are used to thousands of authorities that are collaboratively responsible for the election: at every level of an election procedure there exist a number of audit authorities, typically one from each party, and even international observers that participate in the protocol to ensure that the election runs smoothly. It is by the sheer number of these participants that present elections systems have the potential of offering privacy and fairness convincingly. The emerging social network and P2P computing can serve as infrastructure in an election scheme without relying on a few *big brother* entities.

Finally one may argue that privacy and fairness is just a localized property that is only relevant in the context of a small district or group of people and need not be spread across the organizational structure. This type of thinking lowers the bar for security design as after the very first level of input collection only integrity is required. This localized view of privacy and fairness has actually served us in the past but we argue that it increasingly becomes a precarious design paradigm. This is so as the ability of using algorithms for statistical inference and correlation opens new opportunities for privacy and fairness violations if one has access to fine-grained aggregate data.

The above advocates collaborative effort along the administration tree where the authorities making up the tree need to collaborate and are all involved in maintaining privacy and fairness in a scalable way.

Following these lines, in this work, not only do we discuss how the above problem of scalability of privacy and fairness can be solved but furthermore we propose a security mechanism that further allows a set of voters to protect the

privacy of their votes even in the case where all authorities are malicious. This, combined with the applicability of our designs in the large scale setting as well as their versatility with respect to arbitrary elections structures, make our main design paradigm, which we call “Scalable Secret Ballot Elections”, a realistic large-scale secure elections paradigm.

What one can observe when looking at large scale election schemes in real life, is that they are a hierarchically-organized distributed application. In the most general setting an election can be viewed as a collection of **secure subtrees**, within which ballots are tallied and accumulated securely, distributedly in a bottom-up fashion. The roots of the secure subtrees announce the accumulated sub-tally of the trees, and then there is a non-private distributed tree process which collects the various open sub-tallies into the final tally. A scalable design should be able to arrange the election in trees of any level (depth) and structure. It should deal with how the election officials in a secure subtree get organized for the election and how users at a “voting location” are organized to vote. (More generally, scale and flexibility like we advocate may apply to other protocols).

Ultimately a design with scalable privacy and fairness should take into account issues of “locality” i.e. there should not exist components with direct global scope, e.g. a large distributed entity responsible for the privacy of *all* the voters. Every entity should be clearly assigned to a component and each component should have minimal interaction restricted to other proximate components, and as a result the responsibility of each entity should be limited in a certain locality. However, at the same time, security and privacy will be achieved in a global sense, but not directly, but rather than as a synthesis of the local security properties satisfied by the design.

A number of other useful conditions and properties have been achieved in the traditional secure election literature. For example, the notion of public verifiability which assures that all actions are done as specified, and the notion of robustness which assures that the election cannot be disrupted due to misbehavior of a few individual voters or authorities. In a scalable design we would like to preserve these properties while arranging the distributed organization of election officials. Due to the scaling and global nature of a large-scale voting scheme, there are additional new issues which arise and require novel solutions. For example it is important to deal with global time management, where lower level authorities have to pass their partial tally on time, and results have to be announced by a certain time. Robustness needs to be extended to election sub-trees (faults should be isolated and dealt with in a distributed fashion in the locale they occur).

Let us summarize our results:

First, regarding the secrecy of the ballot, we make the point that in order to achieve scalability of privacy it should hold that the privacy of the ballots should scale across the hierarchical structure of the elections. To this end, we introduce and achieve a new notion of **granular secrecy** that spreads the capability of opening the ballots across all levels of the governmental hierarchy. This suggests that voters are not expected to trust a single distributed governmental entity (as

it is the case in all previous solutions) but rather trust *at least one* distributed governmental entity in a sequence of such authorities at all levels of the election hierarchical structure.

Additionally, we emphasize that voters should be given the option to strengthen further the security of their ballot within the group of people they vote with, so that their privacy is maintained even if all officials are corrupt. We call this (optional) strong privacy measure **paranoid security**. The decision to employ such a measure should be localized and transparent to higher governmental levels without the need for global coordination for its employment in a certain locale. It should be emphasized that the existence of an optional “paranoid security” mechanism enhances the trust of voters in the election system and is intended to emulate security as it is perceived in real-life elections where “paranoid security” can be ensured via physical means (e.g. private voting booths). It should be noted that the mechanism would require extra work from the voters (in particular the ability to be active in more than a single round of computation) something that we deem acceptable under the maxim that users interested in self-regulating their privacy should be de facto willing to invest more resources.

Second relevant issue when adapting to scale is that of fairness. The secure subtree roots officials, should be the ones who are able to announce the results of their subtree and no one else. This assures that one can organize the election under a global clock so that all secure subtrees are to announce the results according to a predetermined schedule, assuming the voting process ends at some time and enough time is given for tallying within subtrees. It should not be the case that a part of the election gets revealed ahead of time (say by officials at a lower level of the subtree). The implications of premature announcement of results, may bias the on-going election process. The design should take into account that such fairness may be mandated by law.

Third requirement is granularity and arbitrary architecture. It means that given any hierarchical structure in the form of a tree, an election community (officials and voters), can implement the protocol over the arbitrary structure. Additionally it should be possible for each individual election instance to decide on a level that will serve as the root of secure subtrees for secrecy. The granularity of security and fairness should be determined on the global structure, and it should be changeable from one instantiation of the voting scheme to another. It should be required that the fault (e.g. delay) in a subtree, should not affect the rest of the tallying process. More specifically if the deadline passes for a certain sub-election tree the local partial tally may be canceled or invalidated by the parent authorities who should be capable of continuing the secure election protocol without interruption or additional administrative costs. Such timely management may be mandated by law.

A preliminary version of the ideas of this paper were presented in [Yu04].

2 Model and Definitions

A voting-scheme needs to fulfill a variety of requirements to become useful. A brief presentation of these requirements follows:

Secrecy. Ensures the security of the contents of ballots. This is typically achieved by relying on the honesty of some of the active participating authorities and at the same time on some cryptographic intractability assumption.

Universal-Verifiability. Ensures that any party, including a casual observer, can be convinced that all valid votes have been included in the final tally.

Robustness. Ensures that the system can tolerate a certain number of faulty participants. Robustness is typically antagonistic to the secrecy property and typically some trade-off based solution should be employed (see also [KY01]).

Fairness. It should be ensured that no partial results become known prior to the end of the election procedure.

Dispute-Freeness. The fact that participants follow the protocol at any phase can be publicly verified by any casual third party.

Receipt-Freeness. The voter cannot provide a receipt that reveals in which way he/she voted [BT94]. Note that we do not deal explicitly with receipt freeness in this abstract, nevertheless standard techniques that use re-randomizers (see e.g. [BFPPS01]) can be readily employed in both of our schemes.

2.1 The Model

It is natural to model a practical large-scale election scheme in a multi-level way following closely the way actual nation-wide elections are performed. In such a case there could be a nation-level, a state-level, a city-level, a county-level and a precinct-level. In order to achieve scalability we can divide the eligible voters in a hierarchy of regional levels. We would call the smallest such component a *microprecinct* — the smallest administrated unit in which a batch of voters is assigned. Up to a certain level the results of the election should be private and from this level and upwards the partial tallying should be open for public reading. Such a secure subtree will be called a *sub-election*; the authority at the top of such subtree will be called the top-level authority. In the remaining we will concentrate on how the system operates in each sub-election. The hierarchical structure of a sub-election is illustrated in figure 1.

Every regional level has an authority associated with it. In a k -level partition an authority at the ℓ -th level will be denoted by A_{i_1, \dots, i_ℓ} whereas the top-level authority will be denoted by A_{top} . Each microprecinct authority A_{i_1, \dots, i_k} corresponds to a leaf in the tree of authorities and is preceded by k authorities in the path from root (the top-level authority A_{top}) to leaf. This path will be called the “active microprecinct path.”

In order to achieve fault-tolerance and other distributed security properties each authority A_{i_1, \dots, i_ℓ} is divided into t sub-authorities denoted by $A_{i_1, \dots, i_\ell}^{(1)}, \dots, A_{i_1, \dots, i_\ell}^{(t)}$. We consider the parameter t to be the same throughout but this choice is merely done for brevity. A parameter $t' < t$ denotes the minimum number of authorities that are required to act in order to perform some task that is expected from the authority A_{i_1, \dots, i_ℓ} . The division of each regional authority to a set of sub-authorities that are equally responsible for carrying out the election procedure is natural as it parallels the way regional committees in

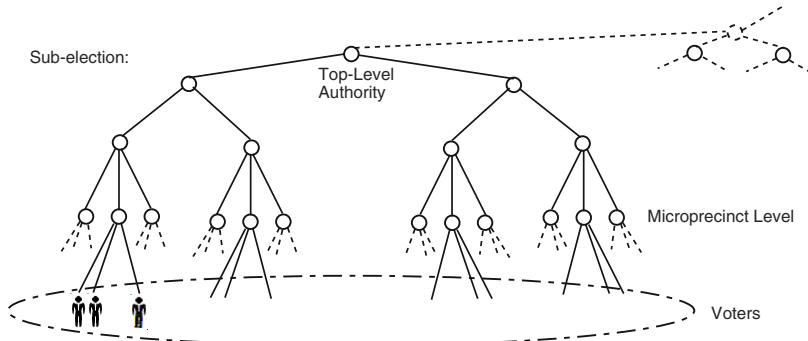


Fig. 1. The Hierarchy of the Authorities

traditional elections are formed by representatives of parties and other bodies who are present at all levels in the hierarchy.

Each authority possesses a “bulletin board” ([CF85]) which is used for all necessary communication, unless stated otherwise. The bulletin board is a public-broadcast channel with memory. Any party (even third-parties) can read information from the bulletin board. Writing on the bulletin board by the active parties is done in the form of appending data in a specially designed area for each party. Erasing from the bulletin board is not possible, and appending is verified so that any third party can be sure of the communication transcript.

In our protocols, each authority may engage in “server-based ciphertext processing.” This helps in reducing the computations of other participants. All computation performed in this manner will be publicly verifiable (e.g., by repeating the computation whenever not trusted).

2.2 Overview of Our Election Paradigm

Initialization. The hierarchical structure of the election is given and the division to sub-elections is decided. Let us denote by \mathcal{V} the set of eligible voters in a certain sub-election. Each voter has a publicly known unique identity string. The same is true for all other active participants of the election scheme. Global parameters are decided and published. Additionally we assume that all parties can consult a global clock for synchronicity purposes.

Sub-Election Set-up. Following the given hierarchical structure each authority A_{i_1, \dots, i_ℓ} which is comprised by t sub-authorities $A_{i_1, \dots, i_\ell}^{(1)}, \dots, A_{i_1, \dots, i_\ell}^{(t)}$ executes a key generation phase that results in a public-key to be used later on in the procedure. Deadlines for reporting at each level in the hierarchy are set and timers are initialized.

Voter Registration. Each voter comes to his assigned microprecinct. His identity (or pseudonym) and eligibility to vote is verified by the microprecinct authority and he gains access to the local bulletin board (note that authorization methods are outside the scope of the current presentation).

Online Phase. The “on-line phase” involves the active participation of the voters.

- **Microprecinct Level Blinding (for Paranoid Security).** If it is decided by the microprecinct, the voters execute a “0-sharing phase” that will be intended to hide each voter’s ballot among the other votes in each microprecinct. We note that such 0-sharing techniques are typical in various settings, e.g. in re-randomization of individual keys in proactive protocols [OY91].
- **Ballot-Casting.** A voter in the microprecinct A_{i_1, \dots, i_k} executes the ballot-casting procedure which combines the public-keys of all the authorities in the active microprecinct path. The vote is cast from a set of possible choices $\mathcal{C} := \{1, M, \dots, M^{c-1}\}$ where c is the number of candidates and $M = 2^{\lceil \log |V| \rceil}$. We call this technique *candidate packing* as it suggests that the sum of all individual votes in a single ($c \log M$)-bit-long register will reveal the number of votes given to each candidate (due to the choice of M , no overflow can occur in each of the c distinct ($\log M$)-bit-long regions of the summation register).
- **Corrective stage for Microprecinct level blinding (Paranoid Security).** Active voters cancel out shares of faulty voters in order to enable tallying.

Peeling Phase. When the local deadline is reached, the microprecinct authority pools votes together to form the encryption of the partial tally and “peels off” its encryption in a publicly verifiable manner. The microprecinct authority might also engage in server-based ciphertext processing (a publicly verifiable procedure that does not involve private data). The encrypted partial tally and other necessary information is passed upwards to the parent authority by writing to its bulletin board. The procedure continues recursively until the top-level authority is reached. Lower level authorities that do not report by the deadline are ignored (if needed they can be processed at a later time).

Tallying Phase. A tallying authority receives the output of the top-level authority and computes and publishes the final sub-election result. The computation of the tallying authority does not involve private data and can be repeated if needed (to ensure correct execution). Given the sum T of all votes, the number of votes μ_i that the i -th candidate received is revealed immediately due to candidate packing: $T = \mu_1 + M \cdot \mu_2 + \dots + M^{c-1} \mu_c$ (by the choice of M it holds that $M > \mu_1, \dots, \mu_c$).

Remark: The sub-election setup phase with the peeling and tallying processes constitute the “tree-homomorphic” encryption scheme.

2.3 Security Properties of Our Model and Objectives for Protocol Design

Granular Secrecy. The security of the ballot of a voter that belongs to a certain microprecinct can only be violated if at least t' of the t sub-authorities **in each level** of the active microprecinct path are malicious.

Paranoid Security. In addition to granular secrecy, the voters in a microprecinct may choose to execute the microprecinct level blinding stage (which is an optional procedure). This step blinds the casted ballots in the microprecinct level in such a way so that only their sum is accessible to the authorities. This provides *security against authorities* since even if all authorities in an active microprecinct path are malicious it will be impossible to reveal the ballot of a voter since it will be hidden in the partial sum of all votes casted in the microprecinct.

Granular Fairness. Even if t' out of the t sub-authorities are malicious in an authority A_{i_1, \dots, i_ℓ} the partial tally that corresponds to the authority cannot be revealed ahead of time unless t' out of t sub-authorities are malicious in all preceding authorities in the path from A_{i_1, \dots, i_ℓ} to the top-level authority.

Scalable Complexity. We require that the time/space complexity of the computation of each authority in a sub-election is linear in the number of children of the authority (which in the case of a microprecinct it corresponds to the number of voters assigned to the particular microprecinct).

Localized Faults. For any two disjoint sub-trees in a sub-election the (malicious or faulty) behavior of participants in one sub-tree does not have any effect in the protocol execution and security properties of the election procedure in the other sub-tree.

2.4 Non Interactive Zero Knowledge Proofs

Non-interactive zero knowledge proofs are very important cryptographic tools in the design of voting schemes. We will use the following notation for such proofs of knowledge: $\text{PKname}(\text{variables} : \text{algebraic expression})$. This notation corresponds to a transcript of a non-interactive zero knowledge proof that can be verified by any interested third party and is convincing that the issuer knows a value for each of the variables so that the given algebraic expression is satisfied. All the proofs that we use in our protocols are listed in the appendix.

2.5 The Number Theoretic Assumptions

We use standard cryptographic assumptions in our two schemes. The first instantiation of our model employs the assumption of secure ElGamal encryption which is equivalent to the Decisional Diffie Hellman assumption, see e.g. [TY98]. Our second scheme employs additionally the security assumption of the Paillier encryption function, see [Pai99]. The zero-knowledge proofs are made non-interactive by employing a random oracle, [FS87].

3 The Granular Voting Schemes

We present two instantiations of our election model. The first of our voting schemes involves more efficient on-line procedures (constant ballot-casting complexity) but an intensive tallying phase which is exponential in the number of

candidates. As a result it applies to settings with a small number of candidates where the tally belongs in a “modest range” of values. Our second voting scheme requires more computation by voters and authorities (proportional to the number of authorities in the active microprecinct path) but it possesses a very efficient tallying phase that is polynomial in the number of candidates and can be applied to the case where the number of candidates is large (and as a result the tally belongs in a “wide range” of values).

3.1 The Modest Range Tally Voting Scheme

Global Parameters: a large prime p so that $p = 2q + 1$ where q is also a prime. Elements $g, h, f \in \mathbb{Z}_p^*$ of order q with unknown relative discrete-logs. The group $\mathcal{G} := \langle g \rangle$ is a cyclic multiplicative sub-group of \mathbb{Z}_p^* of order q that corresponds to the sub-group of quadratic residues modulo p .

Sub-Election Set Up. In each authority A_{i_1, \dots, i_ℓ} , the t sub-authorities $A_{i_1, \dots, i_\ell}^{(1)}, \dots, A_{i_1, \dots, i_\ell}^{(t)}$ execute the key-generation for threshold ElGamal encryption as described in [GJKR99] (based on [Ped91]). This procedure will result in a private share s_j for each sub-authority $A_{i_1, \dots, i_\ell}^{(j)}$ that will be publicly committed and a publicly known generator $h_{i_1, \dots, i_\ell} := g^{\alpha_{i_1, \dots, i_\ell}}$ for \mathcal{G} . The pair $\langle g, h_{i_1, \dots, i_\ell} \rangle$ will serve as the public-key of the authority. We note here that a message encrypted in ElGamal fashion under the public-key of the authority can be decrypted by any t' of the t sub-authorities in a publicly verifiable manner (see e.g. [FGY96]). Finally each authority multiplies its public-key with the combined public key of the parent authority (available through the bulletin board of the parent authority) and publishes the local combined public key defined as $\hat{h}_{i_1, \dots, i_\ell} = h_{i_1, \dots, i_\ell} \cdot \hat{h}_{i_1, \dots, i_{\ell-1}}(\text{mod } p)$ (note that the top-level authority merely sets $\hat{h}_{\text{top}} = h_{\text{top}}$). Note that for the microprecinct that corresponds to the authority A_{i_1, \dots, i_k} it will hold that $\hat{h}_{i_1, \dots, i_k} = h_{\text{top}} \cdot h_{i_1} \cdot h_{i_1, i_2} \cdot \dots \cdot h_{i_1, \dots, i_k}(\text{mod } p)$.

Microprecinct Level Blinding (paranoid security). Suppose that the voters in the l -th microprecinct that corresponds to the authority A_{i_1, \dots, i_k} are denoted by $V_1^{[l]}, \dots, V_n^{[l]}$. Each voter $V_j^{[l]}$ publishes a personal generator for \mathcal{G} denoted by $h_j := h^{\alpha_j^{[l]}}$. Each voter $V_i^{[l]}$ generates n additive shares that sum up to 0: $s_{i,1} + \dots + s_{i,n} = 0(\text{mod } q)$; subsequently he publishes the pairs $\langle R_{i,j}, R'_{i,j} \rangle := \langle g^{s_{i,j}}, h_j^{s_{i,j}} \rangle$ for $j = 1, \dots, n$, together with $\text{PKEQDL}(\alpha : (R_{i,j} = g^\alpha) \wedge (R'_{i,j} = h_j^\alpha))$. The microprecinct authority calculates for each $j \in \{1, \dots, n\}$, the values $R'_j := \prod_{i=1}^n R'_{i,j}$ (this step is a server-based ciphertext processing, that is publicly verifiable).

Ballot-Casting. A voter $V_j^{[l]}$ in the l -th microprecinct controlled by A_{i_1, \dots, i_k} prepares his vote $U_j^{[l]} := f^{v_j^{[l]}} \text{ mod } p$ where $v_j^{[l]} \in \mathcal{C} = \{1, M, M^2, \dots, M^{c-1}\}$; note that if the microprecinct level blinding phase has also been performed the vote is set to $U_j^{[l]} := f^{v_j} (R'_j)^{(\alpha_j^{[l]})^{-1}} \text{ mod } p$. Then, the voter publishes his encrypted ballot

$$\langle W_j^{[l]}, B_j^{[l]} \rangle := \langle g^{r_j^{[l]}} \text{ mod } p, (\hat{h}_{i_1, \dots, i_k})^{r_j^{[l]}} \cdot U_j^{[l]} \text{ mod } p \rangle$$

where $r_j^{[l]}$ is selected at random from \mathbb{Z}_q . Finally the voter publishes the proof of ballot-validity depending on the following two cases:

- (i) $\text{PKENC1}(r : (W_j^{[l]} = g^r) \wedge (\vee_{v \in \mathcal{C}} (B_j^{[l]} = (\hat{h}_{i_1, \dots, i_k})^r f^v)))$ when microprecinct level blinding phase is omitted.
- (ii) $\text{PKENC2}(\alpha', r : (h = h_j^{\alpha'}) \wedge (W_j^{[l]} = g^r) \wedge (\vee_{v \in \mathcal{C}} (B_j^{[l]} = (\hat{h}_{i_1, \dots, i_k})^r (R'_j)^{\alpha'} f^v)))$ when the microprecinct level blinding is performed.

Microprecinct Level Blinding Corrective Phase (optional). Suppose that some voters in the l -th microprecinct controlled by A_{i_1, \dots, i_k} did not cast a ballot when the deadline is reached. If the optional microprecinct level blinding phase for paranoid security was executed this would cause problems in the tallying phase. The microprecinct authority signals to the active voters the identities of the voters that did not cast a ballot. Subsequently the remaining active voters cancel out the shares that they issued by the inactive voters as well as the shares that they computed for them. Denote the set of voters that did not cast a ballot by S' , and the set of remaining voters by $\overline{S'}$. Each voter $V_\kappa^{[l]}, \kappa \in \overline{S'}$, publishes (i) The sum of the shares that $V_\kappa^{[l]}$ issued for the inactive voters, $e_\kappa := \sum_{j \in S'} s_{\kappa, j}$, and (ii) The product of the received shares from the inactive voters: $\Phi_\kappa := (\prod_{j \in S'} R'_{j, \kappa})^{\alpha_\kappa^{-1}}$.

The value e_κ can be universally verified by checking that $g^{e_\kappa} = \prod_{j \in S'} R_{\kappa, j}$ for all $\kappa \in \overline{S'}$. The correctness of the value Φ_κ can be universally verified by having the voter $V_\kappa^{[l]}$ publish the non-interactive proof of knowledge PKEQDL $[\alpha' : (h = h_\kappa^{\alpha'}) \wedge (\Phi_\kappa = (\prod_{j \in S'} R'_{j, \kappa})^{\alpha'})]$. After the completion of this corrective round the microprecinct authority modifies the published ballots $B_\kappa^{[l]}$ for $\kappa \in \overline{S'}$ as follows: $B_\kappa^{[l]} := B_\kappa^{[l]} h^{e_\kappa} (\Phi_\kappa)^{-1} \pmod{p}$.

Peeling Phase. When the local deadline is reached, the microprecinct authority A_{i_1, \dots, i_k} in the l -th microprecinct forms the products $\langle W^{[l]}, B^{[l]} \rangle := \langle g^{\sum_{j=1}^n r_j^{[l]}}, (\hat{h}_{i_1, \dots, i_k})^{\sum_{j=1}^n r_j^{[l]}} f^{\sum_{j=1}^n v_j^{[l]}} \rangle$ (note that the microprecinct-level blinding is cancelled out since it is merely a 0-sharing). Subsequently t' out of the t sub-authorities of the microprecinct peel-off their encryption (in a publicly verifiable manner) by computing distributively $\langle W^{[l]}, B^{[l]} \cdot (W^{[l]})^{-\alpha_{i_1, \dots, i_k}} \rangle$. The new pair is propagated upwards by writing to the bulletin board of the parent authority.

Peeling continues recursively as follows: suppose that the authority A_{i_1, \dots, i_ℓ} gets report by n child authorities in the sub-election tree when the local deadline is reached. Suppose that each child authority reports the value $\langle W_i, B_i \rangle$ for $i = 1, \dots, n$; the authority pools these values to compute $\langle W, B \rangle := \langle \prod_{i=1}^n W_i, \prod_{i=1}^n B_i \rangle$ and writes $\langle W, B \rangle$ to the local bulletin board. Then t' of the t sub-authorities of A_{i_1, \dots, i_ℓ} “peel-off” their encryption in a publicly verifiable manner to compute $\langle W, B(W)^{-\alpha_{i_1, \dots, i_\ell}} \rangle$ and this value is written to the bulletin board of the parent authority as well as in the local bulletin board.

Tallying Phase. The tallying authority reads f^T as the output of the top-level authority from its bulletin board, where $T = \sum_{l,j} v_j^{[l]}$ (where l runs through all microprecincts and j runs through all voters in the l -th microprecinct). Given f^T it is possible to compute T in a brute-force manner in $|\mathcal{V}|^{c-1}$ steps where $c = |\mathcal{C}|$

is the number of candidates and $|\mathcal{V}|$ is the number of voters (this is because T belongs in a space of possible values of this size). Using the baby-step giant-step method, [Sha71], it is possible to reduce the time-complexity to $\mathcal{O}(\sqrt{|\mathcal{V}|}^{c-1})$ using equal amount of additional memory.

3.2 The Wide Range Tally Voting Scheme

Global Parameters: Let $\nu \in \mathbb{N}$ be a security parameter, with $\nu > 2c \log |\mathcal{V}|$. Also a large prime p so that $p = 2q + 1$ where q is also a prime with the property $q > 2^{4\nu} |\mathcal{V}|$. Elements $g, h \in \mathbb{Z}_p^*$ of order q with unknown relative discrete-logs.

Sub-Election Set Up. For each authority A_{i_1, \dots, i_ℓ} , the t sub-authorities $A_{i_1, \dots, i_\ell}^{(1)}, \dots, A_{i_1, \dots, i_\ell}^{(t)}$ execute the key-generation for threshold Paillier encryption as described in [FPS00]. This procedure will result in a private share for each authority $A_{i_1, \dots, i_\ell}^{(j)}$ that will be publicly committed and a joint public-key $\langle N_{i_1, \dots, i_\ell}, g_{i_1, \dots, i_\ell} \rangle$, where $N_{i_1, \dots, i_\ell} > 2^{4\nu} |\mathcal{V}|$ is a safe composite¹.

We note here that a message encrypted following the Paillier first encryption function ([Pai99]) under the public-key of the authority can be decrypted by any t' of the t sub-authorities in a publicly verifiable manner (see [FPS00, DJ03]).

Paranoid Security. The optional microprecinct level blinding phase for paranoid security is omitted here due to lack of space but it follows the same logic as the previous section.

Ballot-Casting. A voter $V_j^{[l]}$ in the l -th microprecinct controlled by authority A_{i_1, \dots, i_k} generates the additive shares $v_{0,j}^{[l]} + \dots + v_{k,j}^{[l]} = v_j^{[l]} \pmod{2^\nu}$ so that each $v_{i,j}^{[l]} < 2^\nu$ and $v_j^{[l]} \in \mathcal{C}$ corresponds to the private choice of the voter. Let $\langle N_{(0)}, g_{(0)} \rangle, \dots, \langle N_{(k)}, g_{(k)} \rangle$ be the public-keys of all the authorities in the active microprecinct path (including the microprecinct authority) for the voter $V_j^{[l]}$. The voter publishes the tuples for $i = 0, \dots, k$,

$$\langle C_{i,j}^{[l]}, B_{i,j}^{[l]} \rangle := \langle g^{v_{i,j}^{[l]}} \cdot h^{r_{i,j}^{[l]}} \pmod{p}, g_{(i)}^{v_{i,j}^{[l]}} \cdot y_{(i)}^{N_{(i)}} \pmod{N_{(i)}^2} \rangle$$

together with the proofs of knowledge PKEQCOMENC1($v, r, y : (v < 2^{4\nu}) \wedge (C_{i,j}^{[l]} = g^v \cdot h^r) \wedge (B_{i,j}^{[l]} = g_i^v y^{N_i})$) and PKSUMSET($r : \forall v \in \mathcal{C}_{\text{ext}} \prod_i C_{i,j} = g^v h^r$). We define $\mathcal{C}_{\text{ext}} := \{M^\theta + \delta 2^\nu \mid \theta = 0, 1, \dots, c-1; \delta = 0, \dots, k+1\}$.

To conclude the description of the ballot-casting procedure, we give a brief overview of the above: during ballot-casting each voter publishes in the bulletin board a 2-column “ballot-matrix” of the following form:

$C_{0,j}^{[l]}$	$B_{0,j}^{[l]}$
\vdots	\vdots
$C_{k,j}^{[l]}$	$B_{k,j}^{[l]}$

¹ A safe composite is defined as the product of two large primes p, q for which it holds that $p = 2p' + 1, q = 2q' + 1$ where p', q' are also prime.

Each row contains two encryptions of the additive share $v_{i,j}^{[l]}$, the left being a commitment to $v_{i,j}^{[l]}$ whereas the right is an encryption of $v_{i,j}^{[l]}$ (using Paillier's first encryption function, [Pai99]). The commitments are used in conjunction with the non-interactive zero knowledge proof PKSUMSET in order to show that the sum of the shares belongs in the proper range (i.e. to show that $\sum_{i=0}^k v_{i,j}^{[l]} \pmod{2^\nu} \in \mathcal{C}$). The encryptions are to be used in the peeling phase for the purpose of pooling ballots together.

Peeling Phase. Suppose that the l -th microprecinct is controlled by authority A_{i_1, \dots, i_k} and involves the voters $V_1^{[l]}, \dots, V_n^{[l]}$. The microprecinct authority multiplies all encryption columns of the published ballot-matrices in the local bulletin board (in a point-wise manner). This results in a combined encryption column of the form $\mathcal{E}^{[l]} := \langle \prod_{j=1}^n B_{0,j}^{[l]}, \dots, \prod_{j=1}^n B_{k,j}^{[l]} \rangle^T$. By the format of the ballots it holds that $\prod_{j=1}^n B_{k,j}^{[l]} = g_{(k)}^{\sum v_{k,j}^{[l]}} (Y)^{\text{multiple}(N_{(k)})} \pmod{N_{(k)}^2}$ where $\langle N_{(k)}, g_{(k)} \rangle$ is the public-key of the microprecinct authority. This is a valid encryption of the sum $\sum_{j=1}^n v_{k,j}^{[l]}$. Then, t' of the t sub-authorities of the microprecinct authority pool their shares together to decrypt in a publicly verifiable manner and obtain $(\sum v_{k,j}^{[l]}) \pmod{N_{(k)}}$ which is equal to $\sum v_{k,j}^{[l]}$ due to the choice of $N_{(k)}$ and the restrictions imposed on the shares $v_{k,j}^{[l]}$. The microprecinct authority substitutes the k -th cell of $\mathcal{E}^{[l]}$ by $\sum v_{k,j}^{[l]} \pmod{2^\nu}$ and the cell is marked as "open." Subsequently the microprecinct authority writes the column $\mathcal{E}^{[l]}$ in the bulletin board of the parent authority as well as in the local bulletin board.

Peeling continues recursively as follows: suppose that the authority A_{i_1, \dots, i_ℓ} has n child authorities in the sub-election tree that report before the local deadline. Suppose that the i -th child authority reports the column \mathcal{E}_i for $i = 1, \dots, n$; it holds that in each column the cells $\ell+1, \dots, s$ are open (and contain partial sums of voters' additive shares) and the cells $0, \dots, \ell$ contain encryptions of partial sums. The authority pools all columns by multiplying point-wise the cells that contain encryptions, whereas it sums up modulo 2^ν the cells that are open. This results in a combined encryption column \mathcal{E} . Then, t' of the t sub-authorities of A_{i_1, \dots, i_ℓ} pool their shares to decrypt the ℓ -th cell of \mathcal{E} in a publicly verifiable manner. As in the case of microprecinct operation the partial sum is reduced modulo 2^ν and the result is written in the ℓ -th cell which is marked as "open." The resulting column is written in the bulletin board of the parent authority as well as in the local bulletin board.

Tallying Phase. The tallying authority receives the encryption column from the top-level authority A_{top} where all partial sums are open. Summing all cells modulo 2^ν reveals the final result of the election $T = \sum_{l,j} v_j^{[l]} \pmod{2^\nu} = \sum_{l,j} v_j^{[l]}$ – due to the choice of ν (where l runs through all microprecincts and j runs through all voters in the l -th microprecinct).

3.3 Security Properties

Let us conclude with the discussion of the properties achieved by our two elections protocols.

Claim. (1) The Modest Range Tally Voting Scheme, assuming the security of ElGamal encryption and a random oracle, (i) satisfies universal verifiability, (ii) granular secrecy, (iii) granular fairness, (iv) dispute-freeness with the exception of the sub-election set up phase (v) security for paranoids, (vi) scalable complexity, (vii) tallying phase with exponential dependency on the number of candidates, (viii) robustness, (ix) ballot-casting with constant time/space complexity in the number of active parties.

Claim. (2) The Wide Range Tally Voting Scheme, assuming the security of Paillier encryption, ElGamal encryption and a random oracle, (i) satisfies universal verifiability, (ii) granular secrecy, (iii) granular fairness, (iv) dispute-freeness with the exception of the sub-election set up phase and the security for paranoids phase (v) security for paranoids, (vi) scalable complexity, (vii) tallying phase with polynomial dependency to the number of candidates, (viii) robustness, (ix) ballot-casting with time/space complexity proportional to the number of levels in the active microprecinct path.

References

- [BFPPS01] Baudron, O., Fouque, P.-A., Pointcheval, D., Poupart, G., Stern, J.: Practical Multi-Candidate Election system. In: The Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC (2001)
- [Ben87] Benaloh, J.: Verifiable Secret-Ballot Elections, PhD Thesis, Yale University (1987)
- [BY81] Benaloh, J., Yung, M.: Distributing the Power of a Government to Enhance the Privacy of Voters. In: The Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC (1986)
- [BT94] Benaloh, J., Tuinstra, D.: Receipt-Free Secret-Ballot Elections. In: STOC 1994 (1994)
- [CF85] Cohen (Benaloh), J.D., Fischer, M.G.: A Robust and Verifiable Cryptographically Secure Election Scheme. In: FOCS 1985 (1985)
- [CGS97] Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authority Election Scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
- [CDS94] Cramer, R., Damgård, I.B., 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)
- [CFSY96] Cramer, R., Franklin, M.K., Qchoenmakers, B., Yung, M.: Multi-Autority Secret-Ballot Elections with Linear Work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
- [DJ00] Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier’s Probabilistic Public-Key System. In: Public Key Cryptography 2001, pp. 169–136 (2001)
- [DJ02] Damgård, I., Jurik, M.: Client/Server Tradeoffs for Online Elections. In: Public Key Cryptography 2002, pp. 125–140 (2002)
- [DJ03] Damgård, I., Jurik, M.: A Length-Flexible Threshold Cryptosystem with Applications. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 350–364. Springer, Heidelberg (2003)

- [DLM82] DeMillo, R.A., Lynch, N.A., Merritt, M.: Cryptographic Protocols. In: STOC 1982, pp. 383–400 (1982)
- [DDPY94] De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On Monotone Formula Closure of SZK. In: FOCS 1994 (1994)
- [FS87] Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
- [FPS00] Fouque, P.-A., Poupart, G., Stern, J.: Sharing Decryption in the Context of Voting or Lotteries. In: The Proceedings of Financial Cryptography 2000 (2000)
- [FGY96] Frankel, Y., Gemmell, P., Yung, M.: Witness-Based Cryptographic Program Checking and Robust Function Sharing. In: STOC 1996 (1996)
- [GJKR99] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In: Eurocrypt 1991 (1991)
- [G04] Groth, J.: Efficient Maximal Privacy in Boardroom Voting and Anonymous Broadcast. In: Financial Cryptography 2004, pp. 90–104 (2004)
- [KY01] Kiayias, A., Yung, M.: Self-Tallying Elections and Perfect Ballot Secrecy. In: Proceedings of Public Key Cryptography 2002 (2002)
- [KY04] Kiayias, A., Yung, M.: The Vector Ballot e-Voting Approach. In: Financial Cryptography 2004, pp. 72–89 (2004)
- [Mer83] Merrit, M.: Cryptographic Protocols, Ph.D. Thesis, Georgia Institute of Technology (1983)
- [Oka97] Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: Workshop on Security Protocols (1997)
- [OY91] Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks. In: The Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), vol. 1291, pp. 51–21
- [Pai99] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 223. Springer, Heidelberg (1999)
- [Ped91] Pedersen, T.P.: A threshold Cryptosystem without a Trusted Third Party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
- [SK33] Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
- [Sch99] Schoenmakers, B.: A Simple Publicly Verifiable Secret Sharing Scheme and its Applications to Electronic Voting. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 148. Springer, Heidelberg (1999)
- [Sha71] Shanks, D.: Class number, a theory of factorization and genera. In: Proc. Symp. Pure Math., vol. 50, pp. 240–415. AMS, Providence (1971)
- [TY98] Tsiounis, Y., Yung, M.: On the Security of ElGamal Based Encryption. In: Public Key Cryptography (1998)
- [Yu04] Yung, M.: Tree-Homomorphic Encryption. In: DIMACS Workshop on Electronic Voting – Theory and Practice (2004)

Automatically Preparing Safe SQL Queries

Prithvi Bisht, A. Prasad Sistla, and V.N. Venkatakrishnan

Department of Computer Science
University of Illinois, Chicago, USA
`{pbisht,sistla,venkat}@cs.uic.edu`

Abstract. We present the first sound program source transformation approach for automatically transforming the code of a legacy web application to employ PREPARE statements in place of unsafe SQL queries. Our approach therefore opens the way for eradicating the SQL injection threat vector from legacy web applications.

Keywords: Static program transformation, Security by construction, Symbolic evaluation, SQL injection.

1 Introduction

In the last decade, the Web has rapidly transitioned to an attractive platform, and web applications have significantly contributed to this growth. Unfortunately, this transition has resulted in serious security problems that target web applications. A recent survey by the security firm Symantec suggests that malicious content is increasingly being delivered by Web based attacks [2], of which SQL injection attacks (SQLIA) have been of widespread prevalence. For instance, the SQLIA based Heartland data breach¹ allegedly resulted in information theft of 130 millions credit/debit cards.

SQL injection attacks are a prime example of malicious input that change the behavior of a program by sly introduction of query structure into the input strings. An application that does not perform input validation (or employs error-prone validation) is vulnerable to SQL injection attacks. Although useful as a first layer of defense, input validation often is hard to get right [3,28,14]. The absence of proper input validation has been cited as the number one cause of vulnerabilities in web applications [24].

There is an emerging consensus in the software industry that using PREPARE statements, a facility provided by many database platforms, to construct SQL queries constitutes a robust defense against SQL injections. PREPARE statements are objects that contain precompiled SQL query structures (without data). This allows a programmer to easily *isolate* and *confine* the “data” portions of the SQL query from its “code”, avoiding the need for (error-prone) sanitization of user inputs. In addition, they are efficient because they do not require any runtime tracking, and provide opportunities to the DBMS server for query optimization [1,9].

The existing practice to transform an *existing application* to make use of PREPARE statements requires extensive manual effort. The programmer needs to obtain a detailed understanding of the program that includes identification of all inter-procedural control

¹ <http://www.wired.com/threatlevel/2009/08/tjx-hacker-charged-with-heartland>

and data flows that generate vulnerable SQL queries. Furthermore, these flows have to be analyzed to obtain the equivalent code for `PREPARE` statement generation. Each such control flow needs to be carefully transformed while ensuring that the changes do not alter semantics of the program in any undesirable fashion. Furthermore, additional manual verification may be needed to ensure that the semantics of the transformed program on non-attack inputs is the same as the original program. This process could be tedious, sometimes error-prone, and certainly expensive for large-scale web applications.

The objective of this paper is to develop a *sound* method to *automate* the above transformation to `PREPARE` statements. This will overcome the deficiencies of manual approach, and would result in considerable savings of program development costs. However, designing a sound method is extremely challenging because a completely automated method needs to replicate the human understanding of the program logic that constructs SQL queries. Quite often, this understanding of program logic is guided by additional documentation such as high-level system designs, flow charts and low level program comments. An automated method that aims to eliminate / minimize human effort cannot depend on the availability or use of any such additional specifications. The web application code is, therefore, the only specification available to our method, from which an understanding of the program logic needs to be automatically extracted to guide the transformation.

This main contribution of this paper is to address this challenge by developing the first automated *sound* program transformation approach that retrofits an existing (legacy) web application to make use of `PREPARE` statements. We develop a new method that constructs a *high-level* understanding of a program's logic directly from its *low-level* string operations. This method relies on a novel insight that a program's low-level string operations along any particular control path can be viewed as a derivation of a symbolic SQL query that is parametrized by its inputs. Our method directly uses this derivation to identify and isolate any unsafe string operations that may otherwise result in injection attacks. The isolated operations are then rewritten using `PREPARE` statements, effectively eliminating the SQL injection attack vector from the web application.

Our approach is implemented in a tool called TAPS (Tool for Automatically Preparing SQL queries) which is the first reported sound tool in the literature to perform this transformation. TAPS has been successfully applied to several real world applications, including one with over 22,000 lines of code. In addition, some of these applications were vulnerable to widely publicized SQL injection attacks present in the CVE database, and our transformation renders them safe *by construction*.

As a concluding remark to the introduction, we note that there is a rich body of literature on SQL injection detection and prevention (see the next section). Our objective is to not propose “one more defense” to this problem. Instead, our contribution is quite the opposite: to develop an automatic method that will assist developers and system administrators to automatically retrofit their programs with the “textbook defense” for SQL injection.

This paper is organized as follows: Section 3 presents the problem description along with a running example. Section 4 describes our approach in detail. Section 5 presents evaluation of TAPS over several open source PHP applications. We conclude in Section 6.

2 Related Work

There has been extensive work on detecting SQL injection vulnerabilities as well as approaches for defending attacks. Due to space limitations, we briefly summarize them here (see [28] for a detailed discussion).

Defenses based on static analysis. There has been extensive research on static analysis to detect whether an application is vulnerable [21,33,3,11,22,32,10,30]. The most common theme of detection approaches is to reason about sources (user inputs) and their influence on query strings issued at sinks (sensitive operations) or intermediate points (sanitization routines). Our approach provides means for fixing such vulnerabilities through PREPARE statements.

Defenses based on dynamic analysis. Dynamic prevention of SQLIA is fairly well researched area and has a large body of well understood prevention techniques: taint based [23,34,12,17], learning based [29,28,25,19,26,5,6,4,31], proxy based [27,20].

At a high level, all these techniques track use of untrusted inputs through a reference monitor to prevent exploits. Unlike the above approaches, the high-level goal of TAPS is not to monitor the program – the goal here is to modify the program to eliminate the root causes of vulnerabilities – isolation of program generated queries from user data while avoiding any monitoring costs.

Automated PREPARE statement generation. [8] investigates the problem of automatically converting programs to generate PREPARE statements. This approach assumes that the entire symbolic query string is directly available at the sinks. This assumption does not hold in many typical applications that construct queries dynamically.

3 Background and Problem Statement

We use the following running example: a program that computes a SELECT query with a user input \$u:

```
1. $u = input();
2. $q1 = "select * from X where uid LIKE '%";
3. $q2 = f($u); // f - filter function
4. $q3 = "%' order by Y";
5. $q = $q1.$q2.$q3;
6. sql.execute($q);
```

The above code applies a (filter) function (*f*) on the input (\$u) and then combines it with constant strings to generate a query (\$q). This query is then executed by a *SQL sink* (query execution statement) at line 6.

The running example is vulnerable to SQL injection if input \$u can be injected with malicious content and the filter function *f* fails to eliminate it. For example, the user input `' OR 1=1 --` provided as \$u in the above example can break out of the expected string literal context and add an additional OR clause to the query. Typically, user inputs such as \$u are expected to contribute as literals in the parse structure of any query, specifically, in one of the two literal *data contexts*: (a) a string literal context which is enclosed by program supplied string delimiters (single quotes) (b) in a numeric literal context. SQL injection attacks violate this expectation by introducing input strings that

do not remain confined to these literal data contexts and directly influence the structure of the generated queries [6,28].

`PREPARE` statement confines all query arguments to the expected data contexts. These statements allow a programmer to declare (and finalize) the structure of every SQL query in the application. Once constructed, the parse structure of a `PREPARE` statement is frozen and cannot be altered by malformed inputs. The following is an equivalent `PREPARE` statement based program for the running example.

```
1. $q = "select * from X where uid LIKE ? order by Y";
2. $stmt = prepare($q).bindParam(0, "s", "%".f($u)."%");
3. $stmt.execute();
```

The question mark in the query string `$q` is a “place-holder” for the query argument `%f($u)%`. In the above example, providing the malicious input `u = ' or 1=1 --` to the prepared query will not result in a successful attack. This is because the actual query is parsed with these placeholders (`prepare` instruction generates `PREPARE` statement), and the actual binding to placeholders happens *after* the query structure is finalized (`bindParam` instruction). Therefore, the malicious content from `$u` cannot influence the structure of query.

The Transformation Problem: In this paper, we aim to replace all queries generated by a web application with equivalent `PREPARE` statements. A web application can be viewed as a SQL query generator, that combines constant strings supplied by the program with computations over user inputs.

Given a large web application, making a change to `PREPARE` statements, is challenging and tedious to achieve through manual transformation. To make the change, a developer must consider each SQL query execution location (sink) of the program and queries that it may execute. Depending on the control path a program may generate and execute different SQL queries at a sink. Looping behavior may be used to introduce a variety of repeated operations, such as construction of conditional clauses that involve user inputs. Sinks that can execute multiple queries need to be transformed such that each control path gets its corresponding `PREPARE` statement. This requires a developer to consider all control flows together. Also, each such control flow may span multiple procedures and modules and thus requires an analysis spanning several procedures across the source code.

A second issue in making this change is : for each control flow, a developer must extract query arguments from the original program statements. This requires reasoning about the data contexts. In the running example, the query argument `%f($u)%` is generated at line 5, and three statements provide its value: `f($u)` from line 3, and enclosing character `(%)` from line 2 and 4, respectively. The above mentioned issues make the problem of isolating user input data from the original program query quite challenging.

4 Our Approach

We will use the running example from the previous section. This application takes a user input `$u` and constructs a query in the partial query string variable `$q`. A *partial query string variable* is a variable that holds a query fragment consisting of some string

constants supplied by the program code together with user inputs. Our approach makes the following assumption about partial query strings.

Main Assumption: We require the web application to be transformed, to not perform content processing or inspection of partial query string variables.

To guarantee the correctness of our approach, we require this assumption to hold. To explain this assumption for the running example, we require that once the query string $\$q$ is formed in line 5 of the application by concatenating filtered user input $f(\$u)$ with program generated constant strings in variables $\$q1$ and $\$q3$, it does not undergo deep string processing (i.e., splitting, character level access, etc.,) further en route to the sink. To ensure that this assumption holds, our approach and implementation checks that the program code only performs the following operations on partial query string variables: (a) append with other program generated constant strings or program variables (b) perform output operations (such as writing to a log file) that are independent of query construction and (c) equality comparison with string constant `null`. Checking the above three conditions is sufficient to guarantee that our main assumption holds.

The above conditions are in fact conservative and can be relaxed by the developer, but we believe that the above assumption is not very limiting based on our experimental evaluation of many real world open source applications. In fact, the above assumption has been implicitly held by many prior approaches for SQL injection defense. Defenses such as SQLRand [5], SQLCheck [28] are indeed applicable to real world programs because this assumption holds for their target applications. We note that all of these approaches change the original program’s data values. SQLRand randomizes the program generated keywords, SQLCheck encloses the original program’s inputs with marker tags. These approaches then require that programs do not manipulate their partial query strings in arbitrary ways. For instance, if a program splits and acts on a partial query string after its SQL keywords has been randomized, it introduces the possibility of losing the effect of randomization. A small minority of query generation statements in some programs may not conform to our main criteria; in this case, our tool reports a warning and requires programmer involvement as discussed in section 4.5.

4.1 Intuitions behind Our Approach

As mentioned earlier, user inputs are expected to contribute to SQL queries in string and numeric data literal contexts. Our approach aims to isolate these (possibly unsafe) inputs from the query by replacing existing query locations in the source code with PREPARE statements, and replacing the unsafe inputs in them with safe placeholder strings. These placeholders will be bound to the unsafe inputs during program execution (at runtime).

In order to do this, we first observe that the original program’s instructions already contain the programmatic logic (in terms of string operations) to build the structure of its SQL queries. This leads to the crucial idea behind our approach: *if we can precisely identify the program data variable that contributes a specific argument to a query,*

then replacing this variable with a safe placeholder string (?) will enable the program to programmatically compute the PREPARE statement at runtime. The above approach will work correctly if our main assumption is satisfied. We indeed can ensure that the resulting string with placeholders at the original SQL sink will have (at runtime) the body of a corresponding PREPARE statement.

The problem therefore reduces to precisely identifying query arguments that are computed through program instructions. In our approach, we solve this problem through symbolic execution [18], a well-known technique in program verification. Intuitively, during any run, the SQL query generated by a program can be represented as a symbolic expression over a set of program inputs (and functions over those inputs) and program-generated string constants. For instance, by symbolically executing our running example program, we obtain the following symbolic query expression :

```
SELECT ... WHERE uid LIKE '%f($u)%' ORDER by Y
```

Notice that the query is expressed completely by constant strings generated by the program, and (functions over) user inputs. (We will define these symbolic expressions formally later.)

Once we obtain the symbolic expression, we analyze its parse structure to identify data arguments for the PREPARE statement. In our running example, the only argument obtained from user input is the string %f(\$u)% .

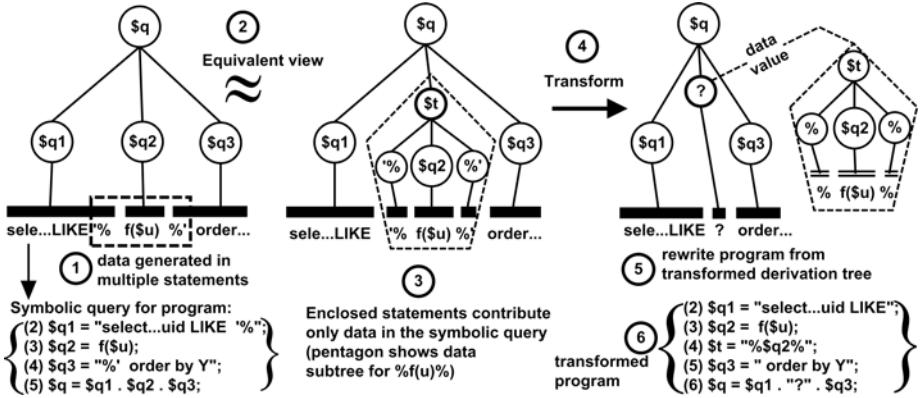
Our final step is to traverse the program backwards to the program statements that generate these arguments, and modify them to generate placeholder (?) instead. Now, we have changed a data variable of a program, such that the program can compute the body of the PREPARE statement at runtime.

In our running example, after replacing contributions of program statements that generated the query data argument %f(\$u)% with a placeholder (?), \$q at line 5 contains the following PREPARE statement body at runtime:

```
SELECT ... WHERE uid LIKE ? ORDER by Y, %%q2%
```

The corresponding query argument is the value %%q2%. Note that the query argument includes contributions from program constants (such as %) as well as user input (through \$q2) .

Approach overview. Figure 1 gives an overview of our approach for the running example. For each path in the web application that leads to a query, we generate a derivation tree that represents the structure of the symbolic expression for that query. For our example, \$q is the variable that holds the query, and step 1 of this figure shows the derivation tree rooted at \$q that captures the query structure. The structure of this tree is analyzed to identify the contributions of user inputs and program constants to data arguments of the query, as shown in steps 2 and 3. In particular, we want to identify the subtree of this derivation tree that confines the string and numeric literals, which we call the *data subtree*. In step 4, we transform this derivation tree to introduce the placeholder value, and isolate the data arguments. This change corresponds to a change in the original program instructions and data values. In the final step 5, the rewritten program is regenerated. The transformed program programmatically computes the body of the PREPARE statement in variable \$q and the associated argument in variable \$t.



4.2 Handling Straight Line Programs

We give a more precise description using a simple well defined programming language. We assume that all the variables in the language are string variables. Let \cdot denote string concatenation operator. The allowed statements in the language are of the following forms: $x = f()$, $x = y$, $x = y_1 \cdot y_2$ where x is a variable and y is a variable or a constant, y_1, y_2 are variables or constants with the constraint that at most one of them is a constant, and $f()$ is any function including the input function that accepts inputs from the user. Here we describe our approach for straight line programs. Processing of more complex programs, that include conditional statements and certain type of simple loops, is presented later in this section. The approach for such complex programs uses the procedure for straight line programs as a building block.

Derivation Trees. Now consider a straight line program P involving the above type of statements. Assume that P has l number of statements. We let S_i denote the i^{th} statement in P . With each i , $1 \leq i \leq l$, we define a labeled binary tree T_i as follows. Let $x = e$ be the statement S_i . Intuitively, T_i shows the derivation tree for the symbolic value of x immediately after execution of S_i . The root node r_i of T_i is labeled with the pair $\langle i, x \rangle$ and its left and right children (T_l, T_r) are defined as follows.

$$(T_l, T_r) = \begin{cases} ((label = x), -) & \text{if } e = f() \\ ((label = c), -) & \text{if } e = c \\ (T_j, -) & \text{if } e = y \\ (T_j, T_k) & \text{if } e = y \cdot z \end{cases}$$

Here c is a constant, T_j and T_k are the derivation trees of last statements j and k before i that update y and z , respectively.

The derivation tree T_i has two sub-trees only when e is $y \cdot z$. Note that if y (or z) is a constant then the left (or right) sub-tree is a leaf node labeled with the constant, otherwise it is a copy of some T as defined above. Figure 2 gives a program and the tree T_6 for this program.

Symbolic strings. For the program P , we construct the trees T_i , for $1 \leq i \leq l$. For each tree T_i , we define a symbolic string, called the string generated by T_i , as the string obtained by concatenating the labels of leaves of T_i from left to right. If S_i is of the form $x = e$, then we define the symbolic value of x after S_i to be the symbolic string generated by T_i . For the program given in Figure 2, the symbolic value of q after statement 6 is the string `select * from employee where salary = x1 + x2`

Data sub-strings. Assume that the last statement of P is $sql.execute(q)$ and that this is the only SQL statement in P . Also assume that statement i is the last statement that updated q . We obtain the symbolic value s of q after statement i from the tree T_i and parse it using the SQL parser. If it is not successfully parsed then we reject the program. Otherwise, we do as follows. From the parse tree for s , we identify the sub-strings of s that correspond to data portions. We call these sub-strings as data sub-strings. For each data sub-string u , we identify the *smallest* sub-tree τ_u , called data sub-tree, of T_i that generated u . Note that τ_u is a copy of T_j for some $j \leq i$. Clearly, u is a sub-string of the string generated by τ_u . Now, we consider the case when the following property (*) is satisfied. (If (*) is not satisfied we transform P into an equivalent program P' that satisfies (*) and we invoke the following procedure on P' ; this transformation is described later).

Property (*): For each data sub-string u , u is equal to the string generated by τ_u .

Program Transformation: We modify the program so that data sub-strings in symbolic strings are replaced by "?" ($Rule_1$) and all such data sub-strings are gathered into argument lists ($Rule_1$ and $Rule_2$). We achieve this as follows. For each relevant variable x , we introduce a new variable $args(x)$ that contains its list of arguments and initialize it to the empty lists in the beginning.

Let the root node of T_i be r_i and the root node of sub-tree τ_u in T_i be r_u . We traverse the tree T_i from node r_u to its root and let t_1, \dots, t_k be the nodes on this path in that order. Note that $t_1 = r_u$ and $t_k = r_i$. For each j , $1 \leq j \leq k$, let the label of node t_j be given by $\langle j, var_j \rangle$ where var_j represents the variable being updated at the node t_j (note that t_j cannot be a leaf node).

Rule₁: Eliminating data subtrees Let j' be the smallest integer such that $1 < j' \leq k$ and $t_{j'}$ has two children. Clearly, the statement $S_{j'}$ is of the form $var_{j'} = y' \cdot z'$. If $var_{j'-1} = y'$ i.e., τ_u appears in the left subtree of $t_{j'}$. We replace $S_{j'} : var_{j'} = y' \cdot z'$

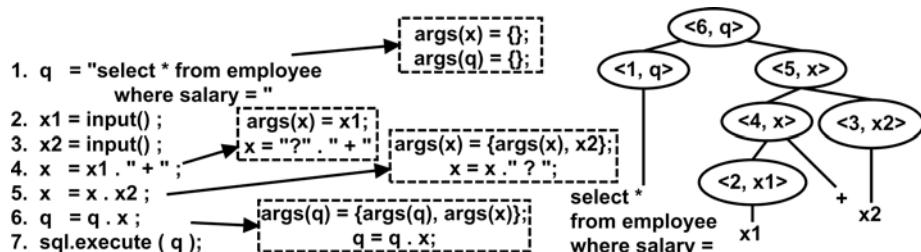


Fig. 2. Labeled derivation tree for symbolic values of q after execution of statement 6

by the following two statements.

$$\begin{aligned} \text{args}(\text{var}_{j'}) &= \begin{cases} [y'] & \text{if } z' \text{ is a constant} \\ [y'] \# \text{args}(z') & \text{if } z' \text{ is a variable} \end{cases} \\ \text{var}_{j'} &= "?" \cdot z' \end{aligned}$$

Note that the second statement above introduces “?” in the query and the first one adds corresponding data sub-string to the argument list. Here $[y']$ represents a list consisting of the single variable y' and operator $\#$ represents a list concatenation operation. The operation $[y'] \# \text{args}(z')$ computes a list by concatenating the list $[y']$ and the list $\text{args}(z')$ in that order. If $t_{j'-1}$ is a right child of $t_{j'}$, then *Rule*₁ is applied in a symmetric fashion i.e., $\text{var}_{j'} = y'."?"$, variable z' is used in place of y' , $\text{args}(y')$ is used in place of $\text{args}(z')$, and z' is added at the end of the list $\text{args}(y')$. This rule is applied to transform the lines 4 and 5 of the Figure 2.

Rule₂: Propagating arguments For each $j'', j' < j'' \leq k$, the following rule adds an additional statement immediately before the $S_{j''}$ to propagate the argument introduced by *Rule*₁.

$$\text{args}(\text{var}_{j''}) = \begin{cases} \text{args}(z'') & \text{if } S_j'' : \text{var}_{j''} = z'' \\ \text{args}(y_1'') \# \text{args}(y_2'') & \text{if } S_j'' : \text{var}_{j''} = y_1'' \cdot y_2'' \end{cases}$$

The argument lists for $\text{var}_{j''}$ is obtained by concatenating the lists $\text{args}(y_1'')$ and $\text{args}(y_2'')$ in that order. If either one of y_1'' or y_2'' is a constant string, the above rule sets the argument list to be the argument list of the non-constant variable. Note that z'' cannot be a constant string. This rule is used to transform the line 6 in the Figure 2.

Ensuring property (*): Now we consider the case when property (*) is not satisfied. In this case, we transform the program P into another equivalent program for which the property (*) is satisfied. Let Δ be the set of all data sub-strings u of the query string s such that property (*) is violated for them, i.e., u is a strict sub-string of the string generated by τ_u .

Now, observe that r_u has two children, otherwise τ_u will not be the smallest sub-tree that generated u . Let the label of r_u be $\langle m, y \rangle$. Clearly S_m is of the form $y = z_1 \cdot z_2$. Observe that each leaf node of T_i is labeled with a constant string or the name of a variable. For each $u \in \Delta$, we transform P as follows. Fix any such u . Choose a new variable x_u and add a new statement at the beginning of P initializing x_u to the empty string.

The transformation outlined below removes part of u that was computed in z_1 and stores it in x_u . Let v be a leaf node of τ_u such that the *left most* element of u falls in the label of v . The label of v can be written as $s' \cdot s''$ such that s'' is the part that falls in u . Let t_1, \dots, t_k be the sequence of nodes in τ_u from the parent of v to r_u where r_u is the root node of τ_u . For $1 \leq j < k$, replace S_j by $\text{New}(S_j)$ as defined below.

$$\text{New}(S_j) = \begin{cases} \{x_u = s'' \cdot x_u; \text{var}_j = s'\} & \text{if } j = 1 \& S_1 : \text{var}_j = s' \cdot s'' \\ \{x_u = x_u \cdot z; \text{var}_j = \text{var}_{j-1}\} & \text{if } 1 < j < k \& S_j : \text{var}_j = \text{var}_{j-1} \cdot z \end{cases}$$

After this, we identify the leaf node w of τ_u such that the *right most* element of u falls in the label of w . P is modified in a symmetric fashion updating variable x_u . Finally, we replace S_m (root of the τ_u) by the following two statements — $x_u = z_1 \cdot x_u$; $y = x_u \cdot z_2$.

The above transformation is done for each $u \in \Delta$. We say that changes corresponding to two different strings in Δ are conflicting if both of them require different changes to the same statement of P . Our handling of the cases of conflicting changes is explained in the next section. Here we assume that changes required by different strings in Δ are non-conflicting; Let P' be the resulting program after changes corresponding to data strings in Δ have been carried out. It can be easily shown that P' is equivalent to P , i.e., the query string generated in the variable q by P' is same as the one generated by P . Further more, P' can be shown to satisfy the property (*).

4.3 Handling of Conditionals and Procedures

In this section, we discuss our approach and implementation for programs that include branching statements, function invocations and loops.

Let us first consider branching statements. For programs that include these constructs, TAPS performs inter-procedural slicing of system dependency graphs (SDGs) [13]. Intuitively, for all queries that a SQL sink may receive, the corresponding SDG captures all program statements that construct these queries (data dependencies) and control flows among these statements. TAPS then computes backward slices for SQL sinks such that each slice represents a unique control path to the sink. Each of these control paths is indeed a straight line program, and is transformed according to our approach described in the previous section. A key issue here is the possibility of conflicts: when path P_1 and P_2 of a program share an instruction (statement) I that contributes to the data argument, then instruction I may not undergo the same transformation along both paths, and TAPS detects such conflicts. Conflict detection and resolution is described in more detail in Section 4.5. Also note that the inter-procedural slicing segregates unique sequences of procedures invoked to construct SQL queries. Such sequences may have multiple intra-procedural flows e.g., conditionals. These SDGs are then split further for each procedure in above construction such that each slice contains a unique control flow within a procedure.

The above discussion captures loop-free programs. Handling loops is challenging as loops in an application can result in an arbitrary number of control paths and therefore we cannot use the above approach of enumerating paths.

4.4 Loop Handling

First of all, let us consider programs that construct and execute the entire query inside a single iteration of the loop. Let us call the query so constructed a *loop independent* query. In this case, the body of the loop does not contain any intervening loops. To ensure whether a query location is loop independent, our approach checks for the following sufficient conditions: (1) the query sink is in the loop body and (2) every variable used in the loop whose value flows into the sink does not depend on any other variable from a previous iteration. Once these conditions are satisfied, our approach handles loop independent queries as described in Section 4.2.

However, there may be other instances where loop bodies do not generate entire queries. The most common example are query clauses that are generated by loop iterations. Consider the following example:

```

1. $u1 = input(); $u2 = input();
2. $q1 = "select * from X where Y =".$u1;
3. while ( --$u2 > 0){
4.   $u1 = input();
5.   $q2 = $q2." OR Y=".$u1;
6. }
7. $q = $q1.$q2;
8. sql.execute($q);

```

In this case, our approach aims to summarize the contributions of the loop using the symbolic regular expressions. In the above case, at the end of the loop, our objective is to summarize the contribution of $\$q2$ as $(\text{OR } Y=\$u1)^*$, so that the symbolic query expression can now be expressed as $\boxed{\text{select * from X where } Y = \$u1 (\text{OR } Y=\$u1)^*}$.

The goal of summarization is essentially to check whether we can introduce placeholders in loop bodies. Once we obtain a summary of the loop, if it is indeed the case that the loop contribution is present in a “repeatable” clause in the SQL grammar, we can introduce placeholders inside the loop. In the above example, since each iteration of the loop produces an OR clause in SQL, we could introduce the placeholder in statement at line 5, and generate the corresponding PREPARE statement at runtime.

Previous work [22] has shown that the body of a loop can be viewed as a grammar that represents a language contributing to certain parts of the SQL query, and a grammar can be automatically extracted from the loop body as explained there. We will need to check whether the language generated by this grammar is contained in the language spawned by the repeatable (pumped) strings generated by the SQL grammar. Note that this containment problem is not the same as the undecidable general language containment problem for CFGs, as the SQL grammar is a fixed grammar. However, a decision procedure specific to the SQL grammar needs to be built.

We instead take an alternative approach for this problem by ensuring that the loop operations produce regular structures. To infer this we check whether each statement in the body of the loop conforms to the following conditions: (1) the statement is of the form $q \rightarrow x$ where x is a constant or an input OR (2) it is left recursive of the form $q \rightarrow qx$, where x itself is not recursive, i.e., resolves to a variable or a constant in each loop iteration. It can be shown that satisfaction of these conditions yields a regular language. The symbolic parser is now augmented to see if the regular structure only generates repeatable strings in the SQL language. If this condition holds, we introduce placeholders as described earlier. We find our strategy for loops quite acceptable in practice, as shown in the next section.

4.5 Implementation

We implemented TAPS to assess our approach on PHP applications by leveraging earlier work Pixy [15,16] and extending it with algorithms to convert programs to Static

Single Assignment(SSA) format [7], and then implementation of the transformation described earlier. We briefly discuss some key points below.

We used an off-the-shelf SQL parser and augmented it to recognize symbolic expressions in query strings. The only minor change we had to make was to recognize query strings with associative array references. An associate array access such as `$x[‘member’]` contains single quotes and may conflict with parsing of string contexts. To avoid premature termination of the data parsing context, TAPS ensures that unescaped string delimiters do not appear in any symbolic expression.

Limitations and Developer Intervention. TAPS requires developer intervention if either one of the following conditions hold: (i) the main assumption is violated (Section 4) (ii) a well-formed SQL query cannot be constructed statically (e.g., use of reflection, library callbacks) (iii) the SQL query is malformed because of infeasible paths that cannot be determined statically (iv) conflicts are detected along various paths (v) query is constructed in a loop that cannot be summarized.

TAPS implements static checks for all of the above and generates reports for all untransformed control flows along with program statements that caused the failure. A developer needs to qualify a failure as: (a) generated by an infeasible path and ignore or (b) re-write of violating statements possible. The number of instances of type (a) can be reduced by more sophisticated automated analysis using decision procedures. In case of (b), TAPS can be used after making appropriate changes to the program. In certain cases, the violating statements can be re-written to assist TAPS e.g., a violating loop can be re-written to adhere to a regular structure as described earlier. The remaining cases can either be addressed manually or be selectively handled through other means e.g., dynamic prevention techniques.

In case of failures, TAPS can also be deployed to selectively transform the program such that control paths that are transformed will generate prepared queries, and those untransformed paths will continue to generate the original program’s (unsafe) SQL queries. The sufficient condition to do this in a sound manner is that the variables in untransformed part be not dependent (either directly or transitively) on the variables of the transformed paths. In this case, the transformation can be done selectively on some paths. All sinks will be transformed to `PREPARE` statements, and any untransformed paths will make use of the `PREPARE` statements (albeit with unsafe strings) to issue SQL queries with an empty argument list.

5 Evaluation

Our evaluation aimed to assess TAPS on two dimensions: (a) *effectiveness* of the approach in transforming real world applications, and (b) *performance* impact of transformation induced changes.

5.1 Effectiveness

Test suite. Table 1 column 1 lists SQLIA vulnerable applications from another research project on static analysis [32] and applications with known SQLIA exploits from Common Vulnerabilities and Exposures (CVE 2009) repository. This table lists their

Table 1. Effectiveness suite applications, transformed SQL sinks and control flows: TAPS transformed over 93% and 99% of the analyzed control flows for the two largest applications

Application	Size (LOC)	CVE Vulnerability (prefix CVE-)	Analyzed SQL Sinks	Analyzed Control Flows	Transformed SQL Sinks	Transformed Control Flows	Human Intervention Flows
WarpCMS	22,773	-	14	200	14	186	14
Utopia NewsPro	7,323	-	2	336	2	333	3
AlmondSoft	6,633	2009-3226	22	33	17	27	6
PortalXP TE	5,121	2009-3148	122	122	122	122	0
Gravity Board	2,422	2009-1277	62	62	62	62	0
MyNews	1,792	2009-0739	1	34	1	34	0
Auth	284	2009-0738	1	5	1	5	0
BlueBird	288	2009-0740	1	5	1	5	0
Yap Blog	264	2009-1038	2	6	2	6	0

codebase sizes in lines of code and any known CVE vulnerability identifiers (column 2 and 3), number of analyzed SQL sinks and control flows that execute queries at SQL sinks (column 4 and 5), transformed SQL sinks and control flows (column 6 and 7) and number of control flows that required developer intervention (column 8). In this test suite, the larger applications invoked a small number of functions to execute SQL queries. This caused the number of analyzed sinks and control flows to vary across applications.

Transformed control flows. For the three largest applications, TAPS transformed 93%, 99% and 81% of the analyzed control flows. Although smaller in LOC size, the Utopia news pro application had a greater fraction of code involving complex database operations and required analyzing more control flows than any other application. For the remaining applications, TAPS achieved a transformation rate of 100%. This table suggests that TAPS was effective in handling the many diverse ways that were employed by these applications to construct queries.

TAPS did not find any partial query string variables used in operations other than append, null checks and output generation / logging (supports main assumption from Section 4). Further, TAPS did not encounter conflicts while combining changes to program statements required for transformed control flows.

Untransformed control flows. The last column of the Table 1 indicates that TAPS requires human intervention to transform some control flows.

As TAPS depends on symbolic evaluation, it did not transform flows that obtained queries at run time e.g., the Warp CMS application used SQL queries from a file to restore the application's database. In two other instances, it executed query specified in a user interface. In both these cases, no meaningful PREPARE statement is possible as external input contributes to the query structure. If the source that supplies the query is trusted, then these flows can be allowed by the developer. The limitations of the SQL parser implementation were responsible for two of the three failures in the Utopia news pro application and the rest are discussed below.

Table 2. Transformation changed less than 5% lines for large applications

Application	Statements changed (%)	Args extracted Avg (max)	Functions traversed Avg (max)	SSA conversion time (%)	Flow enumeration time (%)	Static checks time (%)	Transf- ormation time (%)
WarpCMS	438 (1.9%)	6.6 (27)	2.2 (3)	98.6	0.4	0.4	0.6
Utopia News Pro	333 (4.5%)	1.1 (8)	2.9 (6)	86.9	5.3	6.7	1.1
AlmondSoft	46 (0.7%)	1.3 (4)	1.3 (2)	61.3	12.2	0.1	26.4
PortalXP TE	332 (6.5%)	1.5 (9)	1.0 (1)	96.7	1.0	2.2	0.1
Gravity Board	172 (7.1%)	1.5 (15)	1.0 (1)	94.8	1.3	3.3	0.6
MyNews	56 (3.1%)	2.4 (5)	2.5 (3)	80.7	10.8	2.2	6.3
Auth	17 (6.1%)	3.0 (4)	2.0 (2)	23.4	37.3	8.9	30.4
BlueBird	17 (6.0%)	3.0 (4)	2.0 (2)	23.5	34.6	12.4	29.5
Yap Blog	8 (3.0%)	4.0 (7)	2.0 (2)	53.5	14.2	16.8	15.5

Queries computed in loops. A total of 18 control flows used loops that violated restrictions imposed by TAPS and were not transformed (11 - Warp CMS, 1 - Utopia news pro, 6 - AlmondSoft). These control flows generated queries in loop bodies that used conditional statements or nested loops. We also found 23 instances of queries computed in loops, including a summarization of `implode` function, that were successfully transformed. In all such cases queries were either completely constructed and executed in each iteration of the loop or loop contributed a repeatable partial query.

For untransformed flows TAPS precisely identified statements to be analyzed e.g., the Warp CMS application required 195 LOC to be manually analyzed instead of complete codebase of 22K LOC. This is approximately two orders of magnitude reduction in LOC to be analyzed.

Changes to applications. As shown in the second column of Table 2 a small fraction of original LOC was modified during transformation. The columns 3 and 4 of this table show average (maximum) number of data arguments extracted from symbolic queries and functions traversed to compute them, respectively. 2% of changes in LOC were recorded for Warp CMS - the largest application, whereas approximately 5% of lines changed for database intensive Utopia news pro application. We noticed that a significant portion of code changes only managed propagation of the data arguments to `PREPARE` statements. Some of these changes can be eliminated by statically optimizing propagation of arguments list e.g., for all straight line flows that construct a single query, `PREPARE` statement can be directly assigned the argument list instead of propagating it through the partial queries. Overall, this small percentage of changes points to TAPS's effectiveness in locating and extracting data from partial queries.

Further, as columns 3 and 4 suggest, TAPS extracted a large number of data arguments from symbolic queries constructed in several non-trivial inter-procedural flows. For a manual transformation both of these vectors may lead to increased effort and human mistakes and may require substantial application domain expertise. For successfully transformed symbolic queries the deepest construction spanned 6 functions in the Utopia news pro application and a maximum of 27 arguments (in a single query) were extracted for the Warp CMS application, demonstrating robust identification of arguments.

5.2 Performance Experiment

Performance of transformed applications. TAPS was assessed for performance overhead on a microbench that consisted of an application to issue an `insert` query. This application did not contain tasks that typically interleave query executions e.g., HTML generation, formatting. Further, the test setup was over a LAN and lacked typical Internet latencies. Overall, the microbench provided a worst case scenario for performance measurement.

We measured end-to-end response times for 10 iterations each with TAPS transformed and original application and varied sizes of data arguments to `insert` queries from 256B to 2KB. In some instances TAPS transformed application outperformed the original application. However, we did not find any noteworthy trend in such differences and both applications showed same response times in most cases. It is important to note here that dynamic approaches typically increase this overhead by 10-40%. Whereas, TAPS transformed application's performance did not show any differences in response times. Overall, this experiment suggested that TAPS transformed applications do not have any overheads.

Performance of the tool. We profiled TAPS to measure the time spent in the following phases of transformation: conversion of program to SSA format, enumeration of control flows, static checks for violations described earlier, derivation tree generation and changing the program. The time taken by each phase is summarized in the last four columns of Table 2. The largest application took around 2 hours to transform whereas the rest took less than an hour. The smallest three applications were transformed in less than 5 seconds. For large applications TAPS spent a majority of time in the SSA conversion. The only exception to this case occurred for AlmondSoft application which had smaller functions in comparison to other applications and hence SSA conversion took lesser time. We wish to note here that TAPS is currently not optimized. A faster SSA conversion implementation may improve performance of the tool and by summarizing basic blocks some redundant computations can be removed. For a static transformation these numbers are acceptable.

6 Conclusion

In this paper, we presented TAPS, a static program transformation tool that modifies web applications to make use of `PREPARE` statements. We presented experimental results with several open-source applications to assess the effectiveness of TAPS. Our approach provides evidence that it is possible to successfully design retrofitting techniques that guarantee security (by construction) in legacy applications, and eliminate well known attacks.

Acknowledgments

This work was supported in part by National Science Foundation grants CNS-0716584, CNS-0551660, CNS-0845894, CNS-0917229, ITR-0716498, CCF-0916438 and CCF-0742686. Thanks are due to Mike Ter Louw and Kalpana Gondi for their suggestions on improving the draft. Finally, we thank the anonymous referees for their feedback.

References

1. JDBC: Using a prepared statements,
<http://java.sun.com/docs/books/tutorial/jdbc/basics/prepared.html>
2. Symantec Internet Security Threat Report, vol. XI. Technical report, Symantec (March 2007)
3. Balzarotti, D., Cova, M., Felmetser, V., Jovanovic, N., Kirda, E., Kruegel, C., Vigna, G.: Saner: Composing static and dynamic analysis to validate sanitization in web applications. In: IEEE Symposium on Security and Privacy, Oakland, California, pp. 387–401 (2008)
4. Bandhakavi, S., Bisht, P., Madhusudan, P., Venkatakrishnan, V.N.: Candid: preventing sql injection attacks using dynamic candidate evaluations. In: ACM Conference on Computer and Communications Security, Alexandria, Virginia, USA, pp. 12–24 (2007)
5. Boyd, S.W., Keromytis, A.D.: SQLrand: Preventing SQL Injection Attacks. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 292–302. Springer, Heidelberg (2004)
6. Buehrer, G., Weide, B.W., Sivilotti, P.A.G.: Using parse tree validation to prevent sql injection attacks. In: 5th International Workshop on Software Engineering and Middleware, Lisbon, Portugal, pp. 106–113 (2005)
7. Cytron, R., Ferrante, J., Rosen, B.K., Wegman, M.N., Zadeck, F.K.: Efficiently computing static single assignment form and the control dependence graph. ACM Transactions on Programming Languages and Systems 13(4), 451–490 (1991)
8. Dysart, F., Sheriff, M.: Automated fix generator for sql injection attacks. In: ISSRE 2008: Proceedings of the 2008 19th International Symposium on Software Reliability Engineering, Seattle, WA, pp. 311–312 (2008)
9. Flak, H.: MYSQL prepared statements,
<http://dev.mysql.com/tech-resources/articles/4.1/prepared-statements.html>
10. Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., Tao, L.: A static analysis framework for detecting sql injection vulnerabilities. In: International Computer Software and Applications Conference, Beijing, China, pp. 87–96 (2007)
11. Halfond, W.G.J., Orso, A.: AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In: IEEE/ACM international Conference on Automated Software Engineering, Long Beach, CA, USA, pp. 174–183 (2005)
12. Halfond, W.G.J., Orso, A., Manolios, P.: Using positive tainting and syntax-aware evaluation to counter sql injection attacks. In: ACM SIGSOFT International Symposium on Foundations of Software Engineering, Portland, Oregon, USA, pp. 175–185 (2006)
13. Horwitz, S., Reps, T., Binkley, D.: Interprocedural slicing using dependence graphs. In: ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation, Atlanta, Georgia, pp. 35–46 (1988)
14. Howard, M., Leblanc, D.: Writing Secure Code. Microsoft Press, Redmond (2001)
15. Jovanovic, N., Kruegel, C., Kirda, E.: Pixy: A static analysis tool for detecting web application vulnerabilities (short paper). In: IEEE Symposium on Security and Privacy, Oakland, California, pp. 258–263 (2006)
16. Jovanovic, N., Kruegel, C., Kirda, E.: Precise alias analysis for static detection of web application vulnerabilities. In: PLAS 2006: Proceedings of the 2006 Workshop on Programming Languages and Analysis for Security, Ottawa, Ontario, Canada, pp. 27–36 (2006)
17. Keyzun, A., Guo, P.J., Jayaraman, K., Ernst, M.D.: Automatic creation of sql injection and cross-site scripting attacks. In: IEEE International Conference on Software Engineering, Vancouver, Canada, pp. 199–209 (2009)

18. King, J.C.: Symbolic execution and program testing. *Communications of the ACM* 19(7), 385–394 (1976)
19. Kosuga, Y., Kono, K., Hanaoka, M., Hishiyama, M., Takahama, Y.: Sania: Syntactic and semantic analysis for automated testing against sql injection. In: Computer Security Applications Conference, Annual, pp. 107–117 (2007)
20. Liu, A., Yuan, Y., Wijesekera, D., Stavrou, A.: Sqlprob: a proxy-based architecture towards preventing sql injection attacks. In: ACM Symposium on Applied Computing, Honolulu, Hawaii, pp. 2054–2061. ACM, New York (2009)
21. Livshits, V.B., Lam, M.S.: Finding security vulnerabilities in java applications with static analysis. In: USENIX Security Symposium, Baltimore, MD, p. 18 (2005)
22. Minamide, Y.: Static approximation of dynamically generated web pages. In: International Conference on World Wide Web, Chiba, Japan, pp. 432–441 (2005)
23. Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J., Evans, D.: Automatically hardening web applications using precise tainting. In: IFIP International Information Security Conference, Chiba, Japan, pp. 295–308 (2005)
24. OWASP. The ten most critical web application security vulnerabilities,
<http://www.owasp.org>
25. Pietraszek, T., Berghe, C.V.: Defending Against Injection Attacks through Context-Sensitive String Evaluation. In: Recent Advances in Intrusion Detection, Seattle, Washington (September 2005)
26. Rietta, F.S.: Application layer intrusion detection for sql injection. In: Annual Southeast Regional Conference, Melbourne, Florida, pp. 531–536. ACM, New York (2006)
27. Sekar, R.: An efficient black-box technique for defeating web application attacks. In: Network and Distributed Systems Symposium, San Diego, CA (2009)
28. Su, Z., Wassermann, G.: The essence of command injection attacks in web applications. In: ACM Symposium on Principles of Programming Languages, Charleston, South Carolina, USA, pp. 372–382 (2006)
29. Thomas, S., Williams, L., Xie, T.: On automated prepared statement generation to remove sql injection vulnerabilities. *Inf. Softw. Technol.* 51(3), 589–598 (2009)
30. Tripp, O., Pistoia, M., Fink, S.J., Sridharan, M., Weisman, O.: Taj: effective taint analysis of web applications. In: PLDI 2009: Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation, Dublin, Ireland, pp. 87–97 (2009)
31. Valeur, F., Mutz, D., Vigna, G.: A Learning-Based Approach to the Detection of SQL Attacks. In: Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Vienna, Austria, pp. 123–140 (July 2005)
32. Wassermann, G., Su, Z.: Sound and precise analysis of web applications for injection vulnerabilities. In: ACM SIGPLAN Conference on Programming Language Design and Implementation, San Diego, California, USA, pp. 32–41 (2007)
33. Xie, Y., Aiken, A.: Static detection of security vulnerabilities in scripting languages. In: USENIX-SS 2006: Proceedings of the 15th Conference on USENIX Security Symposium, Vancouver, BC, Canada (2006)
34. Xu, W., Bhatkar, S., Sekar, R.: Taint-enhanced policy enforcement: a practical approach to defeat a wide range of attacks. In: USENIX-SS 2006: Proceedings of the 15th Conference on USENIX Security Symposium, Vancouver, BC, Canada (2006)

PKI Layer Cake: New Collision Attacks against the Global X.509 Infrastructure

Dan Kaminsky¹, Meredith L. Patterson¹, and Len Sassaman²

¹ IOActive, Inc.

² Katholieke Universiteit Leuven

1 Introduction

Research unveiled in December of 2008 [15] showed how MD5’s long-known flaws could be actively exploited to attack the real-world Certification Authority infrastructure. In this paper, we demonstrate two new classes of collision, which will be somewhat trickier to address than previous attacks against X.509: the applicability of MD2 preimage attacks against the primary root certificate for Verisign, and the difficulty of validating X.509 Names contained within PKCS#10 Certificate Requests. We also draw particular attention to two possibly unrecognized vectors for implementation flaws that have been problematic in the past: the ASN.1 BER decoder required to parse PKCS#10, and the potential for SQL injection from text contained within its requests. Finally, we explore why the implications of these attacks are broader than some have realized — first, because *Client Authentication* is sometimes tied to X.509, and second, because Extended Validation certificates were only intended to stop phishing attacks from names similar to trusted brands. As per the work of Adam Barth and Collin Jackson [4], EV does not prevent an attacker who can synthesize or acquire a “low assurance” certificate for a given name from acquiring the “green bar” EV experience.

The attacks we will discuss in this paper fall into the following categories:

1. **MD2RSA Signature Transfer:** Verisign’s MD2 root can be exploited by creating a malicious intermediate with the same MD2 hash as its parent and transferring the signature from the root to the malicious intermediate.
2. **Subject Name Confusion:** Inconsistent interpretation of the Subject X.509 Name in a PKCS#10 request can cause a CA to emit a certificate for an unauthorized Common Name. Existing PKCS APIs vary in their handling of Common Names and Subject Names, and these differences can be exploited in a number of ways which we will explore in detail.
3. **PKCS#10-Tunneled SQL Injection:** Certificate Authorities inserting PKCS#10 Subject Names into a database do not necessarily employ comprehensive string validation for the BMPString, UTF8String and UniversalString types, which allows SQL injection attacks. Given the special trust that a CA’s database backend presupposes for the rest of the Internet, SQL injection is especially problematic.
4. **PKCS#10-Tunneled ASN.1 Attacks:** Certificate Authorities exposing a PKCS#10 receiver may be exposing unhardened ASN.1 BER listeners.

ASN.1 BER is tricky to parse, with many possibilities for consistent and predictably exploitable attack surfaces. The PROTOS project [12] found a large number of vulnerabilities via the SNMP consumer, but it is possible that some of the ASN.1 BER parsers found in commercial CA implementations were not covered in the 2002 PROTOS lockdown and thus are still vulnerable.

5. **Generic SSL Client Authentication Bypass:** The MD2 attacks in this paper may have larger implications in certain deployments. An attacker with the ability to directly issue certificates — rather than just the ability to get an arbitrary X.509 Subject Name past a validator — gets access to the “Client Authentication” EKU (Extended Key Usage) attribute that controls whether a certificate allows for authenticating a client to a server. Since Root CAs do not normally issue certificates with “Client Authentication” set, some systems may not test for what would happen if such a certificate arrived. This may create a generic authentication bypass in some systems. A similar bypass may be extended from Stevens and Sotirov’s MD5 collisions, in situations where the Client Authentication EKU (which is not present in the root certificate they attacked) is insufficiently validated.
6. **EV Hijacking:** EV certs were designed to address phishing attacks where a bank at www.bankoffoo.com is suffering attacks from the owner of www.bank-of-foo.com or www.bankoffoo.com. They were specifically not designed to deal with the case where an attacker has a certificate, even a low assurance certificate, for www.bankoffoo.com, and the attacker has a DNS or other route manipulation attack, e.g. DNS cache poisoning [19]. Barth and Jackson have shown that browsers do not enforce a scripting barrier between <https://www.bankoffoo.com> (EV certified) and <https://www.bankoffoo.com> (Low Assurance certified). Thus, an attacker need simply proxy enough of an SSL session to get the main HTML of a page loaded in EV (thus causing the green bar), then kill the TCP session. After that, the attacker can host any script from the Low Assurance cert, and that script will inevitably be merged with the real site with no negative impact on the EV experience.

We will summarize the recent history of attacks against the CA infrastructure; describe the methodology used to discover the attacks listed above; investigate these attacks in greater detail; outline a principled approach for remediating these issues and what steps browser manufacturers, cryptographic API manufacturers and certificate authorities need to take; and finally, identify directions for future work.

2 Background

The SSL protocol [2] is used for encrypting reliable data flows from one endpoint to another. But encryption without authentication is worthless: one can easily end up encrypting information with the key of an attacker! SSL manages authentication via *certificates* — assertions of identity that are cryptographically

signed by mutually trusted third parties known as Certificate Authorities, or CAs. Verisign is probably the Internet’s most well known CA, but the CA infrastructure includes over 200 issuers, all of whom handle edge cases in slightly different ways [3]. During 2008, the CA system weathered a series of shocks. Mike Zusman of Intrepidus Research was able to bypass WHOIS validation at one CA by claiming his desired certificate — for Microsoft’s www.live.com — was to be used “for internal servers only”. The CA Startcom also discovered a competing CA that entirely failed to check whether a certificate requester was a legitimate representative for the domain in question.

Beyond these implementation flaws, Kaminsky [19] exposed the basic design of CA validation via both WHOIS email and HTTPS-via-IP-in-DNS as faulty by means of DNS cache poisoning attacks. If DNS is compromised at the CA, both the email and the HTTPS connection can easily be subverted. While DNS has been remediated at all known CAs, other route manipulation mechanisms, such as Pilosov’s BGP attacks [11], create some continuing exposure (though the BGP stream is small enough, and logged enough, for firms such as Renesys to know immediately if such an attack took place).

The most widely publicized attack against CAs in some time occurred in December 2008, with Stevens and Sotirov’s applied work against CAs that still used MD5 as their hash algorithm for certificate signing. MD5 had been known to be insecure since at least 1996, with a regular stream of findings against the algorithm, punctuated in particular by the generation of MD5 collisions in 2004 [20] and the extension of these attacks to chosen prefix attacks in 2007 [14]. Stevens and Sotirov demonstrated a real-world application of the chosen-prefix attack by finding a CA, RapidSSL, that used MD5 and generated entirely predictable certificates (in particular, the Serial Number and Signing/Expiration time fields) and giving it a PKCS#10 request that forced it to generate a certificate that had the same MD5 hash as an intermediate certificate they had already generated. They then transferred RapidSSL’s signature to their intermediate certificate, creating a forged certificate that could issue certificates for www.bank.com.

Luckily, very few CAs used MD5 at that time, and since then they have switched to SHA-1. However, the use of SHA-1 does not prevent the attacks we describe below.

2.1 Current Status of These Vulnerabilities

In this paper, we discuss the vulnerabilities we have discovered as they pertain to versions of X.509 certificate vendors and browser software in early 2009, prior to our disclosure of these attacks to the affected parties. Many of the attacks we describe have been patched or mitigated through our collaboration with vendors, prior to our public disclosure in August 2009. In particular, the Verisign MD2 root certificate has been superseded in Internet Explorer, Mozilla, Chrome, Android, Safari, Opera, and NSS; Postfix has remediated the early null termination issue; and Microsoft’s CryptoAPI has remediated early null termination and the integer-overflow inefficient encoding issue.

However, due to the issue of legacy code persisting long after security updates have become available, the potential for attacks executed using malicious X.509 certificates issued prior to the discovery of these flaws, and the impossibility of evaluating every X.509 infrastructure in use (especially those used internally to organizations whose CAs are not exposed to the greater Internet), these attacks remain relevant today. We refer the reader to the individual vendors for comprehensive attack resistance information.

3 Methodology

Although ASN.1 is a well-established standard, not all ASN.1 parsers are created equal. It is a complicated format, requiring a context-sensitive parser. Context-free grammars can easily be converted to parsers using a parser generator such as `yacc` or `bison`, but generating a context-sensitive parser is difficult in mainstream (i.e., strictly evaluated) languages [8]. Moreover, the ASN.1 specification is not written in a fashion conducive to implementing an ASN.1 parser with a parser generator. Thus, in practice, the ASN.1 parsers that X.509 implementations rely on are handwritten, and the likelihood that the parse trees generated¹ by two separate implementations will vary (in other words, that they implement slightly different grammars) is high. The context-free equivalence problem — “given two CFGs, F and G , determine whether $L(F) = L(G)$ ” — is known to be undecidable, and thus the context-sensitive equivalence problem is as well.

Therefore, since there can be no guarantee that two ASN.1 parsers that were not generated from a CSG specification actually parse the exact same language, we examined subtle differences in the ways that different ASN.1 parsers handle X.509 certificates. We also deliberately focused on unusual representations of key components of an X.509 certificate, such as OIDs and Common Names: if one implementation can be tricked into misinterpreting a sequence, S , as a desired sequence, S' , we can get a CA using an implementation which does *not* misinterpret S to sign a certificate containing S , and any browser using the first implementation will treat the certificate as a valid, signed certificate containing S' . All of our Subject Name confusion attacks rely on this strategy, and until ASN.1 implementations can agree on a consistent, well-defined grammar from which to generate their parsers, it is certain that similar attacks will emerge.

4 Attacks

4.1 MD2RSA Signature Transfer

As late as 1998, Verisign was still issuing certificates using a predecessor of MD5, the MD2 algorithm [3]. Historically, in the choice between MD2, MD4, and MD5, MD2 offered the highest security level at the expense of speed [6].

¹ This is a simplification; most ASN.1 parsers do not explicitly generate parse trees that can be recovered, but mathematically a tree structure exists.

However, ten years after RFC 2313 advocated MD2, a 2^{73} preimage attack was published [18]. Although 2^{73} is well outside the bounds of trivial computation, given that the previous attack was on the order of 2^{97} [9], this can be considered one mathematical advance away from a distributed-computation work effort.

Given that MD2RSA has not been used to sign certificates for over a decade, it is reasonable to ask whether it would matter if MD2 fell. Unfortunately, the answer is yes. Verisign’s primary root certificate — which is trusted by all browsers, and required to validate certificates from sites such as <https://www.amazon.com> — is signed with MD2RSA. Anything that this certificate signs is fully trusted. However, signatures are only valid across hashes, and Verisign has signed its own root certificate’s MD2 hash. Thus, if we can generate an intermediate CA certificate with the same MD2 hash as Verisign’s root, we can transfer the RSA signature from the root to the intermediate, and the signature will still be valid. Like Stevens and Sotirov, this attack transfers a signature from a legitimate certificate to a forged one, using the preimage to keep the signature valid. However, this attack can be performed entirely offline: there is no need to trick a CA server into signing something it ought not to.

As of August 2009, Verisign has reissued its root certificate using SHA-1 as the signature hash. The MD2-signed certificate cannot be revoked, but as the new certificate replaces the old one in shipped browsers, the number of browsers still using the MD2 certificate will dwindle, rendering this attack obsolete. We informed Verisign of our discovery early in our work, and their SHA-1 reissue was a direct response to this.

4.2 Subject Name Confusion

Acquiring a certificate involves submitting a public key and a claimed identity to a certification authority, generally via an ASN.1 BER-encoded PKCS#10 request² submitted through a web form. RFC 2986 describes the full schema of PKCS#10 requests; however, due to the nature of ASN.1, the encoding reflects as little of the schema as possible, instead trusting that a decoder will have the schema compiled into it.

We focus on the Subject X.509 Name because it is at the heart of the trust model in certificates. An X.509 Name is an ASN.1 Sequence of Sets of Sequences of OID/String pairs. These pairs can represent many descriptors, including Country, Organization, and Organizational Unit, but in the context of web browsers, the only name that matters is the Common Name, since the name of the website being secured is compared against the Common Name. The Common Name is thus the one element that a CA must validate correctly, or

² In an ideal world, the ASN.1-based protocols mentioned in this paper would use DER rather than BER. Unfortunately, thanks to Postel’s robustness principle — “be conservative in what you send, be liberal in what you accept” — real-world encoders are willing to accept loosely-encoded BER bytestreams when called upon to parse. As we shall see, this practice paves the way for subtle variations between what different CAs will accept, and therein lies substantial danger.

else it will issue a certificate granting rights for names that the user does not legitimately represent.

There are thus two classes of consumer for the same sequence of bytes: CAs and browsers. If a CA and a browser parse the same sequence differently, a CA may grant rights incorrectly, or a browser may misinterpret what entity a certificate represents. We now examine several real-world cases of this problem.

Multiple Common Names in one X.509 Name are handled differently by different APIs. Consider an X.509 Name where 2.5.4.3 is an OID paired with a String, and this pair constitutes a Sequence (embedded in a Set) representing the Common Name. If the Name contains more than one Common Name Sequence, and each Sequence has the OID 2.5.4.3, which one will be interpreted as the Common Name? Unfortunately, this behavior turns out to be implementation-dependent. We identified four possible policies:

1. First: The Sets comprising the Name are scanned for Sequences with an OID of 2.5.4.3. The first one that qualifies returns the associated String.
2. All-Inclusive: Each Sequence that matches the OID has its associated String added to a list, which is returned to the caller.
3. Last: The Sets of the Sequence are scanned, and whenever a Sequence is found that matches the desired OID, the planned response is updated to contain only the associated String. The last Sequence to match has its String returned.
4. Subject: No filtering is done. The entire X.509 subject is returned, either as a string or as a list, and the caller must extract the CNs in which it is interested. In other words, this is a client-side policy.

OpenSSL’s command-line tools use the Subject policy, and require callers to implement text parsers, which must themselves implement one of the above policies. The OpenSSL API provides functions which can be used to extract Common Names; in a few lines of code, one can retrieve a list and iterate through it. However, we discovered that many open-source projects only retrieve the first matching CN — in fact, we could find *no* examples of open-source projects that process this list properly [17,1,21,16,10].

We next consider browsers. Internet Explorer, which uses Microsoft’s CryptoAPI, follows the All-Inclusive policy: if *any* CN in the X.509 Subject Name matches the domain being browsed, then IE assumes that that CN has been validated by the issuing CA. This technically allows an attacker to shoehorn as many CNs into a Subject Name as he wants — the specification never mentions a limit — though in practice CAs limit the size of certificates they will generate. Still, this allows for a degree of parallelization in attack generation against IE.

NSS, the cryptographic library behind Mozilla Firefox, respects only the *last* CN in the Subject Name. This would limit an attacker to one malicious name per certificate — except that the name may be a wildcard, which NSS does not restrict against. As such, one successful breach against one CA will allow SSL bypass against all names under Firefox.

Inefficient BER encodings of OIDs can lead to some APIs recognizing the OID of Common Names. Validating the X.509 Subject Name in a certificate request against the (somehow) validated identity of the user requesting a certificate is the task of the CA. In practice, we found that validation is limited to the Common Name; all other fields, e.g., Country, Organization Name, etc., are ignored. This leads to an even more hazardous source of disagreement between a browser and a CA: what happens when they disagree on what constitutes a Common Name in the first place?

As we know, a CN is an ASN.1 BER Sequence consisting of the OID 2.5.4.3 followed by a String containing the name of the website being authorized. However, BER’s flexibility with respect to byte-level encoding means that more than one possible encoding can be interpreted as 2.5.4.3, whether that behavior is desired or not. We have identified two ambiguities in the ASN.1 Basic Encoding Rules which can lead to this condition.

Leading-Zero Padding. An OID is not encoded using the textual representation of its digits and ‘.’ separating nodes; rather, the encoding uses base-128. Extra 0x80 bytes can be introduced to add leading zeroes to a node, e.g. 2.5.4.0003. OpenSSL’s OID resolver catches leading-zero padding — it does not interpret 2.5.4.03 and the like as the Common Name OID — but its textual representation of 2.5.4.03 is 2.5.4.3. Any implementation which mistakenly operates on this representation instead of the parsed OID is in for a nasty surprise. Worse, however, is CryptoAPI, whose OID parser happily strips off leading-zero padding, interprets 2.5.4.{0}*3 as 2.5.4.3, and resolves it to Common Name. Assuming the CA passes 2.5.4.3 (the textual form) into the final certificate as yet another unrecognized element of the X.509 Subject Name, IE will allow an attacker full access to any name he wants.

Integer Overflow. Since an OID is encoded as a base-128 integer, which is then converted to a native form, an ASN.1 parser which fails to take into account the fact that these integers are unbounded may fall victim to integer-overflow attacks. This is as simple as passing 2.5.4.18446744073709551619 as an OID, since $18446744073709551619 = 2^{64} + 3$. OpenSSL wisely uses a bignum library, and is not susceptible, but until recently, CryptoAPI expected integers to be no larger than a 64-bit unsigned long, and mistakenly recognized anything congruent to n in a field modulo 2^{64} as n . Thus, IE was easily tricked into accepting *anything* as a CN, simply by passing an OID that overflows.

Early null terminators in an X.509 Name can cause some APIs to recognize different Common Name values.³ Having explored the semantics of the CN field itself and how OIDs are recognized, we now turn to the parsing of the CN string. Since ASN.1 BER encodes its strings “Pascal-style”, with an explicit length field, rather than “C-style”, with a string ending at the first 0x00 value, a string with one or more NULL values is still valid BER as long as its length field is correct. Two problems arise from this: first, an incorrect length

³ This attack was independently and simultaneously discovered by Moxie Marlinspike, who presented it at Black Hat 2009 the same day that we did.

field can force reading or writing of data outside the blob being parsed, and second, once the binary data has been resolved to a string, C-style interpretation can cause unexpected behavior.

Consider an X.509 SN containing `CN=www.bank.com[NULL].badguy.com`. OpenSSL parses this as `CN=www.bank.com\x00.badguy.com`. Perl's `Crypt::OpenSSL::X509` module goes even further, eliding the NULL to read `CN=www.bank.com.badguy.com`. And OpenSSL's own `X509_NAME_get_text_by_NID` function terminates on the NULL. However, before we get ahead of ourselves, what do the CAs attempt to validate?

Validation typically occurs either by checking the WHOIS for the domain in question or by attempting to retrieve a selected file from the server identified by DNS. In both cases, with our example, the CA is being asked to validate a strangely named server under `badguy.com`. The technical contact listed in the WHOIS for `badguy.com` will presumably approve any request sent by the CA. But what if a DNS query is actually issued for `www.bank.com\x00.badguy.com`? If the client resolver strips the slash, the query becomes a lookup for `www.bank.comx00.badguy.com`, which the attacker simply hosts. If the DNS query contains a NULL byte, it will likely be rejected by the nameserver, since NULL is an invalid character in DNS. If the slash is not stripped, the query propagates to the attacker, who can then reply with an IP address.

Unfortunately, neither Firefox nor IE handle NULLS in the CN either; both interpret our malicious example above as `CN=www.bank.com`.

OpenSSL's mechanisms for emitting X.509 Subject Names are vulnerable to injection attacks. Although OpenSSL provides many scriptable command-line operations which can automate many aspects of PKI, it cannot automate the process of validating an identity. However, it can and does emit the X.509 Subject Name at various places from the command line, specifically to make it possible to audit the name as necessary. The CA need only write code to parse the text from OpenSSL's command line, rather than linking to OpenSSL's function calls or having to implement its own ASN.1 parser. While this approach is much easier, it does beg the question: Will the X.509 Subject Name parsed by the CA's text parser, after OpenSSL has munged it through its text filters, match the X.509 Subject Name ultimately contained with the PKCS#10 request, embedded within the generated X.509 Certificate, and delivered to the user's browser for validation?

If OpenSSL's default "compat" mode is used to emit X.509 Subject Names, not necessarily. (Three other modes, not enabled by default, are safe against the following attack.)

There are three points at which output from OpenSSL's command line interface might be parsed by a CA, looking to validate an X.509 Subject Name before certificate delivery to a client. The first, and easiest, is while signing a PKCS#10 Certificate Request, as the CN is emitted in a line that begins with "`subject=`". The client could also dump the PKCS#10 request to text and parse that. Or, the CA might sign the certificate no matter what, but suppress returning it to

the user unless it is successfully validated. In this case, the generated certificate can be dumped and the Subject Name extracted.

In all three cases, however, the CA is parsing ASCII characters, rather than the actual ASN.1 tree the browser will ultimately validate. What the CA knows of that structure, it extracts from the ASCII, by splitting on the presence of commas, slashes, and other so-called “escape characters” in the text. But what if the value of one of the non-validated elements in the X.509 Subject Name — OrganizationName (O), perhaps — itself contained escape characters? This constitutes yet another form of injection attack, directly akin to SQL injection (causing a variable in a SQL query to appear to be something more) or cross-site scripting (causing a variable in an HTML page to appear to be something more).

And, indeed, using an organizationName of Badguy Inc/ CN=www.badguy.com with an actual CN of www.bank.com results in a situation where the request and generated certificate appear, to a simple regular-expression match on the emitted ASCII, to have multiple CNs, with the first being www.badguy.com — and no way to tell that the *actual* CN, as denoted by OID, is www.bank.com. The CA’s business logic is the only line of defense, and as we have noted, many CAs do not employ manual review.

Textual CN injection is, however, probably the simplest of all the vulnerabilities listed in this paper to ameliorate. OpenSSL’s `nameopt` command-line flag prepends fields with their field names, which disambiguates the situation for both human readers and scripts. This option should be used in any automated CA system which relies on the OpenSSL command line.

A non-exploitable flaw exists in all of the filtering modes for OpenSSL < 0.9.8a, when a two- or four-byte-wide character set is filtered. The flaw is in the `do_buf` handler in `A_strex.c`, and involves the assumption that ASN.1 strings that contain 2- or 4-byte characters will be a multiple of 2 or 4 bytes. This is true for legitimate strings, but we can craft malicious ones for which it is not true; providing the handler with a string that thwarts its assumption causes the handler to fail. However, the pointer that OpenSSL uses to keep track of its place in the string is never actually written to; thus, what in any other codebase might be a trivial exploit merely becomes a denial of service.

4.3 PKCS#10-Tunneled SQL Injection

As mentioned earlier, ASN.1 allows many string types, with BMPString (UTF-16, supposedly minus certain characters) and UTF8String being the most flexible, but UniversalString is also worthy of analysis. This is a problem of insecurity through obscurity: since these encodings are rather abstruse, strings which use them may be injected into backend CA databases without sufficient validation. Unicode-based database injection attacks also come into play here. SQL injection into a CA’s database backend would be distinctly problematic, due to the special trust this particular data store has to the rest of the Internet.

4.4 PKCS#10-Tunneled ASN.1 Attacks

ASN.1 BER is tricky to parse, with many, many possibilities for consistent and predictably exploitable attack surfaces. The PROTOS project found a large number of vulnerabilities, via the SNMP consumer, but it is possible that some of the ASN.1 BER parsers found in commercial CA implementations were not covered in the 2002 PROTOS lockdown and thus are still vulnerable.

4.5 SSL Client Authentication Bypass

Many of the attacks in this paper have centered on vagaries with X.509 Subject Name validation. Bypassing the checks yields a certificate for somebody else's name. But what does it mean to have a certificate? For what purposes can it be used? In practice, most X.509 implementations support checking of a field called "Extended Key Usage", or EKU. EKUs come from two different sources. First, an EKU can show up in a leaf node as an explicit X.509 extension. In this context, a CA asserts the trustworthiness of a certificate. Second, an EKU can be applied out of band to a CA's root certificate, when the CA's root certificate is added to the browser's trust store. In this context, the browser manufacturer is limiting the trust semantics that a particular CA is allowed to express.

The most commonly used EKU is "Server Authentication", which states that a certificate may be used to validate a server to a client. But there are others, as we see specifically in the MD2 certificate we discussed transferring the self-signature; specifically, we are more interested in Client Authentication.

Most web sites use SSL certificates to authenticate the server to the client, followed by passwords to authenticate the client to the server. It is possible to use certificates to authenticate the client to the server as well, but this has a significant deployment and usability cost and is avoided by all but the most security-sensitive implementations. For these implementations, the user goes through the same CA experience as the server — except the X.509 Subject Name refers to *him*, not a website. A certificate-bearing SSL client, after authenticating the server, can then present his own certificate. If the certificate validates on the server — meaning that it chains back to a root trusted for Client Authentication — then some mapping will occur between the X.509 Subject Name and the application's own user database, and the user will be logged in.

These systems can fail in several major ways. First, one of the root certificates, used by the server to identify the client, might have its certificate compromised. For example, the MD2 attack discussed earlier would yield access to a VeriSign root cert with the Client Authentication EKU set in most trust stores. In the real world circa 2009, most systems are not intended to accept Client Authentication as asserted by a public CA. Instead, private CAs issue certificates to internal X.509 Subject Names, and those certs are accepted by servers in the infrastructure. However, this was not how SSL or X.509 was supposed to work. What was supposed to happen was that every user of the Internet would acquire strong cryptographic credentials from global CAs, which could be presented on demand in lieu of passwords. Through this path, a user at Microsoft could log

into a server at Yahoo, and neither Microsoft nor Yahoo would have to interact with each other's private CAs. To this day, there are systems that accept not only certificates from their own private CA, but any CA in their certificate store with the Client Authentication EKU. They even broadcast their list of accepted CAs, as part of the SSL client certificate exchange.

Thus, if we compromise this VeriSign cert, we may end up with an authentication bypass for some systems. (Since SSL client certificate use is generally limited to extremely secure systems, this is of particular concern, since the limited exposure to the vulnerability is in exactly the most sensitive systems.) There is solid evidence that this bug is exposed in the field, as per a quirk of SSL. The SSL protocol, in order to support client authentication, will not simply accept whatever client certificate a client intends to transmit. It instead provides extensive hints — on Windows, by emitting the list of all root certificate X.509 Issuer Names that have the “Client Authentication” bit set in the trust store. This CTL, or Certificate Trust List, appears to be extensive, as per the “global PKI” model that was originally hoped for. However, the server set up as described at [13] to allow one private key to enter, actually allows by default many CAs to express arbitrary X.509 Subject Names and thus gain access to the server.

Theoretically, Stevens and Sotirov, in their 2008 attack against MD5, were not actually capable of similar damage; the root certificate they compromised did not have the Client Authentication root set on it. Supposedly, this should mean that any certificate signed by Stevens and Sotirov's intermediate certificate should be unable to operate as a client certificate. In practice, actual behavior is subject to implementation quirks. The actual certificate they signed includes an EKU of “Digital Signature, Non-Repudiation, Certificate Sign, CRL Sign”. As per specification, this should not matter, as in-band EKUs are only supposed to be respected on leaf certificates that authenticate a given node, not on intermediates. What actually happens is murkier.

In order to prevent their malicious certificate from being respected, Stevens and Sotirov set the expiration date on their cert to late 2004. While this worked for many browsers, they found applications (a chat client, in their case) that did not realize it was important to check the expiration date. For these applications, their false intermediate worked perfectly. EKUs are an even more obscure part of the X.509 system. It is likely there are systems that ignore them too.

4.6 EV Hijacking

One of the more pernicious problems the web faces is the rise of phishing attacks. Put simply, which is the real Bank of America? Is it www.bankofamerica.com? www.bofa.com? www.bofabank.com? www.bank-of-america.com? Banks, in the real world, use physical trappings of wealth and police authority to regulate the abuse of their brand. Regulating the international DNS upon which the web is built is a much trickier problem. To deal with this pressing issue — widely exploited by phishers impersonating major banking establishments — Extended Validation certificates were developed. It was intended that EV certificates would simply not be issued without a thorough, manual validation of the IP behind the

claimed name. In return, the browser UI would be updated to herald the fully validated identity of the brand. Thus, even an attacker who could legitimately obtain a certificate for `https://www.bank-of-america.com` (since he was the owner, as per DNS), would not be able to emit the same trustworthy user interface, and his phishing attack would be foiled. This is what EV was designed to do, and it succeeds reasonably well.

When Stevens and Sotirov presented their research on using MD5 to generate certificate collisions, they admitted that the EV program was not vulnerable to their attacks. As they pointed out, Extended Validation certificates must, at all points in their validation chain, avoid MD5. Since Stevens and Sotirov were only able to generate attacks against certificate signatures executed with MD5 — and since their attack depended on automatic issuance of certificates, something that EV is specifically designed to avoid — they were technically correct in their assertion. The intermediate certificate they generated could not actually expose the correct bits to force the address bar green for an arbitrary domain.

However, EV was never actually designed to stop their attack, or any of the attacks described throughout this paper. The threat being mitigated was the \$12 registration of `www.bank-of-america.com` combined with the \$20 certificate, not the comparatively exotic MD5 or ASN.1 collision attack combined with the once-obscure DNS cache poisoning attack. EV offers no such defense. As Adam Barth and Collin Jackson wrote in “Beware of Finer Grained Origins”:

The browser’s scripting policy does not distinguish between HTTPS connections that use an Extended Validation (EV) certificate from those that use non-EV certificates. For example, PayPal serves `https://www.paypal.com/` using an EV certificate, but a principal who has a non-EV certificate for `www.paypal.com` can inject script into the PayPal login page without disrupting the browser’s Extended Validation security indicators.

An attacker who can synthesize a DV certificate for `www.bank.com` and DNS cache poison `www.bank.com` can act as a man in the middle, using port forwarding to negotiate EV certification with the actual server while allowing the attacker to both read the traffic flowing between a user and `www.bank.com` and to inject arbitrary data into that stream. This will work on effectively all browsers that have implemented EV SSL. It is difficult to impossible to imagine a defense that would not involve breaking the limited number of EV sites out there. EV was designed to stop phishing attacks, not failure of DV certificates.

5 Remediation

5.1 Immediate Steps

The following table summarizes immediate steps which browser manufacturers, cryptographic API maintainers, and certificate authorities should take to address the issues we have raised:

	Browser Manufacturers	Cryptographic API Manufacturers	Certificate Authorities
MD2RSA	Possibly, to support Cryptographic API changes	Yes, to change validation rules	Yes, to agree to resolution plan
Multiple Common Names	Possibly, to determine policy and measure exposure	Yes	Possibly, to determine policy and measure exposure
Inefficient ASN.1 bypass	Possibly, to determine policy and measure exposure	Yes	Possibly, to determine policy and measure exposure
Null terminator bypass	Possibly, to determine policy and measure exposure	Yes	Possibly, to determine policy and measure exposure
OpenSSL “compat” bypass	No	Yes, definitely for SSL, possibly for others	Yes, to determine if commercial CA implementations have similar string parsing layers
PKCS#10 SQL injection	No	Possibly, to add support for filtering at the API layer	Yes
PKCS#10 ASN.1 exploitation	No for the major browsers, since presumably they've already had to lockdown their ASN.1 engine	Possibly, to make sure that PKCS#10 is being parsed with a post-PROTOS hardened library	Possibly, to make sure that PKCS#10 is being parsed with a post-PROTOS hardened library
Client certificate bypass	No	Yes, to control the list of certificates that a web server will insert into the CTL	No
EV bypass	Yes, to manage PR/understanding around the purpose of EV	No	Possibly, to manage PR, and to perhaps create a “blacklist” of EV certified names for which CAs will not issue a certificate

5.2 EV Remediation

Although the EV attack we describe is particularly pernicious, there are two defenses that might be worth considering. A “`http://`” scheme could be developed, which would force content to only be loaded from an EV certificate.

However, this would require modifications at the crypto layer to support a new X.509 element, declaring that a certificate could only emit the “green bar” positive feedback experience when the `httpiev://` method was used. Otherwise, an attacker could simply use Moxie Marlinspike’s method of forwarding a user from `http://` to `https://` instead of `httpiev://` and acquire 99% of the positive feedback while still being able to use his compromised DV certificate [7].

Another defense that might be interesting to explore would be a blacklist of names that, once issued via EV, should never be issued via DV. This paper describes many ways around the CA system. It might be interesting to have an emergency check, just for EV, before a certificate is sent to a user that might have the same X.509 Subject Name as an issued EV certificate.

6 Future Work

We continue to investigate certificate chain validation. It has not gone unnoticed that X.509, which was supposed to be a fully delegatable system, never actually found a safe way to delegate signing authority across chunks of DNS namespace. However, we believe we can still attack chain validation. There are multiple ways to find an issuer in X.509, and we have ways of creating valid certificates with near-arbitrary subject names. If we can find a validation path that confuses our certificate (which we do have the key for, but does not have any special capabilities) with another certificate (which we do not have the key for, but does have special capabilities) due to them sharing the exact same X.509 Subject Name, then we believe we can generate fully Root CA equivalent certificates without the still-temporarily-impractical MD2 attacks.

Another area we are investigating are the three Issuer paths: Authority Information Access, Authority Key Identifier, and the actual X.509 Issuer Name. Any and all of these can be used to generate collisions.

The wide array of string and length encodings available in ASN.1 also provides a rich attack surface. We suspect that it is possible to cause two encoders to read two entirely different ASN.1 trees by cleverly manipulating length fields, but have not yet developed a proof of concept.

Finally, there may be interesting attacks down the path of Internationalized Domain Names. For the most part, IDNs are blocked at major CAs due to the homograph attacks of Eric Johanson and the Shmoo Group in 2005 [5]. However, Moxie Marlinspike showed in early 2009 that wildcards in certificates allow IDN characters to pass validation [7]. We must consider new attacks down this path, particularly with alternate representations that may collapse back to wildcards.

References

1. Open1x IEEE 802.1x open source implementation,
<http://open1x.sourceforge.net/>
2. Dierks, T., Rescorla, E.: The transport layer security (tls) protocol (August 2008),
<http://tools.ietf.org/html/rfc5246>

3. Gutmann, P.: X.509 style guide (October 2000),
<http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt>
4. Jackson, C., Barth, A.: Beware of finer-grained origins. In: Web 2.0 Security and Privacy, W2SP 2008 (2008)
5. Johanson, E.: The state of homograph attacks (2005),
<http://www.shmoo.com/idn/homograph.txt>
6. Kaliski, B.: Pkcs #1: Rsa encryption (March 1998),
<http://tools.ietf.org/html/rfc2313>
7. Marlinspike, M.: New tricks for defeating ssl in practice (July 2009),
<http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf>
8. Marlow, S.: Happy user guide (2001),
<http://www.haskell.org/happy/doc/html/sec-AttributeGrammar.html>
9. Muller, F.: The md2 hash function is not one-way. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 214–229. Springer, Heidelberg (2004)
10. neon HTTP and WebDAV client library, <http://www.webdav.org/neon/>
11. Pilosov, A., Kapela, T.: Stealing the internet: An internet-scale man-in-the-middle attack. In: DEFCON, vol. 16 (August 2008)
12. Rning, J., Laakso, M., Takanen, A., Kaksonen, R.: Protos - systematic approach to eliminate software vulnerabilities (2002)
13. Singh, S.: Certificate trust list not being honored by iis 5.0/6.0/7.0 (December 2007),
http://blogs.msdn.com/saurabh_singh/archive/2007/12/07/certificate-trust-list-not-being-honored-by-iis-5-0-6-0-7-0.aspx
14. Stevens, M., Lenstra, A., Weger, B.: Chosen-prefix collisions for md5 and colliding x.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
15. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate. In: Cryptology ePrint Archive, Report 2009/111 (2009),
<http://eprint.iacr.org/>
16. Bacula the open source network backup software solution,
<http://www.bacula.org/en/>
17. Claws Mail: the user-friendly lightweight and fast email client,
<http://www.claws-mail.org/>
18. Thomsen, S.S.: An improved preimage attack on md2. In: Cryptology ePrint Archive, Report 2008/089 (2008), <http://eprint.iacr.org/>
19. US-CERT. Vulnerability note vu#800113: Multiple dns implementations vulnerable to cache poisoning. US-CERT Vulnerability Notes Database (2008),
<http://www.kb.cert.org/vuls/id/800113>
20. Wang, X., Feng, D., Lai, X., Yu, H.: Collisions for hash functions md4, md5, haval-128 and ripemd. In: Cryptology ePrint Archive, Report 2004/199 (2004),
<http://eprint.iacr.org/>
21. GNU Wget, <http://www.gnu.org/software/wget/>

Three-Round Abuse-Free Optimistic Contract Signing with Everlasting Secrecy

(Extended Abstract)

Xiaofeng Chen¹, Fangguo Zhang², Haibo Tian², Qianhong Wu^{3,4},
Yi Mu⁵, Jangseong Kim⁶, and Kwangjo Kim⁶

¹ Key Laboratory of Computer Networks and Information Security,
Ministry of Education, Xidian University, P.R. China

² School of Information Science and Technology, Sun Yat-sen University, P.R. China

³ Department of Computer Engineering and Mathematics,
UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, Catalonia

⁴ Key Laboratory of Aerospace Information Security and Trusted Computing,
Ministry of Education, Wuhan University, P.R. China

⁵ School of Computer Science and Software Engineering,
University of Wollongong, Australia

⁶ Department of Computer Science, KAIST, Korea

Abstract. We introduce the novel notion of Verifiable Encryption of Chameleon Signatures (VECS), and then use it to design a three-round abuse-free optimistic contract signing protocol.

Keywords: Verifiable encryption, Chameleon signatures, Contract signing.

1 Introduction

Contract signing is an important part of business transactions. Fairness is a basic requirement for contract signing. However, most of the existing contract signing protocols only focus on the fairness while ignoring the privacy of the players. We argue that the privacy of the players is closely related to the fairness. For example, if one player or the trusted third party can reap profits at the expense of the other player by intentionally releasing some useful information related to the contract, then the contract signing protocols cannot achieve the true fairness.

Garay et al. [9] first introduced the notion of abuse-free contract signing, which ensures neither party can prove to others that he is capable of choosing whether to validate or invalidate the contract in any stage of the protocol. To illustrate by example, suppose Bob and Carol are two potential competitors who will sign a contract with Alice. If Alice can convince Carol that Bob would like to sign a contract m with her, she may obtain a better contract m' from Carol. In this sense, a contract signing protocol without the property of abuse-free cannot ensure the fairness for both parties. However, it seems that all the efficient

contract signing [1,2,4,7] based on the state-of-the-art technique of verifiable encryption of digital signatures (VEDS) are not abuse-free since VEDS is universal verifiable.

On the other hand, we should consider the misbehavior of the trusted third party in contract signing protocols. Although the third party is (by definition) trusted, it is difficult to find a fully trusted third party in the internet. Asokan et al. [3] and Garay et al. [9] introduced the property of accountability in contract signing, *i.e.*, it can be detected and proven if the third party misbehaved. However, all of the existing contract signing protocols do not consider the following misbehavior of the third party: if the third party can know all the information related a contract such as the contract content and the corresponding signatures of two parties, he may sell this associated commercial secret to an interested party. In this sense, it is unfair for both parties, though the contract signing protocol is fair as defined.

In this paper, we first introduce a novel notion named Verifiable Encryption of Chameleon Signatures (VECS), which can be referred to as a special instance of VEDS. Meanwhile, we use this notion to design an efficient optimistic contract signing protocol, which enjoys the properties of completeness, fairness, abuse-freeness, accountability, and invisibility of the third party. The distinguishing property of our signing protocol is the *everlasting secrecy* about the contract against the third party. That is, the third party cannot know any useful information of the contract in any stage of the protocol, which prevents him from illegally selling the commercial secret to any interested party. Moreover, our exchange protocol is only three-pass in the normal situation and thus much efficient for practical use.

2 Verifiable Encryption of Chameleon Signatures

2.1 Formal Definition

Definition 1. (*Verifiable Encryption of Chameleon Signatures*) A secure VECS scheme consists of a five tuple $(\mathcal{PG}, \mathcal{KG}, \mathcal{SG}, \mathcal{VE}, \mathcal{SR})$.

- **System Parameters Generation \mathcal{PG} :** An efficient probabilistic algorithm that, on input a security parameter k , outputs the system parameters SP .
- **Key Generation \mathcal{KG} :** An efficient algorithm that, on input the system parameters SP , outputs a secret/public key pair (sk, pk) for each user.
- **Signature Generation \mathcal{SG} :** An efficient probabilistic algorithm that, on input a label L , the public key pk_V of the verifier V , the secret key sk_P of the prover P , a message m , and an auxiliary random element r , outputs a signature σ on the chameleon hash value $h = \text{Hash}(L, m, r, pk_V)$.
- **Verifiable Encryption \mathcal{VE} :** A non-interactive protocol between the prover P and the verifier V . Let (E, D) be the encryption/decryption algorithm as well as the public/secret key of a secure public key encryption system. Let $V_P(E, \sigma, r)$ denote the output of V when interacting with P on input (E, σ, r) .

- **Signature Recovery \mathcal{SR} :** An efficient deterministic algorithm that, on input the decryption algorithm D and the ciphertext $V_P(E, \sigma, r)$, outputs a chameleon signature (σ, r) on message m with respect to the public key pkv .

2.2 A Concrete Construction from RSA Signatures

Ateniese has proposed various efficient VEDS schemes [4]. Since a chameleon signature scheme is a general construction, it can naturally be used in the Ateniese's VEDS schemes, which results in various VECS schemes. Note that we should use the key-exposure-freeness chameleon signature schemes [5,6,8] in order to avoid the key exposure problem of chameleon hashing.

There are three parties, a prover P , a verifier V , and a trusted third party T in our scheme.

- **System Parameters Generation \mathcal{PG} :** Let t and k be security parameters. For $i = 1, 2$, define $n_i = p_i q_i$ with the two safe primes $p_i = 2p'_i + 1$ and $q_i = 2q'_i + 1$ in the set $\{2^{k-1}, \dots, 2^k - 1\}$, where p'_i, q'_i are primes. Let $\mathcal{H}_1 : \{0, 1\}^* \rightarrow \{0, \dots, 2^{2k} - 1\}$ and $\mathcal{H}_2 : \{0, 1\}^* \rightarrow \{0, \dots, 2^t\}$ and $\mathcal{H}_3 : \{0, 1\}^* \rightarrow \mathbb{Z}_{n_1}$ be three collision-resistant hash functions.
- **Key Generation \mathcal{KG} :** For $i = 1, 2$, choose a random prime integer $e_i > 2^t$ which is relatively prime to $\phi(n_i) = (p_i - 1)(q_i - 1)$, and compute d_i such that $e_i d_i \equiv 1 \pmod{p'_i q'_i}$. The public key of P is (n_1, e_1) and his secret key is (p_1, q_1, d_1) . The public key of V is (n_2, e_2) and his secret key is (p_2, q_2, d_2) . T randomly chooses an element $\tilde{g} \in \mathbb{Z}_{n_1}$ and computes $g = \tilde{g}^2 \pmod{n_1}$. The public key of T is $(g, y = g^x \pmod{n_1})$ and his secret key is x .
- **Signature Generation \mathcal{SG} :** Let L be a label, define $J = \mathcal{H}_1(L)$. To sign a message m , P chooses a random integer $u \in_R \mathbb{Z}_{n_2}^*$ and computes the chameleon hash value $M = J^{\mathcal{H}_2(m)} u^{e_2} \pmod{n_2}$. He then computes the signature $\sigma = \mathcal{H}_3(M)^{d_1} \pmod{n_1}$ on message M .
- **Verifiable Encryption \mathcal{VE} :** P and V perform the following protocol:
 1. P computes $\mathcal{C}_1 = (m||u)^{2e_2} \pmod{n_2}$, where $||$ denotes concatenation.
 2. P randomly chooses an integer r and encrypts the chameleon signature σ via the ElGamal encryption scheme with T 's public key y . That is, P computes $\mathcal{C}_2 = (K_1, K_2, c, s)$, where $K_1 = \sigma^2 y^r \pmod{n_1}$, $K_2 = g^r \pmod{n_1}$, $c = \mathcal{H}_3(M||y^{e_1 r}||g^r||y^{e_1}||g||(y^{e_1})^t||g^t)$, and $s = t - cr$.
 3. P sends the ciphertext $(\mathcal{C}_1, \mathcal{C}_2)$ to V .

V firstly decrypts \mathcal{C}_1 to obtain the pair (m, u) , and then computes $M = J^{\mathcal{H}_2(m)} u^{e_2} \pmod{n_2}$, $W = K_1^{e_1} \mathcal{H}_3(M)^{-2} \pmod{n_1}$, and

$$c' = \mathcal{H}_3(M||W||K_2||y^{e_1}||g||(y^{e_1})^s W^c ||g^s K_2^c).$$

If $c' = c$, V accepts the fact that \mathcal{C}_2 is a valid T -verifiable encryption of P 's chameleon signature on message m .

- **Signature Recovery \mathcal{SR} :** In case of dispute, T can compute $\sigma^2 = K_1 / K_2^x \pmod{n_1}$ and then get σ .

3 Secret Abuse-Free Contract Signing

3.1 Security Model

Asokan et al. [2] presented a formal security model for fair signature exchange, which is also suitable for contract signing. In the optimistic two-party contract signing, there are two players A and B , and a trusted third party T that acts as a server: it receives a request from a client, updates its internal state and sends a response back to the client. We assume that all participants have secret/public keys which will be specified later.

We assume that communication channels between any two participants are *confidential*, which means that eavesdroppers will not be able to determine the contents of messages in these channels. Moreover, we assume that the communication channel between any player and T is *resilient*. The resilient channel assumption leads to an asynchronous communication model without global clocks, where messages can be delayed arbitrarily but with finite amount of time.

Since the misbehavior of dishonest participants could lead to a loss of fairness, we consider the possible misbehavior of the participants in the contract signing. Firstly, although T is by definition trusted, T may collude with one party to weaken the fairness, or gain some benefits by selling the commercial secret of the contract. Therefore, T must be *accountable* for his dishonest actions, *i.e.*, it can be detected and proven if T misbehaves. Secondly, A or B may reap benefits at the expense of the other party. The abuse-freeness contract signing protocol can only partially solve this problem. For example, a dishonest A can execute the **Abort** protocol after correctly executing the **Exchange** protocol with B [10]. As a result, B obtains A 's signature while A obtains B 's signature and the abort-token. Trivially, the output of the protocol violates the original definition of fairness. This means that Asokan et al.'s security model is not perfect. The reason is that it does not consider the misbehavior of A and B . Therefore, we should define the accountability of A and B , *i.e.*, it can be detected and proven if A and B misbehaves. Moreover, It can be a part of the agreed contract content for how to punish the dishonest party.

The security properties of contract signing are defined in term of completeness, fairness, abuse-freeness, accountability, T invisibility [2,9]. Besides, we define a new property named T secrecy. We argue that a contract and the corresponding signatures of two players should be a commercial secret and T cannot reveal it to outsiders for some benefits in any stage of the protocol.

- **Completeness:** It is infeasible for the adversary to prevent honest A and B from successfully obtaining a valid signature (or the non-repudiation token) of each other. The adversary has the signing oracles that can be queried on any message except the contract. The adversary can interact with T , but cannot interfere with the interaction of A and B , except insofar as the adversary still has the power to schedule the messages from A and B to T .
- **Fairness:** We consider a game between an adversary and an honest party. Generally, we let the adversary play the role of the corrupt party, who completely controls the network, arbitrarily interacts with T , and arbitrarily

delays the honest party's requests to T . We argue that the misbehavior of the adversary may weaken the fairness. So, if the honest party can provide a proof that the adversary misbehaves, then he has the power to validate or invalidate the contract for the punishment of the adversary. In this sense, the fairness means that it is infeasible for the adversary to obtain the honest party's signature on a contract, while without allowing the honest party to obtain the adversary's signature or a proof that the adversary misbehaves.

- **Abuse-freeness:** It is infeasible for one party at any point in the protocol to be able to prove an outside party that he has the power to terminate (abort) or successfully complete the contract.
- **Accountability:** It can be detected and proven if any participant misbehaves.
- T **invisibility:** It is infeasible to determine whether T has been involved in the protocol or not.
- T **secrecy:** It is infeasible for T to obtain any useful information about the contract in any stage of the protocol.

3.2 Our Protocol

In this section, we use the proposed VECS to present an efficient abuse-free contract signing protocol. We first give some notations. Let \mathcal{H} be a key exposure free chameleon hash function. Denote by $Sig(SK_X, M)$ the signature on message M with the secret key SK_X of the party $X \in \{A, B, T\}$; Denote by $O_B(E, \sigma_A, PK_T)$ a verifiable encryption of A 's signature σ_A under T 's public key PK_T . Our abuse-free contract signing protocol has three sub-protocols: **Exchange**, **Abort**, and **Resolve**. In the normal case, only the exchange protocol is executed.

Suppose A and B have agreed on a message $\mathcal{M} = (m, r_A, r_B)$, where m is a common contract and (r_A, r_B) are two random integers. We do not describe this agreement in details here and it may require a number of rounds of communication between A and B through an authenticated channel. Moreover, this agreement should not achieve the non-repudiation property, *i.e.*, neither party should generate any non-repudiation token on the agreed message.

Exchange Protocol

1. A computes the chameleon hash value $h_A = \mathcal{H}(m, r_A, PK_B)$ and the signature $\sigma_A^* = Sig(SK_A, h_A || T)$, where $||$ denotes concatenation. A then computes the ciphertext $C = O_B(E, \sigma_A^*, PK_T)$ and sends it to B .
2. If C is invalid, B quits. Otherwise, B computes the signature $\sigma_B = Sig(SK_B, h_B)$ on the chameleon hash value $h_B = \mathcal{H}(m, r_B, PK_A)$ and then sends σ_B to A .
3. If σ_B is invalid, A runs the **Abort** protocol. Otherwise, A computes the signature $\sigma_A = Sig(SK_A, h_A)$ and sends it to B . If σ_A is not valid, B runs the **Resolve** protocol.

Abort Protocol

1. A computes the signature $Sig(SK_A, \text{abort}||\mathcal{C})$ on message “abort|| \mathcal{C} ” and then sends $(\mathcal{C}, Sig(SK_A, \text{abort}||\mathcal{C}))$ to T . If the signature is valid and B has not resolved, T issues an abort-token $\mathcal{AT} = Sig(SK_T, Sig(SK_A, \text{abort}||\mathcal{C}))$ to A and stores it. The abort token is not a proof that the exchange has been aborted, but a guarantee by T that it has not and will not execute the **Resolve** protocol.

2. If B has resolved, T sends A the stored value $\hat{\sigma}_B$ in the **Resolve** protocol.

Resolve Protocol

1. B firstly sends T the triple $(\mathcal{C}, h_A, \hat{\sigma}_B)$, where $\hat{\sigma}_B = Sig(SK_B, \text{resolve}||A||h_A)$ denotes the *resolved* signature of B . Generally, it is no difference with an ordinary signature $Sig(SK_B, \text{resolve}||A||h_A)$ of B on message “ $\text{resolve}||A||h_A$ ”. Additionally, it also denotes $Sig(SK_B, m)$ on condition that only A can provide a pair (m, r_A) which satisfies $h_A = \mathcal{H}(m, r_A, PK_B)$.
2. If A has aborted, T then sends the abort-token \mathcal{AT} to B . Else, if \mathcal{C} is a valid T -verifiable encryption of A ’s signature on message h_A and $\hat{\sigma}_B$ is valid, T decrypts \mathcal{C} to obtain σ_A^* and sends it to B .
3. T stores the value $\hat{\sigma}_B$.

3.3 Misbehavior in the Protocol

Since the set of the possible output for A and B is $\{\sigma_B, \hat{\sigma}_B, \mathcal{AT}\}$ and $\{\sigma_A, \sigma_A^*, \mathcal{AT}\}$, respectively. Therefore, the possible output of our proposed protocol is as follows:

- Case 1: A obtains σ_B and B obtains σ_A . This means that both parties are honest.
- Case 2: A obtains σ_B and B obtains σ_A^* . This means that B successfully runs the **Resolve** protocol at some point after sending σ_B .
- Case 3: A obtains $\hat{\sigma}_B$ and B obtains σ_A^* . This means that A has already sent \mathcal{C} to B , and then B runs the **Resolve** protocol before A aborted.
- Case 4: Both A and B obtain \mathcal{AT} . This means that A has already sent \mathcal{C} to B , and then runs the **Abort** protocol at some point before B resolved.
- Case 5: A obtains σ_B and B obtains \mathcal{AT} . This means that A has received σ_B and then runs the **Abort** protocol before B resolved. If this case happens, we claim that A misbehaves in the protocol.
- Case 6: A obtains \mathcal{AT} and B obtains σ_A . This means that A runs the **Abort** protocol after sending σ_A to B . If this case happens, we also claim that A misbehaves.
- Case 7: A obtains \mathcal{AT} and B obtains σ_A^* . This means that both A and B successfully runs the **Abort** and **Resolve** protocol, respectively. If this case happens, we claim that the T misbehaves.
- Case 8: A obtains $\hat{\sigma}_B$ and B obtains \mathcal{AT} . Due to the fact that B obtains the abort-token only when A has obtained the abort-token, this case will not happen if the T is honest. Therefore, we also claim that the T misbehaves in this case.

- Case 9: A obtains $\hat{\sigma}_B$ and B obtains σ_A . If this case happens, we claim that A misbehaves because B cannot obtain σ_A unless A has obtained σ_B successfully.

If the first four cases occur, the protocol achieves the fairness since both parties obtain either the signature of each other, or the abort token. Since the chameleon signature is not universal verifiable, σ_B means nothing if B does not perform the denial protocol of chameleon signatures. On the other hand, A is not allowed to run the **Abort** protocol after having received σ_B . Similarly, A is not allowed to run the **Abort** protocol after sending σ_A to B . Moreover, A should never send σ_A to B unless A has obtained σ_B successfully. That is, if the case 5, or case 6, or case 9 occurs, it is a proof that A misbehaves. If the case 7 or 8 occurs, then T must be accountable for his misbehavior.

4 Security Analysis of the Contract Signing Protocol

Due to the properties of non-repudiation and non-transferability of chameleon signatures, the proposed contract signing protocol satisfies the completeness and abuse-freeness, respectively. Also, as discussed in section 3.3, it is trivial that the proposed contract signing protocol satisfies the accountability. Due to the space consideration, we only focus on the fairness, T invisibility and T secrecy.

Theorem 1. *The proposed contract signing protocol satisfies the property of fairness.*

Proof. We first prove the fairness for A . Consider an honest A playing against a dishonest B . We say that B wins the game if and only if either B obtains σ_A while A does not obtain σ_B , or B obtains σ_A^* while A obtains neither σ_B nor $\hat{\sigma}_B$. Assume A does not obtain σ_B , A must run the **Abort** protocol at some point after sending C to B and thus B cannot obtain σ_A . If B does not run the **Resolve** protocol before A aborted, then both parties obtain the abort-token AT . Else, B can obtain σ_A^* from the T . However, it ensures that A can also obtain $\hat{\sigma}_B$ from T . Therefore, the successful probability for B to win the game is negligible.

We then prove the fairness for B . Consider an honest B playing against a dishonest A . We say that A wins the game if and only if either A obtains σ_B while B obtains neither σ_A nor σ_A^* , or A obtains $\hat{\sigma}_B$ while B does not obtain σ_A^* . Firstly, we argue if A obtains $\hat{\sigma}_B$, then B must obtain σ_A^* unless the T is dishonest. Secondly, assume B does not obtain σ_A , so B must run the **Resolve** protocol at some point after sending σ_B to A . If A does not run the **Abort** protocol before B resolved, then B can obtain σ_A^* from the T . Else, B can obtain the abort-token AT . However, it is a proof that A misbehaves in the protocol and A must be accountable for this. Therefore, the successful probability for A to win the game is negligible. \square

Theorem 2. *The proposed contract signing protocol satisfies the property of T invisibility and T secrecy.*

Proof. Note that the distribution of σ_A and σ_A^* is computationally indistinguishable. Similarly, the distribution of σ_B and $\hat{\sigma}_B$ is also computationally indistinguishable. Therefore, it is impossible to determine whether T has been invoked in the protocol or not. On the other hand, note that the message $\mathcal{M} = (m, r_A, r_B)$ is agreed beforehand and never revealed in any stage of the protocol. Moreover, the chameleon signature is not universal verifiable. Therefore, T cannot obtain any useful information about the contract in the protocol. \square

5 Conclusions

In this paper, we first introduce the notion of Verifiable Encryption of Chameleon Signatures (VECS). We then use the notion to design a secret abuse-free optimistic contract signing protocol, which is only three-pass in the normal situation. Moreover, we prove that our protocol achieves the desired security properties.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 60970144, 60773202, 60970114, 60970115, 60970116), Guangdong Natural Science Foundation (No. 8451027501001508), Program of the Science and Technology of Guangzhou, China (No. 2008J1-C231-2).

References

1. Asokan, N., Shoup, V., Waidner, M.: Asynchronous protocols for optimistic fair exchange. In: IEEE Symposium on Security and Privacy, pp. 86–99 (1998)
2. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
3. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. IEEE Journal on Selected Areas in Communications 18(4), 593–610 (2000)
4. Ateniese, G.: Verifiable encryption of digital signatures and applications. ACM Transaction on Information and System Security 7(1), 1–20 (2004)
5. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
6. Ateniese, G., de Medeiros, B.: On the key-exposure problem in chameleon hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005)
7. Bao, F., Deng, R., Mao, W.: Efficient and practical fair exchange protocols with off-Line TTP. In: IEEE Symposium on Security and Privacy, pp. 77–85 (1998)
8. Chen, X., Zhang, F., Kim, K.: Chameleon hashing without key exposure. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 87–98. Springer, Heidelberg (2004)
9. Garay, J.A., Jakobsson, M., MacKenzie, P.: Abuse-free optimistic contract signing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 449–466. Springer, Heidelberg (1999)
10. Shmatikov, V., Mitchell, J.C.: Analysis of abuse-free contract signing. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 174–191. Springer, Heidelberg (2001)

Designing for Audit: A Voting Machine with a Tiny TCB

(Short Paper)

Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin

Johns Hopkins University, Baltimore MD 21218, USA

Abstract. Thoroughly auditing voting machine software has proved to be difficult, and even efforts to reduce its complexity have relied on significant amounts of external code. We design and implement a device that allows a voter to confirm and cast her vote while trusting only 1,034 lines of ARM assembly. The system, which we develop from scratch, supports visually (and hearing) impaired voters and ensures the privacy of the voter as well as the integrity of the tally under some common assumptions. We employ several techniques to increase the readability of our code and make it easier to audit.

1 Introduction

Electronic voting has become the instrument of democracy in many parts of the world as a result of historic dilemmas stemming from the paper ballot and a perceived voter preference for touch-screen machines [16]. However, in the last 6 years, a multitude of studies have analyzed the security of electronic voting devices used in elections [10,4], and each has found significant security flaws in every revision of their software. Consequently, it has become clear that it is extremely difficult to exhaustively audit or ensure security guarantees of the voting software used in current elections.

Several researchers have made significant progress toward designing voting systems that may provide increased assurances that the software is free of vulnerabilities and backdoors. Two of the most notable works are those of Yee *et al.* [20,19] and Sastry *et al.* [13]. Yee *et al.* reduce the complexity of voting software by prerendering the electronic ballot design [20] and write a full voting machine application, Pvote, in only 460 lines of original Python code [19]. It is clearly more feasible to audit and ensure the robustness of such a small piece of software than that of the Diebold AccuVote-TSX, for example, which contains over 65,000 lines of C++ [4], or the Sequoia Edge, which consists of over 124,000 lines of C [1]. Sastry *et al.* physically separate modules of a voting system in hardware to allow security properties to be verified more easily by examining the individual components [13]. Their prototype contains only 5,085 lines of trusted code. While both of these studies make significant advances in the state of voting machine software, they also rely on the integrity of several large pieces of critical, external code, including libraries, operating systems, compilers, and interpreters, and 5085 lines remains a fairly large number for audit.

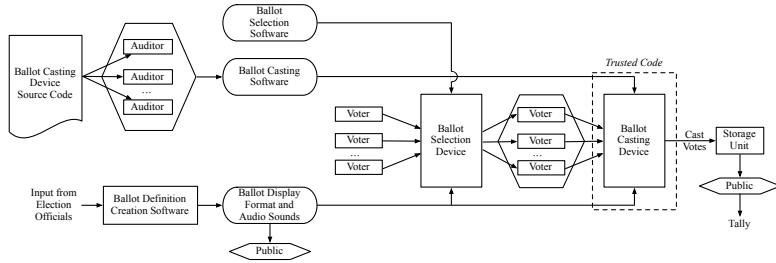


Fig. 1. Flow of data through an election. Hexagons represent reviews by a large number of parties.

Carefully auditing *all* components that contribute to the development and execution of a piece of software that casts votes is essential. Consider pygame, for example, a multimedia library used by Pvote. It contains 28,660 lines of code and numerous bugs have been reported on it including possible inappropriate memory accesses, which may allow exploitation of Pvote itself¹. Similarly, along with operating systems and code interpreters, the use of a compiler enables potential attack vectors to the voting application as the compiler could create a vulnerable or malicious executable [17].

In this work, we create a new vote casting system that drastically decreases the amount of software that must be trusted. Utilizing the idea of Frogs [3], we separate the device that voters use to confirm and cast their votes from the hardware they use to select the votes. Then, we build the code for the confirmation and casting device from the foundation up. That is, we compose every byte of code or data that is written to the device, and we eliminate the need to rely on the integrity of a compiler by writing purely in assembly language. We significantly reduce the complexity of this confirmation device by pushing non-sensitive and verifiable functions to stages before and after the ballot is cast. Our device supports visually impaired voters, prevents tampering with stored ballots, and ensures that voters can only cast one ballot, under some common assumptions. In total, its code is written in 1,034 lines of ARM assembly. Here we summarize the architecture and a few aspects of the implementation of our vote casting system with a more complete description and details in the full version of the paper [7].

2 High-Level Approach

One of the primary objectives of our work is to reduce the amount of code that needs to be trusted for voters to electronically cast votes. We take an approach of reducing trust by redundantly dividing it among many entities.

The basic flow of data in our architecture is illustrated in Figure 1. Many months before the election, the source code for the vote casting device is

¹ <http://pygame.motherhamster.org/bugzilla/>

published or, if preferable [8], made available to a large but select group of auditors. Then, auditing on a wide variety of systems is encouraged. The idea is that even if a number of systems have compromised components, at least one system will not and can perform an impartial audit. Finally, auditors or other individuals assemble the code. Because assembling is (mainly) a one-to-one process, binaries can be compared and analyzed until consensus is reached on a correct image for the device. Hence, we eliminate trust in any single system or component by distributing the auditing and binary generation in this way.

In addition to enabling verification of a correct binary, redundantly dividing trust also provides us with a means of drastically simplifying the sensitive vote casting process without significantly increasing threats to the other steps of an election. This is the general technique used by Bruck *et al.* with Frogs [3] and Yee *et al.* with prerendered user interfaces [20,19] and allows us to remove the vast majority of vote processing from the ballot casting device.

Several months prior to the election, the display format for the ballot is standardized and publicized. This specification includes not only the exact visual appearance and sounds of the ballot as they are presented to the user but also their raw format as it is sent directly to the display and audio hardware of the casting device. The public can review the format and ensure that it is clear and accurate. Then, on the day of the election, each voter is given a memory card for storing her vote when she enters the polls. She selects her votes using a ballot selection device, and her ballot is recorded to her card in its raw format. She then passes her card on to the ballot casting machine, and the card's data is sent directly to the display and audio hardware of the device so she may view it and confirm her selections. This step also allows the voter to detect any dishonest behavior of the ballot selection device, and in aggregate, the review of large numbers of voters minimizes the probability of undetected malicious activity. From this point, if the voter chooses to cast, the same, unprocessed data that was displayed is anonymously recorded to non-volatile ballot storage along with some authentication information. Again, because an abundance of people can independently interpret and process the information, the anonymous ballots can be made public by disclosing the storage data exactly as it is, in raw form. Despite the data's increased complexity, the risk of undetected error or dishonesty is minimal due to the extensive number of people potentially reviewing it. By moving all the untrusted processing to before and after the point when the ballot is cast in this way, we greatly simplify this critical point in the election process.

3 Our Voting System

We now outline the functionality of our voting system and describe aspects of the implementation of our ballot casting device on an LPC2148 ARM microcontroller interfacing with additional hardware².

² All of our code is available at http://cs.jhu.edu/~ryan/min_tcb_voting/

3.1 Functionality Overview

The voting process involves 3 primary components: an authentication device, a ballot selection device, and a ballot casting device. Prior to the election, each political party generates a set of authentication and encryption keys. These keys are transferred to the ballot casting device via a smart card the morning of the election and stored in the device's RAM. An alternative approach could keep the keys exclusively on the smart cards and compute cryptographic operations on the cards themselves [3] although it requires more smart-card interfacing hardware. We compute all cryptographic operations on the ballot casting device to make the device's code easy to adapt to either approach.

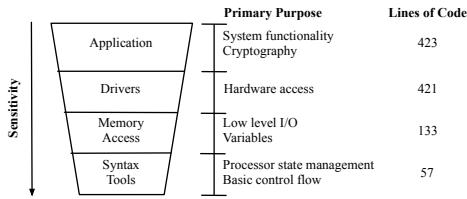
On election day, each voter first obtains a voter card from a poll worker who authenticates the card using the authentication device. An authenticated card, which we implement using a memory smart card, contains a unique, encrypted authenticator that acts as the voter's ticket to vote. (Here, encryption prevents the voter from potentially obtaining her authenticator and selling her vote by identifying the authenticator and her corresponding vote during tallying.) Next, the voter takes her card to vote as described in Section 2. She selects her votes on the ballot selection device, which writes her ballot to her voter card. As she then transfers her card to the ballot casting device, that device presents visual and audio versions of the ballots and gives the option to cast. If the voter casts her ballot, the raw data that was presented is written to a randomly chosen, empty location on the device's non-volatile storage along with the voter's decrypted authenticator. The device also updates a counter of cast votes and signs all data written for the vote using a key from each political party. Unfortunately computing the decryptions and all the signatures takes several minutes on our device with its 60 MHz processor and 20 MHz memory bus, but faster or more specialized hardware could preclude this problem. Lastly, the device erases the voter's card, which is returned for reuse.

3.2 Structure and Readability

We organize our code into 4 distinct layers to simplify the task of auditing. Our basic approach is to compact security-sensitive functionality and common syntax abstractions and functions into *small* portions of code at the lowest layers. Then we write the larger, higher layers using these tools so that their implementation utilizes fewer, and more familiar constructs. This structure also contributes significantly to a shorter code length by maximizing code reuse.

The structure of our code is illustrated in Figure 2. Each layer is only permitted to use a subset of the instructions and functions that are used by the layer below it. In turn each layer exports functionality and syntax in the form of macros to those above it. We explain the function and privileges of each layer briefly below.

Syntax tools: This is the lowest layer of code and is designed to provide syntax for control flow and functions to enable simpler macro design and processor-state management at higher levels. It is the only layer allowed to access the stack or

**Fig. 2.** Structure of our code

Our Code	Comment or Equivalent Java/C Style Syntax
declare_var_empty_array buff, 256	@declare buff, 256 byte array or $256/4=64$ int array
declare_var_empty_array sum, 4	@declare sum, 4 byte array or 1 int array
...	@... (any code, changes buff and sum)
mov r2, #0	@r2=0
for checksum_buff,	r0, #V_intlen_buff @for(r0=0; r0<buff.length; r0++) {
read_int_array	buff, r0, r1 @ r1=buf[r0]
add	r2, r2, r1 @ r2=r2+r1
end_for checksum_buff	@}
write_int_array	sum, #0, r2 @sum[0]=r2

Fig. 3. Example code for computing the 32-bit, 2's complement addition checksum over the data in `buff` and writing the result to `sum`. Uses constructs from the syntax tools and memory access layers. (`r0`, `r1`, and `r2` are ARM registers.)

process status (`cpsr` or `spsr`). Constructs provided by this layer include tools for preserving and changing state during macro instantiations and interrupts and syntax for basic for-loops.

Memory access: The primary function of the memory access layer is to provide user-friendly interfaces for declaring and accessing variables and conducting hardware I/O. It also implements a weak form of type safety, discussed in Section 3.3. No code above the memory access layer is allowed to use memory accessing or I/O instructions. Our code currently supports two basic types of variables, byte arrays and integer (32-bit element) arrays because these structures are flexible and all that we require. One example of code for computing a checksum that can be written above the syntax tools and memory access layers is given in Figure 3.

Drivers: Drivers provide application friendly interfaces to the hardware. Note that they do not actually directly access hardware themselves but rather use the I/O constructs provided by the memory access layer, and layers above the drivers are prohibited from accessing the hardware using anything but the functions provided by the drivers. This layer provides tools for writing to the display, accessing flash memory cards, accessing smart cards, playing audio clips (using the timer and D/A converter), determining button state, and reading from the A/D converter. One of our primary techniques for simplifying drivers is to carefully limit error checking (discussed in the full version of the paper [7]).

Application: The highest layer implements the actual functionality of the vote casting device. A significant portion of the code at this layer implements cryptographic operations.

3.3 Type Safety

One of the features we implement into our code is a weak form of type safety. The term “type safety” generally refers to the restriction that “the only operations that can be performed on data in a language are those sanctioned by the type of data” [12]. It helps reduce the risk of many common exploits such as stack and heap overflows by preventing data intended to be written to one variable from being written elsewhere. We do not have true type-safe code but rather achieve a *weak form of type safety* because we rely on auditors to enforce the simple restrictions of each code layer as discussed in Section 3.2³. Specifically, type safety has meaning only in the context of a language, and as we refer to it here, we consider it in the “language” allowed for the drivers and application layers of our code. However, if we assume that the simple “language” allowed at that layer is enforced, we implement a form of what is often referred to as “dynamic type safety”, where the code performs run-time checks to ensure that the operations performed on given data are allowed for that data.

Our code has only two essential data types, static variable arrays, and device I/O registers. When an array is statically declared, our code defines metadata that describes its position and size, which is associated with its name. Then, whenever the variable is read from or written to using one of the data access macros, the assembler expands a series of code around the access to verify that the resulting memory address is actually associated with that variable before the access occurs. If, at run-time, the check determines otherwise, the device displays an error and halts.

One unfortunate side-effect of dynamic address checking, however, is that it is slow. This does not matter for the vast majority of our vote casting device’s functions and hence, through nearly all of the code, every time data is accessed, the operation is first verified as described above. However, when it comes to public-key cryptographic operations, speed is important. For this reason, we also implement some specific type-safe arithmetic operations in the memory access layer of code. These include large number addition, subtraction, and shifting. Because everything about these operations and their parameters is known at assemble time, the loops that iterate through the large number arrays are actually unrolled. This way addresses can easily be checked *statically* and execution speed is also maximized.

3.4 Cryptographic Operations

We utilize several cryptographic operations to help our vote casting device preserve voters’ privacy and ensure the authenticity of cast votes. Implementations

³ Recall that enforcing these restrictions only involves verifying that certain instructions are not used at specific layers.

of cryptographic tools are often quite long and complex, however, so we take several approaches to keeping them as simple as possible.

Our primary approach to minimizing the complexity of our code's cryptography is to utilize constructs all based on the same operations, namely basic modular arithmetic and exponentiation. We use Schnorr signatures [14] to authenticate data written to the flash card and ElGamal encryption [6] for preserving the privacy of unique voter authenticators. With an appropriate hash function, both of these fit our criteria nicely and require a relatively minimal number of operations.

A hash function is required to compute Schnorr signatures as is the case for nearly all other signatures. Since typical hash functions are rather complex, we chose to use a discrete logarithm hash, computed as $\text{hash}(x) = g^x \bmod n$ for $n = pq$ with large, unknown primes p, q [9]. The function is very simple, albeit inefficient, and is provably collision-resistant assuming the hardness of factoring [9]. To compress the result of the function, we xor blocks of 160 bits as suggested by Senderek [15].

Furthermore, the ElGamal decryption for a ciphertext (c_1, c_2) with secret key x and modulus m is classically described as $p = \frac{c_2}{c_1^x} \bmod m$. We clarify that to avoid the need to include (and thus audit) code for the extended Euclidean algorithm, we send $z = -x \bmod q$ to the vote casting device on the key smart cards (where q is the order of the relevant group). Then our code computes the ElGamal decryption as $p = c_2 c_1^z$.

Our ballot casting device requires random numbers for several operations. Random number generation is performed by sampling our microcontroller's analog to digital converter (ADC). We leave the ADC disconnected as suggested by Eastlake *et al.* to pick up electrical noise in the air [5] and use the Von Neumann transition mapping technique [18] followed by parity computation [5] to eliminate skew from the samples. This method allows us to generate all the randomness needed for one voter very quickly. We used NIST's test suite for random number generation [11] to verify that the values obtained are statistically indistinguishable from random. We could use Blum Blum Shub [2] as a pseudo-random number generator (PRNG) with minimal additional code. However, since we would still need the ability to generate a seed and the hardware RNG is sufficiently fast, we use it for all of our random number generation.

We avoid authentication operations, such as authenticator or signature verifications, entirely by pushing them to the public, tallying phase of the election as outlined in Section 2. This also increases election transparency.

4 Conclusion

The security-sensitive functions of a voting machine can be simple, and simplicity reduces oversight and error. We have presented a voting system on which one can cast her vote while trusting only 1,034 lines of code. Reducing this trusted code base eases the task of verification and may thereby provide higher voting assurances.

References

1. Blaze, M., Cordero, A., Engle, S., Karlof, C., Sastry, N., Sherr, M., Stegers, T., Yee, K.-P.: Source code review of the Sequoia voting system. Technical report, California Secretary of State (July 2007)
2. Blum, L., Blum, M., Shub, M.: Comparison of two pseudo-random number generators. In: CRYPTO 1982: Advances in Cryptology (1982)
3. Bruck, S., Jefferson, D., Rivest, R.L.: A modular voting architecture (“Frogs”). In: WOTE 2001: Workshop on Trustworthy Elections (2001)
4. Calandrino, J.A., Feldman, A.J., Halderman, J.A., Wagner, D., Yu, H., Zeller, W.P.: Source code review of the Diebold voting system. Technical report, California Secretary of State (July 2007)
5. Eastlake, D.E., Crocker, S.D., Schiller, J.I.: RFC1750 - randomness recommendations for security, <http://www.faqs.org/rfcs/rfc1750.html>
6. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
7. Gardner, R.W., Garera, S., Rubin, A.D.: Designing for audit: A voting machine with a tiny TCB (full version) (2009), http://cs.jhu.edu/~ryan/min_tcb_voting/
8. Hall, J.L.: Transparency and access to source code in electronic voting. In: EVT 2006: USENIX/ACCURATE Electronic Voting Technology Workshop (2006)
9. Gibson, J.J.K.: Discrete logarithm hash function that is collision free and one way. In: IET Computers and Digital Techniques, vol. 138(6) (November 1991)
10. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: IEEE Symposium on Security and Privacy (2004)
11. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications. In: NIST Special Publication 800-22 (2001)
12. Saraswat, V.: Java is not type-safe. Technical report, AT&T Research (August 1997)
13. Sastry, N., Kohno, T., Wagner, D.: Designing voting machines for verification. In: USENIX Security Symposium (2006)
14. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
15. Senderek, R.: A discrete logarithm hash function for RSA signatures, <http://senderek.com/SDLH/discrete-logarithm-hash-for-RSA-signatures.ps>
16. Stein, R.M., Vonnahme, G., Byrne, M., Wallach, D.: Voting technology, election administration, and voter performance. Election Law Journal: Rules, Politics and Policy 7(2) (June 2008)
17. Thompson, K.: Reflections on trusting trust. Communications of the ACM 27(8) (1984)
18. von Neumann, J.: Various techniques used in connection with random digits. National Bureau of Standards Applied Mathematics Series, vol. 12 (1951)
19. Yee, K.-P.: Extending prerendered-interface voting software to support accessibility and other ballot features. In: EVT 2007: USENIX/ACCURATE Electronic Voting Technology Workshop (2007)
20. Yee, K.-P., Wagner, D., Hearst, M., Bellovin, S.M.: Prerendered user interfaces for higher-assurance electronic voting. In: EVT 2006: USENIX/ACCURATE Electronic Voting Technology Workshop (2006)

Attacking of SmartCard-Based Banking Applications with JavaScript-Based Rootkits

(Short Paper)

Daniel Bußmeyer, Felix Gröbert, Jörg Schwenk, and Christoph Wegener

Horst Görtz Institute for IT Security
Chair for Network and Data Security
Ruhr-University Bochum

{daniel.bussmeyer,felix.groebert,joerg.schwenk,christoph.wegener}@rub.de

Abstract. Due to recent attacks on online banking systems and consequent soaring losses through fraud, different methods have been developed to ensure a secure connection between a bank and its customers. One method is the inclusion of smart card readers into these schemes, which come along with different benefits, e.g., convenience and costs, and endangerments, especially on the security side.

We give a review on a security concept and its implementation deployed as an online banking solution, which consists of a USB smart card reader and a customized browser. We propose a thread model and an attack vector exploiting the limited capabilities of the class one smart card reader. Furthermore a proof of concept malware is presented, which utilizes the primary vulnerability, i.e., class one reader, and otherwise supporting vulnerabilities, to show how transactions may be manipulated.

1 Introduction

Recent developments have shown that the endusers' environment must be considered as vulnerable and insecure. In order to provide secure banking, a lot of precautionary measures have been developed. Most aim to establish a secure communication channel, which supplies confidentiality, integrity and authenticity for transactions made between an enduser and the bank. A common method is the deployment of a class one, two, three or four smart card reader and supporting software at the enduser.

Invoking smart card readers from online banking software offers the possibility to encrypt and sign messages independent of the endusers' computer system. Thereby several differences between the smart card reader classes exist, which mainly vary in their equipped hardware:

Class 1. reader without pin pad

Class 2. reader with pin pad

Class 3. reader with pin pad and display

Class 4. reader with class 3 properties, RSA functionality and a bytecode VM

A class one reader thus only guarantees the lowest security level: The cryptographic primitives, e.g., signatures or encryption, are still executed on the host system which might be infected by malware. Besides the possibility to use a physical device for signing, the class one reader does not increase the attack complexity for a malware author, who infected an endusers machine.

Smart card readers of class two and higher, in contrast, offer a pin pad onboard. The pin code for the smart card entered into the reader does not leave the reader and thus is not readable by malware on the host system.

Furthermore class three readers are equipped with a display that is able to show the data to be signed or encrypted. This gives extra security to the signature data, because the user can actually see what he or she is about to sign. This follows the important principle *only sign what you see*.

Preceding security features can only be exceeded by the highest category of smart card reader classes, security class four. These readers also support RSA functionality and come with a virtual machine making it possible to run small software components. It is a clear advantage for the security of software to run in a virtual machine that cannot be accessed by the host system. Intercepting messages or manipulating procedures running in the virtual machine are very hard to conceal and expend a lot of effort to stay undetected.

In this paper we take a closer look on a deployed solution for internet banking on the basis of a USB device containing the banking software and a smart card reader class one. As shown before, it is difficult for malware to intercept and modify messages undetected, if an online banking system includes smart card readers. But if we consider the *man in the box* scenario, in which the host system is infected by a malware, a smart card reader satisfying only class one standards is not sufficient. As already mentioned before, readers of security class one do not offer the possibility of entering the pin on the device itself. The pin is always processed within the host system. Hence malware does not need to launch attacks on the reader's hardware since all important data is located in the machine's memory it is plugged into.

We discuss the prerequisites in Section 2, propose the attack vector and threat model for the scenario in Section 3.1 and present a proof of concept attack on the implementation in Section 3.2. We conclude in Section 6 on the basis of the related work in Section 4 and mitigation potentialities in Section 5.

2 Prerequisites

In this section we describe the audited solution and its intended usage. Furthermore we describe and discuss a typical usage environment for the application.

2.1 Target Solution

The targeted solution consists of a USB smart card reader including a USB flash drive. The flash drive comes with a customized Firefox web browser, which is run once the device has been inserted into a host system. The solution is used by a large German, a Turkish, and a Swiss financial institution for internet banking.

This paper focuses on the German version with a release date in spring 2009. When the USB device is inserted into a Windows-based machine the flash drive and the smart card reader are recognized by the operating system and (if enabled) autorun launches the application from the flash drive.

At first, the launcher application is started, which is responsible for the creation of other processes and temporary execution directories under %TEMP% in which the processes are executed. A dedicated application and a batch job ensures that if the enduser unplugs the device no temporary files, e.g., cookies or caches, of the internet banking session are left and then terminates the main application.

Next, the launcher application starts an update client which communicates with a server to download updates for the flash drive. The update mechanism might be an attack vector itself, for example by redirecting network traffic and emulating the update server to introduce malicious, infected software on the flash drive. We did not further analyze this vector, because the updates were signed and infecting the web browser is a more direct approach when assuming a *man in the box* threat model.

Third, the web browser is launched. The customized Firefox is the main application for the enduser and when it terminates, the launcher application conducts the removal of temporary files and then finally removes the flash drive.

The browser is based on Firefox 1.5.0.9 and is customized to interact with the USB smart card class one reader and includes a Java plugin. It also contains some code integrity checks:

- Every two seconds a thread checks whether the addresses of SSL_*(), PR_*() and other essential functions have changed.
- Every 42 seconds a thread checks if isDebuggerPresent() returns true and if any other Firefox extensions are installed.

Besides the primary, conceptual vulnerability of using a class one reader, we found several secondary vulnerabilities concerning the concrete implementation:

- Timing is done using Sleep() and time(). By hooking these functions an adversary is able to disable the integrity checks of the application. In general, the reverse engineering process is not very complex as no code obfuscation is done. The code even contains debug strings and OutputDebugString() gives numerous error messages. A code protector, e.g., Themida, might increase the effort to analyze the application.
- Due to the class one reader a keylogger is capable of reading the smart card PIN using GetAsyncKeyState(). No countermeasures exist to prohibit this simple attack.
- The used Firefox version (1.5.0.9) and Java version are obsolete and may be attacked from the network using exploits already publicly known. This way the operating system can be compromised.
- The domain restriction for the user may be circumvented via the menu *Extras* → *Themes* → *Download Themes*. Originally the browser is restricted to the SSL server of the bank.

- The SSL implementation is flawed, so the Null-Byte-X.509-Attack [4] is possible.
- The certificate authorities are obsolete, so the MD5-Signing-Attack [5] is possible.
- No Certificate-Revocation-Lists are included.

2.2 Target Environment

A normal usage environment presumes a login, a password, and a PIN for the smart card. As the primary customers are corporate users, a main advantage of using a class one reader is the possibility of mass signing of transactions. Rather than signing each transaction separately by checking the display and entering a pin on a class three reader, it is possible to sign several hundreds of transactions with a small user effort. In our correspondence with one affected bank this feature outweighed the security disadvantages. Nevertheless we think this is a design decision with too much drawbacks on the security properties of the solution, which we explain in detail in the next section.

3 Proof of Concept Attack

In this section we determine the attack preconditions and present the proof of concept implementation.

3.1 Attack Vector and Threat Model

The adversary's goal is to trigger signed transactions without the user perceiving it. We demarcate the set of manipulations an adversary may introduce to only system-level software, commonly known as the *man in the box* threat model. Software which runs on the external device and on remote (banking) servers, can be considered well secured: the private key is generated with strong parameters and is properly saved on the smart card. The main attack assumption is that the security of the enduser's system is compromised by malware.

Thus, three technical attack vectors to manipulate the user interface arise:

- Operating system level: peripheral manipulation (keyboard, screen, etc.)
- Application level: browser manipulation
- Web application level: client-side JavaScript manipulation

The web application level attack vector poses less effort, but most flexible attack surface. JavaScript may be ported to other operating systems and the attack may be reconstructed for other commercial browser-based banking solutions.

The attack is divided into two stages: at first we modify the running application to include our malicious procedures in any webpage (see Figure 1). Secondly, we conduct the malicious manipulation of transaction using dynamically loaded JavaScript from external sources (see Figure 2).

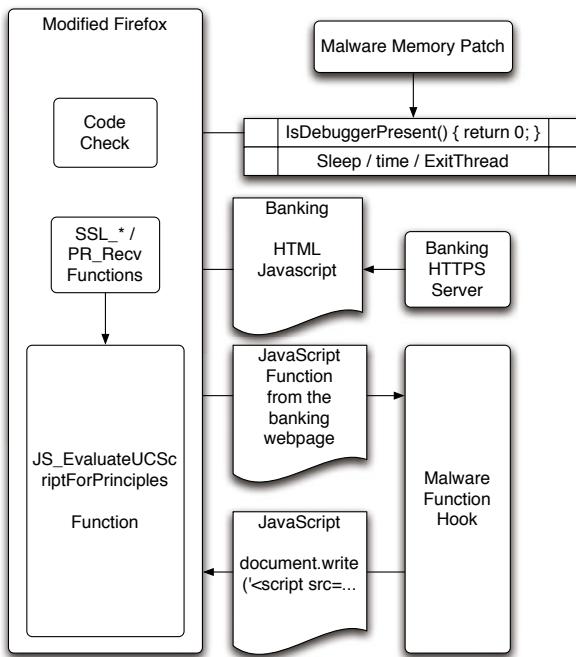


Fig. 1. Overview of stage one of the proof of concept attack

3.2 Implementation

The implementation of stage one is done using the Immunity Debugger¹ and its Python-based scripting interface. Although a real world rootkit would certainly not use a debugger to hook and modify the relevant functions, we choose this technique because we only wanted to demonstrate the general feasibility.

To circumvent the code integrity checks and to manipulate the JavaScript processing functions, we first patch the `IsDebuggerPresent()` function to remain undetected. We therefore let it return zero for any call, as displayed in Figure 1.

```

1 function = imm.getAddress("kernel32.IsDebuggerPresent")
2 imm.writeMemory(function, imm.Assemble("xor eax, eax\n ret"))

```

Listing 1.1. `IsDebuggerPresent` always returns zero.

To introduce the malicious JavaScript code, we hook the Firefox function `JS_EvaluateUCScriptForPrinciples()` from `js3250.dll`, which processes any JavaScript found in the originally legitimate HTML received via SSL from the banking

¹ For details see: <http://www.immunityinc.com/products-immdbg.shtml>

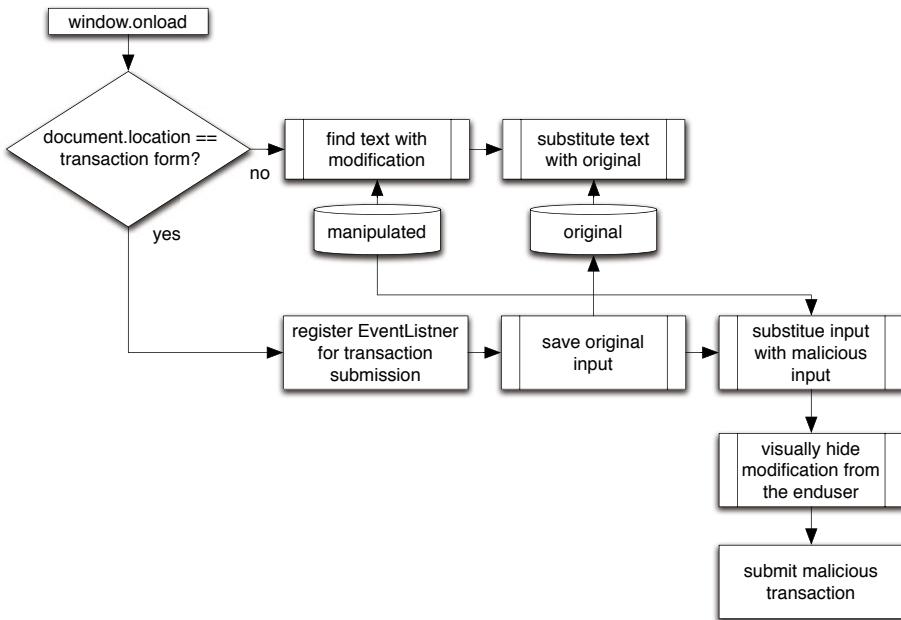


Fig. 2. Overview of stage two of the proof of concept attack

server. We then modify the legitimate JavaScript to include a remote script loaded from the adversary's page.

```

1 document.write(
2   '<script src="http://example.com/malicious.js"></script>')

```

Listing 1.2. Injection of malicious JavaScript.

This way every page viewed in the secure SSL banking context includes malicious and remote JavaScript rootkit which may modify every in- and output on the webpage.

The second stage, displayed in Figure 2, is implemented in the loaded file `malicious.js`. First, when the `window.onload` event fires, the `document.location` is checked whether we are on a banking webpage where the user enters a new transaction destination. If this is the case an event handler is registered for the submit button which in addition to the transfer of specific crucial content of the banking transactions also modifies these values, e.g., the destination account or the amount of money to be transferred. To graphically hide the replacement after the user has clicked the submit button, the text color changes to white. The original input by the user is saved to a cookie for later use. The form is then submitted to the banking server, which expects a transaction to the manipulated account.

If we then land on a webpage which summarizes the transaction, or otherwise displays content, we have to make sure that the manipulated transaction is substituted with the original transaction. We therefore evaluate a regular expression on all text fields, which replaces the manipulated transaction values with the saved original values from the submission above. This way the enduser never sees the manipulated transaction, because the transaction is only shown in the document object model over which the JavaScript rootkit has full control.

The advantages of implementing the main malicious routines using JavaScript is that the adversary may dynamically react on the content of the target victim site. This is enabled by the dynamic loading of the JavaScript each time a website is rendered and eliminates the need for a malware update function.

4 Related Work

Several approaches exist, e.g. [7], to leverage USB devices towards secure transaction signature devices.

Vulnerabilities of transaction signature software has been shown before [3,6,2]. The main attack against class one readers is the keylogger attack, although the modification of signature text has been demonstrated for common banking software.

Several frameworks exist to modify a running application to inject our malicious payload into the webpage. Although we also leverage code modification techniques to inject JavaScript code, the main rootkit functionality is implemented in JavaScript. Besides [1] this is one of the first attacks on internet software using JavaScript as a main rootkit environment.

5 Mitigation

As already mentioned, the usage of a class one smart card device is a conceptual problem. By using this hardware device further security features are useless.

Because the enduser has no extra control over data signed with his or her private key, all malware attacks can take place on his or her computer. Due to a missing display on the USB device or the possibility of entering the PIN code at the smart card reader itself, the enduser has to rely on the data shown on his or her screen. An additional check whether the signed and actually shown data differ is not a valid option.

All security features that smart cards are capable of, do not apply to the concept considered here, because all software is running on the enduser's system. A good approach to provide more security within this concept is the usage of an at least class two smart card reader. When integrating this hardware, the smart card's PIN does not leave the reader. A smart card reader class three, in contrast, is one step ahead by showing the data to be signed or encrypted on an extra display. In this case the user is able to actually see the data and track the corresponding data flow.

If strong security requirements are defined, as for example for online banking software, concepts have to be considered thoroughly. Just adding extra hardware to the solution neither makes it more secure nor bulletproof. Only a well designed scheme has good chances to win the arms-race between upcoming advancing malware attacks and a feasible usability.

6 Conclusion

In this paper we have shown how a class one smart card reader implementation can be attacked under the man in the box model. Although this has been common knowledge for years and attacks on these systems have occurred in the past, there are still commercially available products building their security concept on the basis of a class one reader. To conduct the proof of vulnerability, we used a JavaScript-based attack vector and could successfully manipulate the content of the financial transaction without being visible for the user. In summary it turns out that a class one security model is not sufficient when transaction security is demanded. Whenever transaction authenticity is required, at least a class three reader has to be used which allows a check of the transaction details on a trusted display. But it is also clear that some application scenarios exist where a class one reader is the only choice as a result of the business constraints. In this case a classic threat model analysis has to show which transaction mechanism satisfies the required process security.

References

1. Adida, B., Barth, A., Berkeley, U., Jackson, C.: Rootkits for JavaScript Environments. In: 3rd USENIX Workshop on Offensive Technologies, WOOT 2009 (2009)
2. Drimer, S., Murdoch, S., Anderson, R.: Thinking inside the box: system-level failures of tamper proofing. In: IEEE Symposium on Security and Privacy, SP 2008, pp. 281–295 (2008)
3. Langweg, H., Langweg, H., Snekkenes, E.: A Classification of Malicious Software Attacks. In: Proceedings of 23rd IEEE International Performance, Computing, and Communications Conference (2004)
4. Marlinspike, M.: More Tricks For Defeating SSL In Practice. Blackhat USA (2009)
5. Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D., de Weger, B.: MD5 considered harmful today: Creating a rogue CA certificate. In: 25th Chaos Communications Congress, Berlin, Germany (December 2008)
6. Spalka, A., Cremers, A., Langweg, H.: The Fairy Tale of What You See Is What You Sign. Trojan Horse Attacks on Software for Digital Signatures. In: Proceedings of the IFIP WG, vol. 9 (2001)
7. Weigold, T., Kramp, T., Hermann, R., Horing, F., Buhler, P., Baentsch, M.: The Zurich Trusted Information Channel—An Efficient Defence against Man-in-the-Middle and Malicious Software Attacks. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 75–91. Springer, Heidelberg (2008)

Security Applications of Diodes with Unique Current-Voltage Characteristics

(Short Paper)

Ulrich Rührmair^{1,*}, Christian Jaeger², Christian Hilgers¹, Michael Algasinger¹, György Csaba³, and Martin Stutzmann¹

¹ Computer Science Department

² Walter Schottky Institute

³ Institute for Nanoelectronics

TU München, Germany

ruehrmai@in.tum.de, christian.hilgers@mytum.de, csaba@tum.de,

{christian.jaeger,michael.algasinger,stutz}@wsi.tum.de

<http://www.pcp.in.tum.de>

Abstract. Diodes are among the most simple and inexpensive electric components. In this paper, we investigate how random diodes with irregular $I(U)$ curves can be employed for crypto and security purposes. We show that such diodes can be used to build Strong Physical Unclonable Functions (PUFs), Certificates of Authenticity (COAs), and Physically Obfuscated Keys (POKs), making them a broadly usable security tool. We detail how such diodes can be produced by an efficient and inexpensive method known as ALILE process. Furthermore, we present measurement data from real systems and discuss prototypical implementations. This includes the generation of helper data as well as efficient signature generation by elliptic curves and 2D barcode generation for the application of the diodes as COAs.

Keywords: Physical Cryptography, Physical Unclonable Functions, Certificates of Authenticity, Random Diodes, ALILE Crystallization, SHIC PUFs.

1 Introduction

The use of physical systems with an irregular, at least partly random finestructure recently has gained strong attention in the security and crypto community. In lack of an established, common term, one might call the related field *physical cryptography*, distinguishing it from quantum cryptography or DNA-based approaches. As has been shown in a number of publications starting as early as in the 1980s [1], such disordered physical systems can lead to security applications with enhanced cost efficiency and/or security. Classes of systems that are useful in the area include Strong Physical Unclonable Functions (PUFs) [2] [3] [4], Certificates of Authenticity (COAs) [5] [6], or Physically Obfuscated Keys (POKs) [7] (also called Weak PUFs in [4]).

* Corresponding author.

In this paper, we are concerned with the security applications of diodes with irregular $I(U)$ curves. Such diodes have been prepared in our group by a special, crystallization-based fabrication method known as ALILE process [8] [9]. As we are going to show, they can be employed as building blocks for all three named systems, i.e. both for Strong PUFs, COAs and POKs. Furthermore, they are cheap, take very small chip area, and have a good temperature stability. Therefore, so we argue, they have the potential to become a useful and broadly applicable tool in *physical cryptography*.

The paper is organized as follows. In section 2, we explain the ALILE fabrication process for our diodes. Section 3 describes the use of the diodes as COAs or unforgeable labels, and Section 4 discusses their employment as POKs. In Section 5, we illustrate how our diodes can help us to realize a special type of Strong PUF with high information content, which is naturally immune against machine learning attacks. Section 6 concludes the paper.

2 Sample Preparation

For the preparation of the random diodes we use the aluminum-induced layer exchange (ALILE) process [8] [9], which is known to result in polycrystalline films with p-type conduction [10]. This process is used to crystallize amorphous silicon (a-Si) layers exploiting the catalytic effect of aluminum. Here, an Al/oxide/a-Si layer stack is annealed at temperatures below the eutectic temperature of the Al-Si system. Annealing of the sample leads to diffusion of the Si atoms into the Al layer. Crystallite formation occurs where local supersaturation of the Al with Si is achieved. In addition to that, atomic-scale irregularities and defects, e.g. grain boundaries in the Al, can serve as crystallization sites. Thus, the actual crystallization sites can neither be predicted nor controlled, in particular not by the manufacturer of the structure. The same holds for the irregular crystallite growth.

To illustrate the natural randomness of the process, Fig. 1 a depicts the first step of crystallization recorded by an optical microscope, showing the random distribution of the initial crystallization sites. Fig. 1 b illustrates the random crystallite development in later states of the process. In the ALILE-based fabrication of our random diodes, we chose n-type crystalline silicon wafers as the substrate (see Fig. 1 c) [11].

Medium rectification rates of the diodes are observed for diodes prepared on highly doped wafers (e.g. $\rho = 0.003 - 0.007 \Omega\text{cm}$). Such diodes, which exhibit random

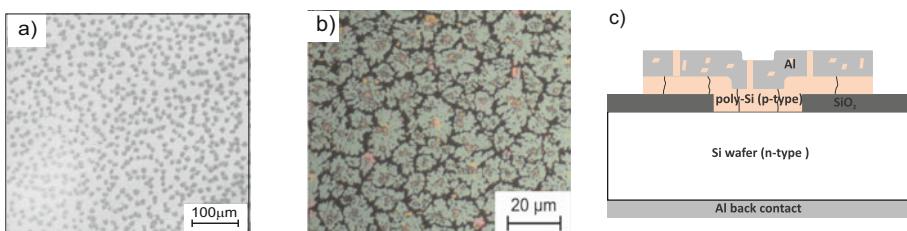


Fig. 1. (a) First crystallites (dark spots) appearing in the Al-matrix during the ALILE process. (b) Irregular growth of the crystallites. (c) Schematic sketch of the diodes' structure.

$I(U)$ characteristics over the whole current-voltage range (see Fig. 2 a), are ideally suited for applications such as electrical COAs (Sec. 3) or POKs (Sec. 4). A very high rectification ratio of the diodes (up to 2×10^7) is obtained for using low doped wafers (e.g. $\rho = 1 - 10 \Omega cm$); see Fig. 3 a. This high rectification allows the application of the diodes in large crossbars structures with high information density, i.e. as Strong PUFs (see Sec. 5). Further details of the fabrication of ALILE layers and the diode fabrication can be found in [10] [11].

3 Electrically Readable Certificates of Authenticity

The use of a disordered physical structure as unforgeable label in connection with an accompanying digital signature has first been proposed in [1], and was termed Certificate of Authenticity (COA) in [5] [6]. COAs require a unique structure that generates a non-imitable *analog* measurement signal, which must be measured by an *external* measurement device. (Note that in opposition to that, most PUFs generate a digital output and have an integrated measurement device.)

Due to the complex and varying $I(U)$ curves, ALILE-diodes can be employed for said task. They can form cheap COAs whose electrical read-out allows very inexpensive readers.

Prototypical Implementation. To test how many different diodes can be distinguished reliably and repeatedly, we collected measurement data of 16 different individual diodes on one chip (Figure 2 a). For 10 out of 16 diodes we repeated every measurement 5 times, and determined the average $I(U)$ curves by taking the arithmetic mean. We also calculated the maximum deviation and the average deviation from the average $I(U)$ curve. In Figure 2 b, the deviation is given in per cent of the respective average value. We observe a decreasing deviations for higher positive voltages, whereas the deviation is slightly lower in the forward direction of the diodes (negative voltages).

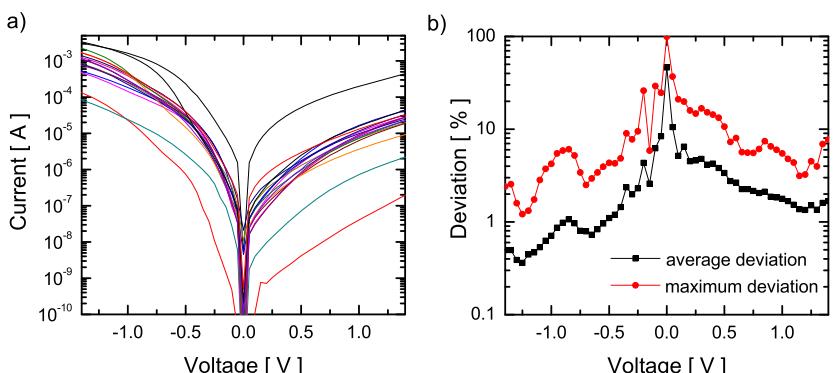


Fig. 2. a) Characteristic $I(U)$ -curves of various diodes. b) Average and maximum deviation of the current values upon multiple measurements.

As straightforward helper data for reliable diode identification, the average curves were tabbed at the fixed voltages -1.3 V, -0.65 V, 0.65 V and 1.3 V. An obvious condition for reliable identification is that the average current values at each supporting point must at least allow a deviation as large as the maximum deviation shown in Figure 2 b. The maximum deviation values for our supporting points are as follows: 1.60 at -1.3 V; 2.95 at -0.65 V; 5.67 at 0.65 V; 3.96 at 1.3 V.

The data gathered by us shows that even at a (hypothetical) variance of up to 27% all the 16 diodes could still be distinguished reliably. At the same time, the diodes only showed deviation values of up to 6% in our experiments. This confirms the possibility for reliable identification.

Along the same lines, we executed a first estimation of the overall number of diodes that can be distinguished with the four supporting points at said voltage levels. We assumed a maximal, practically occurring measurement variance of 10% in our calculation, and obtained roughly 160 distinguishable diodes within the broad band, and around 200 distinguishable diodes in the whole current range. To further increase the complexity of the unique, analog COA-signal, experiments are on the way in our group to investigate the frequency spectra arising from networks of 10 to 100 random diodes. Such periodic networks of non-linear components can exhibit rich, complex spectra [12].

To collect additional support for the applicability of random diodes as unforgeable labels, we carried out a prototypical COA implementation on the basis of 2D barcodes. Parameters of interest here are the resulting barcode sizes and longterm security.

We started by selecting a suitable 2D-barcode, choosing the widely used Data Matrix Code, and implemented it by use of the libdmtx library [13]. Due to the limited storage capacity of barcodes, shorter signatures than RSA are preferred in the generation of COAs; our implementation is based on the bilinear pairing based scheme by Zhang, Safavi-Naini und Susilo (ZSS) [14], which allows signatures of only 160 bits. For the implementation we chose the PBC library [15] with the elliptic curve type F . We assumed that a single waver with 20 diodes is applied as unique object, and that the following information must be stored on the product: Manufacturer ID, product related information (16+48 bit); helper data (20 x 14 bit); digital signature (160 bit). Using a barcode module width 0.25 mm, this leads to a barcode of size 0.81 cm^2 . We successfully generated such a barcode with data from our real measurement data and for exemplary product related data.

Our diode-based approach to COAs therefore leads to inexpensive labels with barcode sizes of less than 1 cm^2 . It allows one of the first electrical COAs with high security and complex analog output; previous COAs were mainly based on optical structures or radiowave scatterers. According to the estimate given in [16], the employed 160-bit elliptic curve signature will be secure until 2019. Signature security until the year 2050 is possible, again on the basis of elliptic curves, with key bitlength around 206 [16] and barcode sizes of still around 1 cm^2 .

4 Physically Obfuscated Keys from Random Diodes

Random physical structures can also be used as a non-volatile storage for secret binary keys. Due to their disordered and/or tamper sensitive nature, they may be harder to

extract invasively than binary keys stored in EEPROM, for example. This concept has been termed a Physically Obfuscated Key (POK) [7], a Weak PUF [4] or also an obfuscating PUF [3]. Applications of POKs naturally include any cryptographic protocols based on secret binary keys, including hardware identification schemes of all sort. They are particularly well suited to store keys safely in small, inexpensive mobile systems, where effective key protection is otherwise difficult to achieve. As we are going to show, random ALILE-diodes can also be used as cheap, stable POKs with remarkably high information density.

Reliable Key Extraction. In the application of ALILE-diodes as POKs, our focus lies on the highly robust extraction of a string (the later key) from the $I(U)$ curves in Fig. 2a). In opposition to COAs, our helper data furthermore should not reveal any information about the binary key which it helps to extract from the POK (see also [17]), since the key must remain secret. We applied ideas taken from Linnartz et. al [18], where the y -axis of the verification measurement is split in equal sections, and the measured data points are shifted towards the arithmetic mean of these sections (i.e. away from the section borders in order to avoid bit flips) by the helper data.

Our data base were the $I(U)$ -curves of the 16 diodes that we already used in section 3. Once more, we set the four supporting points at -1.3 V, -0.65 V, 0.65 V and 1.3 V. Our aim is to extract one bit from the current value at each of the four supporting points, four bits in total per $I(U)$ -curve. Inspired by [18], we proceeded as follows: Firstly, we calculated at each supporting point k ($k = 1, \dots, 4$) the median c_k of the current values of all diodes at this supporting point. Secondly, for each supporting point k , we divided the current-axis into 8 sections. Each section i ($i = 1, \dots, 8$) is determined by its lower border b_i^k and upper border b_{i+1}^k , where $b_i^k = ((p + 1)/(1 - p))^{i-4} \cdot c_k$ for $i = 1, \dots, 8$. In other words, the sections are of equal length on a logarithmic scale, and center around $b_4^k = c_k$. We choose $p = 0.5$ to compensate measurement errors of up to +/-50%. We further denote the arithmetic mean of the section i (with the borders b_i^k and b_{i+1}^k) as $m_{i,i+1}^k$. As is supported by our measurement data, we assume that the measurement points are distributed approximately uniformly over all sections. Under these circumstances, the helper data leaks few/none information about the extracted bit; see also [18].

During the enrollment phase of the POK at the manufacturer, we generate for every measurement s_k at the supporting point k helper data h_k in the following way:

$$h_k = \frac{m_{i,i+1}^k}{s_k} \quad \text{for the unique } i \in \{1, \dots, 8\} \quad \text{that satisfies } b_i^k \leq s_k < b_{i+1}^k \quad (1)$$

During the verification the extracted bit $x(k)$ can be computed with a verification measurement v_k at supporting point k :

$$x(k) = \begin{cases} 0 & \text{if } b_{2i}^k \leq h_k v_k < b_{2i+1}^k \\ 1 & \text{if } b_{2i+1}^k \leq h_k v_k < b_{2i+2}^k \end{cases} \quad (2)$$

With $p = 0.5$ we could obtain 11 different bit strings out of the 16 diodes, while the helper data leaks less information about the bit strings. This means that at least 3 bits

per diode can be extracted in a stable manner and at an error compensation rate of 50% measurement deviation. Our results suggest the usability of one of the simplest and smallest electrical components – namely diodes – as POKs.

5 Machine Learning Resistant Strong PUFs via Crossbar Structures

A Strong PUF is a physical system S which meets the following requirements: (i) S can be excited with external stimuli or challenges C_i , upon which it reacts with corresponding responses R_{C_i} . (ii) It is infeasible, even for the original manufacturer of S , to produce a second system S' which has the same challenge-response-behavior as S . (iii) It is difficult for an adversary to correctly predict an unknown response R_C to a randomly chosen challenge C numerically, without conducting an actual measurement on S . This security feature shall hold even if many other challenge-response pairs (C_i, R_{C_i}) are known to the adversary, or if he had previous physical access to S for a limited period, during which he could conduct any physical measurement on S . In theory, these properties can be met due to the high disorder/information content and/or the complex internal model of S .

Applications of Strong PUFs include identification and key establishment between central authorities and mobile decentral systems [2] [19]. Their complex challenge-response behavior is sufficient to guarantee security in such applications. No execution of costly asymmetric schemes in the mobile systems is necessary.

Current candidates for electrical Strong PUFs contain only a relatively small (max. several hundreds) of *interacting* components. Thus, relatively few (again some hundred) internal parameters completely determine their behavior. This is one of the main reasons why basically all of them have been attacked successfully by machine learning techniques [3] [20]. An alternative design route to Strong PUFs, that has been suggested by our group in [21], is to employ as many (up to billions), densely packed random sub-units as possible, which are read out *individually* and *independently* of each other. Our principle is comparable to a read-only memory with maximal size, random information content, and intrinsically limited read-out rate. We showed in [21] that large, monolithic, memory-like crossbar structures (Fig. 3b) based on random diodes are very well suited to realize this approach. Due to their simple and regular geometry, they can reach optimal information densities (up to 10^{10} to 10^{11} bits per cm^2). The crossbars can be designed in such a way that (i) parallel read-out of different memory units (i.e. diodes) is impossible; (ii) faster read-out than a preset limit leads to overloading and immediate destruction of the wiring, rendering the remaining structure unreadable. Note that the slow read-out rate is not enforced by an artificially slow access module or the like, but by the inductive and resistive capacitances of the structure itself [22].

The resulting Crossbar PUFs are provably immune against machine learning attacks: Their security merely depends on the access time of the adversary, and on the ratio of the already read-out bits vs. the number of overall bits stored in the structure. Modeling attacks subsequent to the read-out are fruitless, since all components are independent of each other. The exact security properties of Crossbar PUFs thereby depend on the employed circuit technology. With a 30nm technology, for example, Crossbar PUFs

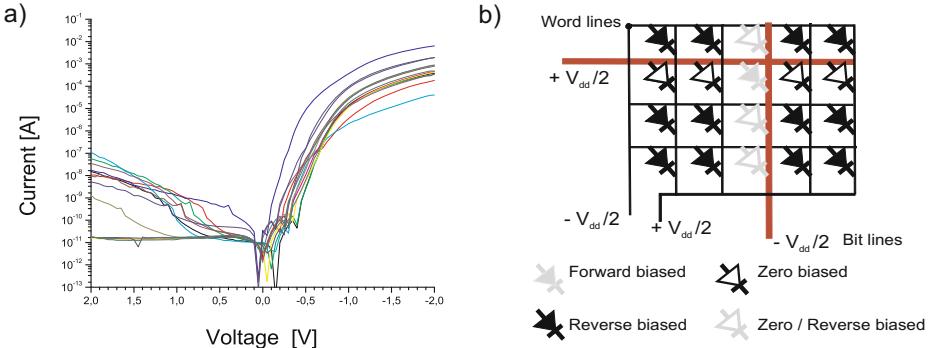


Fig. 3. a) $I(U)$ curves of diodes with high rectification rates; b) schematics of a crossbar structure

of size 1 cm^2 could achieve security of up to 3 years of continuous, uninterrupted adversarial access, while enabling read-out rates of 10^3 bits per second [21]. They could easily be implemented in plug-in devices and on chipcards. Note that all current Arbiter PUFs and variants that run at a 1 MHz CRP frequency become susceptible to modeling attacks after less than a second of uninterrupted adversarial read-out [3] [20].

One prerequisite left open in [21] was whether random diodes with a rectification ratio of at least 10^5 could be produced by inexpensive techniques. Such high rectification rates are necessary to realize stable read-out and to limit parasitic current paths in the monolithic, large crossbar [21] [22]. We have now been able to fabricate diodes with even higher rectification by use of the ALILE process (Fig. 3 a). They indeed enable the first electrical PUFs that remain secure in the face of adversarial access of up to years and against machine learning attacks, further illustrating the security potential of random diodes. We suggest the term SHIC PUFs (pronounce as “*chique PUFs*”) for this new type of PUF, where the acronym SHIC stands for Super High Information Content.

6 Summary

We have argued on the basis of real measurement data and prototypical implementations that random, irregular diodes can be applied for the construction of COAs, POKs and Strong PUFs at the same time. They have the advantage of being one of the smallest and simplest electrical components, and that they can be produced by inexpensive methods. This gives them a strong potential for physical cryptography applications.

Acknowledgements

The presented work was conducted within the Physical Cryptography Project at the TU München. We acknowledge financial support by the International Graduate School of Science and Engineering (IGSSE) and the Institute for Advanced Study (IAS) at the TU München. We thank Michael Scholz and Matthias Bator for useful discussions.

References

1. Bauder, D.W.: An Anti-Counterfeiting Concept for Currency Systems. Research report PTK-11990. Sandia National Labs, Albuquerque, NM (1983)
2. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical One-Way Functions. *Science* 297, 2026–2030 (2002)
3. Rührmair, U., Söltner, J., Sehnke, F.: On the Foundations of Physical Unclonable Functions, <http://eprint.iacr.org>
4. Tuyls, P., Schrijen, G.J., Skoric, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-Proof Hardware from Protective Coatings. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 369–383. Springer, Heidelberg (2006)
5. Vijaywargi, D., Lewis, D., Kirovski, D.: Optical DNA. In: Financial Cryptography 2009, pp. 222–229 (2009)
6. DeJean, G., Kirovski, D.: RF-DNA: Radio-Frequency Certificates of Authenticity. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 346–363. Springer, Heidelberg (2007)
7. Gassend, B.: Physical Random Functions, MSc Thesis, MIT (2003)
8. Nast, O., Wenham, S.R.: Elucidation of the layer exchange mechanism in the formation of polycrystalline silicon by aluminum-induced crystallization. *Journal of Applied Physics* 88, 124–132 (2000)
9. Nast, O., Hartmann, A.J.: Influence of interface and Al structure on layer exchange during aluminum-induced crystallization of amorphous silicon. *Journal of Applied Physics* 88, 716–724 (2000)
10. Antesberger, T., Jaeger, C., Scholz, M., Stutzmann, M.: Structural and electronic properties of ultrathin polycrystalline Si layers on glass prepared by aluminum-induced layer exchange. *Appl. Phys. Lett.* 91, 201909 (2007)
11. Jaeger, C., Algasinger, M., Rührmair, U., Csaba, G., Stutzmann, M.: Random pn-junctions for physical cryptography. *Appl. Phys. Lett.* 96, 172103 (2010)
12. Berkemeier, J., Dirksmeyer, T., Klempt, G., Purwins, H.-G.: Pattern Formation on a Non-linear Periodic Electrical Network. In: Zeitschrift für Physik B Condensed Matter. Springer, Heidelberg (1986)
13. Laughton, M.: (2009), <http://www.libdmtx.org/documentation.php>
14. Zhang, F., Safavi-Naini, R., Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)
15. Lynn, B., et al. (2009), <http://crypto.stanford.edu/pbc/>
16. Lenstra, A.K.: Selecting cryptographic key sizes. *Journal of Cryptology* (2001)
17. Guajardo, J., Kumar, S., Schrijen, G., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
18. Linnartz, J.P., Tuyls, P.: New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688, pp. 393–402. Springer, Heidelberg (2003)
19. Suh, G.E., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: DAC 2007, pp. 9–14 (2007)
20. Rührmair, U., Sehnke, F., Soelter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling Attacks on Physical Unclonable Functions, <http://eprint.iacr.org>
21. Rührmair, U., Jaeger, C., Bator, M., Stutzmann, M., Lugli, P., Csaba, G.: Applications of High-Capacity Crossbar Memories in Cryptography. Accepted at IEEE Transactions on Nanotechnology (to appear)
22. Csaba, G., Lugli, P.: Read-out design rules for molecular cross bar architectures. *IEEE Transactions on Nanotechnology* 8(3), 369–374 (2009)

Verified by Visa and MasterCard SecureCode: Or, How Not to Design Authentication (Short Paper)

Steven J. Murdoch and Ross Anderson

Computer Laboratory, University of Cambridge, UK
<http://www.cl.cam.ac.uk/users/{sjm217,rja14}>

Abstract. Banks worldwide are starting to authenticate online card transactions using the ‘3-D Secure’ protocol, which is branded as Verified by Visa and MasterCard SecureCode. This has been partly driven by the sharp increase in online fraud that followed the deployment of EMV smart cards for cardholder-present payments in Europe and elsewhere. 3-D Secure has so far escaped academic scrutiny; yet it might be a textbook example of how not to design an authentication protocol. It ignores good design principles and has significant vulnerabilities, some of which are already being exploited. Also, it provides a fascinating lesson in security economics. While other single sign-on schemes such as OpenID, InfoCard and Liberty came up with decent technology they got the economics wrong, and their schemes have not been adopted. 3-D Secure has lousy technology, but got the economics right (at least for banks and merchants); it now boasts hundreds of millions of accounts. We suggest a path towards more robust authentication that is technologically sound and where the economics would work for banks, merchants and customers – given a gentle regulatory nudge.

1 Introduction

Card-not-present transactions take place over the Internet, phone, or post, where the merchant and point-of-sale are not in the same physical location as the card and its holder. Fraudulent transactions of this type now account for a large proportion of bank fraud losses. In the UK, for example, it increased 118% from 2003 to 2008, when it accounted for £328.4m of losses to banks and merchants – over half the £610m total for all bank card fraud [3].

This rapid increase has been driven by the deployment of smart cards based on the EMV (Europay, MasterCard, Visa) framework [7] (branded in the English-speaking world as ‘Chip & PIN’). The UK started this in 2003 and completed it around 2006; most of Europe has now finished the rollout and other countries, such as Canada, are starting. Figure 1 shows the effects on the UK fraud figures. Chips reduced fraud via lost and stolen cards, and made card counterfeiting harder for a while (until crooks learned to use the cards overseas), but card-not-present fraud rose dramatically.

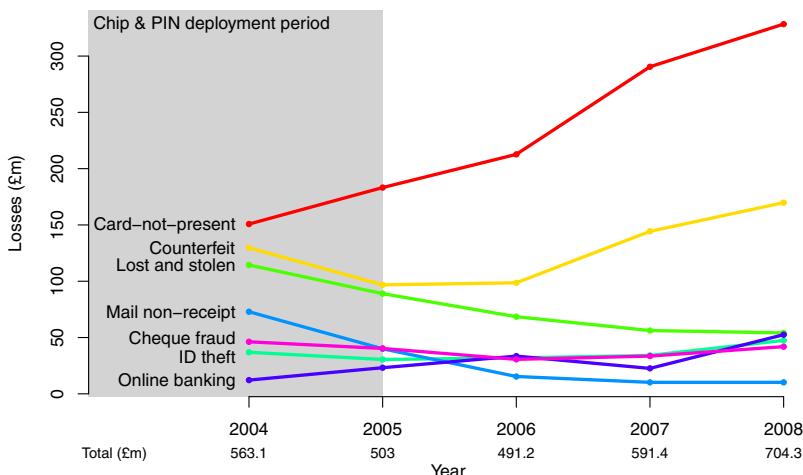


Fig. 1. Fraud totals in the UK [3]

The industry's response to this surge has been 3-D Secure (3DS), known under its brand names 'Verified by Visa' and 'MasterCard SecureCode' [1]. In its initial form, 3DS would pop up a password entry form to a bank customer who attempted an online card payment; she would enter a password and, if it was correct, would be returned to the merchant website to complete the transaction. Difficulties arose with pop-up blockers and now the recommended mode of operation uses inline-frames ('iframe'). The merchant passes the card number to Visa or Mastercard, and gets back a URL to embed in an iframe to display to the customer. If the customer executes the protocol successfully, the merchant gets an authorisation code to submit to his bank.

2 Security Weaknesses

The primary purpose of 3DS is to allow a merchant to establish whether a customer controls a particular card number. It is essentially a single-sign on system, operated by Visa and MasterCard, and it differs in two main ways from existing schemes such as OpenID or InfoCard. First, its use is encouraged by contractual terms on liability: merchants who adopt 3DS have reduced liability for disputed transactions. Previous single sign-on schemes lacked liability agreements, which hampered their take-up. Few organizations are willing to trust a third-party service provider to authenticate users when they have no recourse in the event of error or attack. (In any case, security economics teaches that you're unlikely to get a secure system if Alice guards it while Bob pays the cost of failure.) Second, in other respects 3DS does not adopt the lessons learned from single-sign on, and breaks many established security rules.

2.1 Confusing the User – Hiding Security Cues

The standard advice given to customers to prevent phishing attacks is that they should only enter their bank password in TLS secured sites, and where they have verified the domain name matches what they expect. Browsers have introduced measures to help customers, such as changing the colour of the address bar if TLS is enabled, and making it clearer who the domain name belongs to (e.g. through extended validation certificates). Because the 3DS form is an iframe or pop-up without an address bar, there is no easy way for a customer to verify who is asking for their password. This not only makes attacks against 3DS easier, but undermines other anti-phishing initiatives by contradicting previous advice (as do emails from banks containing clickable URLs). In fact, when one of the authors first encountered 3DS, he established that the iframe came from securesuite.co.uk and called his bank, who informed him that this was a phishing site. Actually this domain name belongs to Cyota (owned by RSA), the company to which many UK banks have outsourced the 3DS authentication process.

2.2 Activation During Shopping

Before 3DS can be used to authenticate transactions, cardholders must register a password with their bank. A reasonably secure method would be to send a password to the customer's registered address, but to save money the typical bank merely solicits a password online the first time the customer shops online with a 3DS enabled card – known as activation during shopping (ADS). To confirm that the customer is the authorized cardholder, the ADS form may ask for some weak authenticators (e.g. date of birth), although not all banks do even this. From the customer's perspective, an online shopping website is asking for personal details. This further undermines customers' security usability and trust experience; and it is being exploited by criminals, as phishing websites impersonating the ADS form to ask for banking details [8] (see Figure 2).

2.3 Informed Consent and Password Choice

By setting up a 3DS password, the customer is deemed to have accepted new terms and conditions. But ADS is not an effective way to obtain informed consent: at the time the terms and conditions are presented, the customer's primary task is to complete the online purchase, so she will not pay much attention to contract terms. Also, because setting a password is a secondary task, they are more likely to choose a poor password, or one they use elsewhere. While Visa requires that customers can opt out at least the first three times, banks may try to force 3DS activation after this stage by preventing the purchase. One of the authors attempted to opt out of using 3DS with a Maestro product; the issuer, the NatWest Bank (now majority-owned by the UK Government), did not allow even one card use without activating 3DS for the account.

Verified by Visa

MasterCard.
SecureCode

Verified by Visa / MasterCard SecureCode Enrollment:
Due to recent changes to FDIC Deposit Insurance Rules all our customers must be enrolled in Verified by Visa or MasterCard SecureCode program depending on type of your Check Card. To continue complete this form and click Activate Now.

Social Security #: - -

Card Number: (16 digits)

Expiration Date: / (MM/YY)

Signature Code: (Last 3 digits on the back)

Card PIN Code: (4-6 digit code that you enter in ATM)

Choose Password: How will it be used?

Confirm Password: (6-12 characters length)

Activate Now

If you already enrolled in Verified by Visa or MasterCard SecureCode program to continue please enter current password or select new then click **Activate Now**.

**VERIFIED
by VISA**

Welcome, 00034-5432-PSI-54256

Verified By Visa

Enter Account Information
Please enter the information below and click the "Continue" button. You can review this information verified by visa account..

Payment Information
Tell us the card to add to your Account

Card Nickname: (example: My Bank One Visa)

Card Number:

Expiration Date:

CVV2:

ATM Pin:

Name on Card (first/last):

Fig. 2. Examples of phishing sites targeting 3DS

2.4 Liability Shifting

As few customers object to terms and conditions, banks are free to set terms that shift liability to customers. For example, the Royal Bank of Scotland says [2]: “You understand that you are financially responsible for all uses of RBS Secure.” So despite the bank having made many poor security choices, the customer must accept the losses – a clear example of misplaced incentives. The use of passwords also harms customer interests because they no longer have the statutory protection afforded by signatures where, in the UK at least, the law makes a forged signature void and thus prevents banks from using their terms and conditions to make customers liable for forged cheques. It has already been documented that many banks used the move away from manuscript signatures to make customers liable for fraud [4].

2.5 Mutual Authentication

3DS may help the customer verify that she’s talking to her actual bank by displaying a memorable phrase she chooses during the ADS process. But first, customers are unlikely to choose a good phrase, given that their goal during ADS is not security but shopping; and second, the memorable phrase is trivially vulnerable to a man-in-the-middle attack.

2.6 Inconsistent Authentication Methods

The 3DS specification only covers the communication between the merchant, issuer, acquirer and payment scheme, not how customer verification is performed. This is left to the issuer, and some have made extremely unwise choices. For instance, one bank asks for the cardholder’s ATM PIN. It’s bad enough that EMV

has trained cardholders to enter ATM PINs at terminals in shops; training them to enter PINs at random e-commerce sites is just grossly negligent. (Phishermen are also asking for ATM PINs on bogus ADS forms.)

Another issuer-specific choice is how to reset the password when a customer forgets it; here again corners are cut. Some banks respond to one or two failed password attempts by prompting an online password reset using essentially the same mechanisms as ADS. In a number of cases the bank requires only the cardholder's date of birth, which is easily available from public records; with one (UK-government-owned) bank, two wrong password attempts simply lead to an invitation to set a new password.

A third variable factor is whether the 3DS implementation asks for a whole password or for some subset of its letters. The idea behind asking for a subset is that a single-round keyboard logging attack does not compromise the whole password. However this compels users to select relatively simple passwords, and probably to write them down. (Thereby they will be in breach of the bank's terms and conditions, and can be refused a refund in case of fraud; so asking for a subset may actually be a rational design choice for the bank.)

2.7 Privacy

An early single sign-on system (Microsoft Passport) was criticised on privacy grounds; modern technologies such as Credentica's U-Prove (being built into the Microsoft InfoCard framework) prevent customers being profiled, even by their authentication provider. 3DS, by contrast, requires that for the cardholder to be shown a description of the transaction, this must be sent to the issuer.

With interbank payment systems, the issuer is told which merchant the customer is dealing with, but the online payment protocol SET (Secure Electronic Transactions) at least arranged things so that the merchant and the bank each got only the transaction data they needed; the bank did not get a description of the goods. So 3DS provides less privacy than either the SET proposal or the existing legacy systems. Furthermore, most banks outsource 3DS authentication and their contractors (e.g. Cyota) see detailed information on more transactions than any individual bank. As these contractors are vulnerable to compulsion (e.g. by FBI National Security Letters), the same tension may arise between U.S. 'anti-terror' law and European privacy law as arose with SWIFT.

3 The Way Forward

3-D Secure has received little public scrutiny despite the fact that with 250 million users of Verified by Visa alone, it's probably the largest single sign-on system ever deployed. What's more, Visa is introducing 'original credits', a payment system based on it, that can support person-to-person money transfer [9]; and EU banks are about to start implementing the Single European Payment Area E-mandate, which will work somewhat like 3DS (a customer will fill out a

bank transfer form at a supplier's website, but using her e-banking password). So it's important to understand what's wrong with 3DS, and how to fix it.

This paper has shown that while previous systems, such as InfoCard and OpenID, had good engineering, they had no incentives for adoption. 3DS fixes the economics, at least for merchants and banks: merchants who adopt it get transactions treated as cardholder-present transactions with much less risk of repudiation, while banks get to shift liability in turn to customers. (In fact the '3D' stands for three domains – the bank, the merchant and the payment network; the customer seems not to have been considered at design time.) Visa's marketing emphasises VbV's 'global liability shift' and claims that it 'addresses' 73% of merchant chargebacks [9].

But 3DS ignores the other lessons learnt from earlier systems. The result is that customers receive little benefit in security, while suffering a huge increase in their liability for fraud. They are also trained in unsafe behaviour online. Now our experience in recent years is that when attacks can be profitably industrialised, they will be; the growth of man-in-the-middle attacks and malware will ensure that 3DS is not sustainable in its present form.

What should be done technically? We believe that single sign-on is the wrong model. What's needed is transaction authentication. The system should ask the customer, "You're about to pay \$X to merchant Y. If this is OK, enter the auth code". This could be added to 3DS using SMS messaging, or systems like Cronto [5] or CAP (Chip Authentication Program) [6] as a stopgap. In the long term we need to move to a trustworthy payment device. This is not rocket science; rather than spending \$10 per customer to issue CAP calculators, banks should spend \$20 to issue a similar device but with a USB interface and a trustworthy display.

What must be done to make it happen? As this paper should bring home, incentives are the key. Visa and MasterCard have managed to get 3DS deployed by arranging so that merchants and banks benefit (at least in the short term) while consumers lose out. What's needed now is for regulators to intervene on behalf of the consumer. The EU already has the Electronic Signature Directive, which contemplates shifting the liability for electronic transactions to bank customers if they are equipped with a secure electronic signature creation device. The missing word is 'only'. If the liability shift is permitted only once the technology actually empowers the customer to decide what transactions she will authorise, then the incentives will line up and finally we might start to move toward a sustainable infrastructure for cardholder-not-present payments.

Acknowledgements

We thank the anonymous reviewers for their comments, Saar Drimer for his contributions to discussions, and John Henderson for his description of the SEPA E-mandate system. Steven Murdoch is funded by the Tor Project and employed part-time by Cronto Ltd.

References

1. 3-D Secure system overview,
https://partnernetwork.visa.com/vpn/global/retrieve_document.do?documentRetrievalId=119
2. RBS Secure Terms of Use (December 2009),
https://www.rbssecure.co.uk/rbs/tdsecure/terms_of_use.jsp
3. APACS. 2008 fraud figures announced by APACS (March 2009),
http://www.ukpayments.org.uk/media_centre/press_releases/-/page/685/
4. Bohm, N., Brown, I., Gladman, B.: Electronic commerce: Who carries the risk of fraud? *The Journal of Information, Law and Technology* 2000(3) (2000)
5. Cronto, http://www.cronto.com/download/Cronto_Products_Datasheet.pdf
6. Drimer, S., Murdoch, S.J., Anderson, R.: Optimised to fail: Card readers for online banking. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 184–200. Springer, Heidelberg (2009)
7. EMVCo, LLC. EMV 4.1 (June 2004), <http://www.emvco.com/>
8. Internet Retailer. Verified by Visa security program used as bait in phishing scams (January 6, 2005),
<http://www.internetretailer.com/dailyNews.asp?id=13764>
9. Varco, J.: Verified by Visa update,
http://www.barclaycardbusiness.co.uk/information_zone/customer_forum/pdf/1315_jon_varco_visa.pdf

All You Can Eat

or

Breaking a Real-World Contactless Payment System*

(Short Paper)

Timo Kasper, Michael Silbermann, and Christof Paar

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
`{Timo.Kasper,Michael.Silbermann,Christof.Paar}@rub.de`

Abstract. We investigated a real-world contactless payment application based on MIFARE Classic cards. In order to analyze the security of the payment system, we combined previous cryptanalytical results and implemented an improved card-only attack with customized low-cost tools, that is to our knowledge the most efficient practical attack to date. We found several flaws implying severe security vulnerabilities on the system level that allow for devastating attacks including identity theft and recharging the amount of money on the cards. We practically verify and demonstrate the attacks on the commercial system.

1 Introduction

A growing number of payment systems incorporate contactless technology, as it offers additional benefits in terms of flexibility and convenience over its contact-based counterpart. With the recent trend towards issuing contactless smartcards in large companies, universities and government entities, a number of privacy- and security-related concerns have been raised.

The “ID-Card” analyzed in the following is, according to the manufacturer, used by more than a million people in Germany. The multi-purpose electronic ID and payment card is based on a dual-interface smartcard, i.e., both contact-based and contactless interfaces are provided. Besides the use for payments, e.g., for food, printing services and washing machine, the functionality of the contactless part includes access control to workplace and apartment, and automatic recording of the working hours of employees. The ID-Card can be charged at dedicated charging terminals with a maximum amount of € 150. The wireless technology implies new threats compared with contact-based systems, for instance, a card could be read out from the pocket or wallet without the owner taking note of it. Thus, we examine the security of the ID-Card as an example for a widespread

* The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

payment system and intend to answer the question: How secure are today's electronic payment systems really? We thereby focus on a realization of the ID-Card e-cash system that is operational since fall 2006 at a large enterprise in Germany.

Through interacting with the card we identified it as a MIFARE Classic from NXP, for which a number of attacks have been published recently, as summarized in Sect. 2. We then used our custom-built radio-frequency identification (RFID) hardware described in Sect. 3.1 to implement the most efficient practical attack on MIFARE Classic known to date and to extract the secret keys from the card. During the subsequent analysis of the system we discovered severe flaws that enable a variety of real-world attacks, as practically evaluated in Sect. 3. The dramatic implications of the attacks are finally described in Sect. 4.

Contactless Payment Systems. A typical contactless payment system consists of RFID readers at the points of sales, and contactless smartcards in the field that operate as electronic wallets and supposedly store the current balance safely. A reader generates a strong electro-magnetic (EM) field at a frequency of 13.56 MHz for supplying a card with energy for its operation and starts the communication. Contactless smartcards, as standardized in ISO 14443 [6], get activated from up to 25 cm [8], while the communication can be passively eavesdropped from a distance of several meters [11].

To allow for rapid operation, the cash registers typically work autonomously without an on-line connection to a database in the back-end, and synchronize their data, for instance, blocked cards or the balance on the account of the cash register, only infrequently, e.g., once per day. This scenario also applies to the payment system analyzed in this paper. For some systems, so-called "shadow accounts" exist for each card that contain the amount of money stored at the instant of the last synchronization and could be used for detecting fraud or functional errors in the system.

MIFARE Classic. Basically, the contactless MIFARE Classic cards are memory cards, i.e., the information is stored in an internal non-volatile memory with an integrated digital control unit to handle the communication with a reader. Furthermore, the interchanged bitstreams can be encrypted. The cards generally comply to Parts 1-3 of the ISO/IEC 14443A [6], but are using a higher-level communication protocol that diverges from Part 4 of the standard. This proprietary protocol for authentication and subsequent data encryption promises to prevent replay attacks, cloning, and eavesdropping by means of the proprietary "CRYPTO1" cipher. The stream cipher is based on a 48-bit linear feedback shift register and six non-linear filter functions. In addition to the small key space, its cryptanalysis revealed several security vulnerabilities, hence CRYPTO1 is commonly regarded as cryptographically weak. A more detailed description of the cipher and its operation principle can be found in [5].

The memory of a MIFARE Classic 1K chip is partitioned into 16 sectors, each consisting of four 16-byte blocks. The read-only block 0 of the first sector contains the factory-programmed Unique Identifier (UID) of the card and the fourth block of each sector contains amongst others two 48-bit keys for the authentication to that sector.

2 Previous Attacks on MIFARE Classic

Since its invention in 1995 by Philips [10], all details of the MIFARE Classic chip had been kept secret until 2007, when the cipher was reverse-engineered [9]. Afterwards, the research on MIFARE Classic revealed numerous security-relevant vulnerabilities, providing the basis for our system break.

Keystream Recovery. The first discovered weakness [9] is that a random nonce n_C generated by the card depends on the time elapsed between the power-up of the card and the issuing of the authentication command by the reader. Hence, the authors are able to reproduce the same nonce with a certain probability by controlling the timing.

A first consequence of this weakness is a keystream recovery attack [3] requiring a recorded authentication session between a genuine reader and a MIFARE Classic card. Afterwards, several queries to the card are issued by a specially prepared reader. Altogether, all 16 bytes of keystream for sector 0 and up to 12 keystream bytes for higher sectors can be recovered. The attack does not enable card cloning, as the cryptographic keys remain secret. However, it is possible to read or modify the whole sector 0 and partially the higher sectors.

Key Recovery from Genuine Authentications The possibility to recover parts of the keystream led to an improved attack [4] that allows to extract a secret key from just two eavesdropped authentications between a card and a genuine reader. No precomputation is required and, after recording the two authentication sessions, it takes only 0.1 s to recover the secret key of one sector, using ordinary computers.

The attack further exploits another weakness of MIFARE Classic cards: Instead of computing parity bits from the actual (encrypted) bits transmitted, the parity is derived from the plaintext. In addition, the same bit of the keystream that is used to encrypt the parity bit of byte N is again used to encrypt the first bit of the next byte $N + 1$ sent.

Card-Only Key Recovery. In a card-only scenario, an attacker only needs to be close enough to the targeted MIFARE Classic card to activate it and communicate with it by means of a special-purpose reader, in order to recover a secret key. Garcia et al. [5] propose four different attacks to obtain the secret key of one sector. Their most time-effective approach requires a precomputed table with a size of 384 GB. On average 4096 authentication attempts are required, thereby keeping the nonce of the reader n_R constant while varying n_C , to obtain one secret key in about two minutes. In addition, if one sector key is known to the attacker, an authentication to that sector can be decrypted to obtain the corresponding n_C . Due to a weakness of the random number generator in the card, the subsequent n_C that is used for authenticating to another sector can be predicted, to recover 32 bit of the keystream generated by the secret key of the new sector. After three authentication attempts 64 bit of keystream are obtained and used in an offline computation step to find the correct key in less than 1 s on

a standard PC. The authors make use of the fact that the card answers with an encrypted NACK = 0x5 command under certain conditions. This known plaintext allows to establish a side-channel to recover four bits of the keystream with a probability of 1/256 per authentication attempt.

The latest and most efficient card-only attack [1] requires only 300 queries to the card on average and a few seconds of off-line computation to find the sector key. It exploits a mathematical vulnerability of the filter functions used in CRYPTO1 to mount a differential attack on the cipher. The off-line computations are negligible, as on average only $2^{48-32} = 2^{16}$ key candidates need to be tested in order to recover a secret key. Note, that the authors of the above summarized papers report difficulties in fixing the card nonce n_C in practice.

3 Tampering with a Real-World System

Analyzing the security of the payment system, we observed that the ID-Card replies with ATQA = 00 04 and SAK = 08, indicating that it contains MIFARE Classic chip [12]. Hence, we performed a practical key-recovery, as detailed in the following.

3.1 Hardware and Software Set-Up

For the security analyses we use a self-built, freely programmable device termed “RFID Tool” [7]. In contrast to commercially available products, our RFID reader allows to fully control the communication and the RF field with a high timing accuracy. This includes arbitrary chosen challenges for the authentication protocol, intentionally wrong calculated checksums and modified parity bits.

The microcontroller of the RFID Tool was programmed to support the full MIFARE Classic authentication protocol described in Sect. 1, and to allow comfortable reading, writing, and cloning of MIFARE Classic cards. The software for the key-recovery is subdivided into two parts: During an on-line step, the embedded part, running on the RFID Tool, emulates a MIFARE Classic reader which collects the data needed for the attack. The data acquisition can be performed in the field by powering the RFID Tool from a battery and storing the acquired information internally. The data is afterwards sent via USB to a desktop PC, where the second part of the software computes the sector keys off-line.

3.2 Recovering the Secret Keys

We implemented our attack based on a combination of the existing attacks [1,5], as described in Sect. 2, and an open-source implementation of the CRYPTO1 cipher [2]. During a normal protocol run, the response times of the MIFARE card vary. Hence, using commercial readers, the same card nonce n_C can only be reproduced with a relatively low probability during an authentication. In order to precisely fix the timing, we use the capability of our reader to wait a fixed multiple of 75 ns between the power-up of the card and the authentication

command. As a result, we can force the card to generate exactly the same n_C in *every* attempt, which highly improves the efficiency of our key-recovery attack. We found that the EM field has to be turned off for approx. 70 ms to ensure a complete reset of the card — smaller time windows did not allow to fix n_C to one value. This defines a new practical lower bound for the time required for an attack and questions some theoretical estimations in the literature.

With our current hardware setup it takes less than 30 s to perform the required authentication runs for revealing one sector key of the card. Once the data is collected, it takes less than 3 s to recover the key on a standard PC. To extract all 16 sector keys from the card we need less than $16 \cdot (30 + 3) = 528$ s \approx 9 min, which is by a factor eight faster than the 80 min for 16 sectors proposed in [1]. No precomputation or other data storage is needed for our attack.

We tested our implementation on an ID-Card and successfully revealed all its secret keys, thereby noticing that the revealed keys differ from the factory-programmed default keys of MIFARE Classic. When analyzing a second card, we found that identical secret keys are used for all sectors. We conjectured that all ID-Cards in the system might have the same keys. By analyzing a dozen more cards we verified the hypothesis: *All* ID-Cards in the payment system have *identical* sector keys.

From comparing the content of several ID-Cards, before and after carrying out charges and payments, we learned which data the smartcard contains apart from the read-only UID and where it is stored in the memory. We revealed that only some bytes of the first three sectors (0, 1, and 2) are affected by monetary transactions, while most parts of the memory remain unused. In more detail, sector 0 contains a card number, which is for some ID-Cards also printed on the card, and a checksum that is calculated as a XOR over all bytes of blocks 1 and 2. The credit value is stored twice in the value blocks 4 and 5 of sector 1 on the card and is not secured by any checksum or other means. Sector 2 contains some information about the last charging and payment process, e.g., a time stamp, a transaction number, and the ID of the last terminal used.

3.3 Practical System-Level Tests

To determine which threats emerge from the vulnerabilities of the system, we carried out a number of tests as detailed in the following. First, we charged an empty ID-Card with, e.g., €5. We copied it to a blank card — which naturally has a different UID — and then pay with this (almost) cloned card: The cloned card behaves exactly as a genuine one, and the money is withdrawn (e.g., €5 \rightarrow €2.70), implying that no checks of the UID are performed.

Three hours later, we paid again with the cloned card to see whether it is now detected or blacklisted. Still, we were able to carry out our payment (e.g., €2.70 \rightarrow €2). Afterwards, we inserted the original ID-Card still charged with, e.g., €5 and the cloned one (e.g., with €2 balance) into the charging machine: The existence of two cards with identical card numbers but different credit amounts is *not* detected, one card shows €5, the other one €2.

Four days later we tried to pay with the original ID-Card charged with €5. The money was withdrawn (e.g., €5 → €2.20), indicating that shadow accounts are not used or at least no action is taken based on inconsistencies.

Finally, we changed the card number, created some fake credit value (e.g., €2.30) and wrote the content to a blank card, thereby taking the check byte at the end of block 2 into account. Even this newly produced card was recognized as genuine and accepted for payments.

3.4 Summary of the Vulnerabilities

We conclude that the security of the analyzed ID-Card payment system is extremely low based on today's cryptanalytical knowledge, since it relies solely on weak MIFARE Classic cards with identical keys. Once the secret keys are extracted from one card, all cards of the system can be read and written to wirelessly from up to 25 cm [8] in less than a second. According to our practical results, the one-time key-extraction is easily performed in minutes. All data on the cards is stored in plaintext and almost no integrity checks are performed. Neither the UID of cloned cards is verified, nor are cards with the same card number but a different balance detected. Seemingly, no additional checks are performed, neither in the front-end, nor in the back-end of the system, allowing for various attacks as illustrated in the following.

4 Resulting Attacks and Their Implications

The implications of the security flaws described in Sect. 3 are dramatic. Once the secret keys are recovered from one card, a variety of devastating attacks can be mounted for any card in the system. The adversary does not need to be an employee to gain access to a card, as an anonymous ID-Card can be obtained at any cashpoint for a deposit of €10. Blank MIFARE Classic cards can be purchased for less than €0.50 on the Internet. All our attacks, including the one-time key extraction, require no special knowledge, but can be performed by any attacker who possesses our public-domain reader and the corresponding software. In court, the actions would be difficult to prove, because no physical traces are left when wirelessly modifying cards.

Impersonation. An adversary needs about 40 ms to covertly extract the card number, and the credit value from the ID-Card of a victim from a distance. Then she has two options: For *digital pickpocketing*, she will lower the money on the card of the victim (requiring another 40 ms) and produce a new card with the same card number and the “stolen” amount of money on it. The adversary can now pay with an ID-Card that is known to the system, using the money of the victim. Hence, the fraud will not be detected in the back-end. This option would harm only the victim, while no damage is caused for the issuing institution.

As a second option, the attacker *impersonates* the victim by generating a new card with the extracted card number and the extracted credit value, to obtain a

duplicate. In addition, she can now increase the amount of money on her card, which will (according to our tests in Sect. 3) still not be noticed by the current realization of the payment system. This time, the card of the victim remains unchanged, and all losses are on the side of the issuing institution.

Trafficking ID-Cards An adversary can produce counterfeit ID-Cards with a new, random card number and a fake credit value of, e.g., €100, and sell them to others. It is also conceivable that a criminal offers a service to charge the cards of other users. Both option imply high losses for the issuing institution.

Denial of Service For a Denial of Service (DoS) attack, we assume an attacker who wirelessly resets credit balances of ID-Cards in the field to a zero value, while leaving the rest of the data unchanged. Performing this attack would harm and confuse all affected legitimate card owners, and result in considerable costs for the customer service and loss of credibility for the contactless payment system.

Distributed All-You-Can-Eat In contrast to a DoS attack, an attacker can wirelessly set credit balances of ID-Cards in the field to a high value, e.g., €100. For both DoS and this rather anarchistic “all-you-can-eat” attack, modifying the two value blocks storing the credit value takes again just about 40 ms. Hence, the attacks can be carried out easily in practice, e.g., by setting up a disguised reader device near a waiting line of the point of sales that modifies the balance of anyone who gets close enough to it accordingly. It is unlikely that a legitimate user of the payment system complains about having too much money on his card, hence the fraud might be detected very late and the financial losses for the institution would be dramatic. In the long term, the attack would render the payment system inoperative.

Emulation of Arbitrary Cards Employing an electronic device that can emulate a MIFARE card, e.g., the “fake tag” developed in [7], an adversary can emulate any ID-Card including its UID, e.g., to behave like an exact clone of a card. Thus, the fake tag can be used (hidden in the wallet) to pretend the presence of a new card with a random UID, a random card number and a random credit value for every payment, thus making detection of fraud and blacklisting impossible. For criminals, selling this device could be very profitable, since it allows for unlimited payments.

5 Conclusion

We summarized the existing attacks on the MIFARE Classic and implemented the most efficient practical card-only attack to date using our custom-built equipment at a cost of below €40. Testing it at hand of the widespread ID-Card payment system we show that recovering the relevant secret keys takes less than 2 min. Once the keys are compromised, the security of the whole system collapses instantaneously, as it turns out that no additional cryptographic mechanisms or other checks are implemented.

Our subsequent analysis of the ID-Card payment application reveals obvious vulnerabilities that pose a great threat to its overall security. An adversary can, in 40 ms and imperceptibly for the victim, read out a card or write to it, increase or decrease its credit balance, impersonate the victim or simply clone his card. Furthermore, a criminal can sell counterfeit cards or electronic devices that emulate a new random card, and hence permit an unlimited amount of payments. Prosecution of the adversary or the victims is not promising. Since the attacks leave no physical traces, it is difficult (or in some cases impossible) to prove that a crime has been committed.

The security flaws do not rely solely on weaknesses of the contactless smart-cards used, but are mainly caused by the realization on the system level. The demonstrated attacks, that can be performed by non-specialists, would become a lot harder or even infeasible in practice, if the system integrator would address the problems detailed in this paper. Using basic cryptographic knowledge, countermeasures could be implemented to obtain a higher security level that would probably be sufficient in the context of micropayments, even on the basis of the weak MIFARE Classic cards.

References

1. Courtois, N.: The Dark Side of Security by Obscurity - and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime. In: SECRYPT, INSTICC, pp. 331–338 (2009)
2. Crypto1. Open Implementation of CRYPTO1 (2008), <http://code.google.com/p/crypto1/>
3. de Koning Gans, G., Hoepman, J.-H., Garcia, F.D.: A Practical Attack on the MIFARE Classic. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 267–282. Springer, Heidelberg (2008)
4. Garcia, F.D., de Koning Gans, G., Muijres, R., van Rossum, P., Verdult, R., Schreur, R.W., Jacobs, B.: Dismantling MIFARE Classic. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 97–114. Springer, Heidelberg (2008)
5. Garcia, F.D., van Rossum, P., Verdult, R., Schreur, R.W.: Wirelessly Pickpocketing a Mifare Classic Card. In: IEEE Symposium on Security and Privacy, pp. 3–15. IEEE, Los Alamitos (2009)
6. ISO/IEC 14443-A. Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part 1-4, (2001), <http://www.iso.ch>
7. Kasper, T., Carluccio, D., Paar, C.: An Embedded System for Practical Security Analysis of Contactless Smartcards. In: Sauveron, D., Markantonakis, K., Bilas, A., Quisquater, J.-J. (eds.) WISTP 2007. LNCS, vol. 4462, pp. 150–160. Springer, Heidelberg (2007)
8. Kirschenbaum, I., Wool, A.: How to Build a Low-Cost, Extended-Range RFID Skimmer. In: USENIX Security Symposium (2006)
9. Nohl, K., Evans, D., Starbug, Plötz, H.: Reverse-Engineering a Cryptographic RFID Tag. In: USENIX Security Symposium, pp. 185–194 (2008)
10. NXP. Mifare Classic (1995), <http://www.nxp.com>
11. NXP. AN200701: ISO/IEC 14443 Eavesdropping and Activation Distance. Technical report (2007)
12. NXP. Mifare Classic 1K MF1 IC S50 Functional Specification (2008), www.nxp.com

Shoulder-Surfing Safe Login in a Partially Observable Attacker Model

(Short Paper)

Toni Perković¹, Mario Čagalj¹, and Nitesh Saxena²

¹ FESB, University of Split

² Polytechnic Institute of New York University

Abstract. Secure login methods based on human cognitive skills can be classified into two categories based on information available to a passive attacker: (i) the attacker *fully observes* the entire input and output of a login procedure, (ii) the attacker only *partially observes* the input and output. Login methods secure in the fully observable model imply very long secrets and/or complex calculations. In this paper, we study three simple PIN-entry methods designed for the partially observable attacker model. A notable feature of the first method is that the user needs to perform a very simple mathematical operation, whereas, in the other two methods, the user performs a simple table lookup. Our usability study shows that all the methods have reasonably low login times and minimal error rates. These results, coupled with low-cost hardware requirements (only earphones), are a significant improvement over existing approaches for this model [9,10]. We also show that side-channel timing attacks present a real threat to the security of login schemes based on human cognitive skills.

1 Introduction

Personal Identification Numbers (PINs) are widely used in modern information systems to authenticate users. Unfortunately, classical PIN-entry methods (via keyboards, keypads and alike) are all vulnerable to *observation attacks* [1]. Many proposals aimed at countering the threat require the user to perform some form of cognitive tasks - so called *cognitive authentication schemes*. The problem of designing a usable cognitive PIN-entry method secure against eavesdroppers is truly challenging. Indeed, it was recently shown in [4] that the cognitive scheme proposed in [12] and all its variants are fundamentally vulnerable to attacks based on SAT solvers.

We can roughly divide existing PIN-entry methods in two classes based on information available to a passive adversary: (i) the adversary *fully observes* the entire input and output of a PIN-entry procedure, and (ii) the adversary can only *partially observe* the input and/or output. For example, the PIN-entry method [5] belongs to the first class (fully observable). In this class of methods, all information exchanged between the user and the interrogator is available to the adversary. Unfortunately, this fact significantly increases the amount of cognitive effort for the user; a 15 digit long PIN required 166 seconds on average [5].

On the other hand, PIN-entry methods [9,10] belong to the second class (partially observable). In this method, the user first receives a challenge via a *protected channel*, and enters the response through a public keypad. In this class of methods, a *passive*

adversary eavesdrops on all public communication between the user and the end system. In another solution described in US patent [13], the user receives a challenge in form of a random number from $\{0, 1, \dots, 9\}$, adds modulo 10 each digit of his PIN to the digits of the random challenge, and finally enters back the outcome via a public keypad. We term this scheme the Mod10 method.

In this paper, we design a novel *Simple Table Lookup (STL)* login method aimed at improving the Mod10 method [13]. Unlike Mod10, our method does not require users to perform any mathematical or mentally demanding operations. It requires the user to perform nothing more than a simple table lookup. Our usability study shows that both Mod10 and STL login methods are user-friendly and have reasonably low login times. The obtained results reveal that Mod10 has slightly lower login time at the cost of a higher error rate compared to the STL. Interestingly, the major source of errors with the Mod10 method are cases in which the sum of a challenge and the PIN digit exceeds 10, which indicates that non-math oriented people might need additional assistance when using the Mod10 method. Indeed, by extending this method with a simple lookup table (referred to as *Mod10-table*) the usability study reveals that older people prefer to use the Mod10-table method rather than Mod10.

All the methods analyzed in this paper essentially implement the “one-time pad” paradigm. As such they are all perfectly secure against passive observation attacks. However, this is conditional on the fact that proper mechanisms for preventing *side-channel timing attacks* are put in place. Indeed, we show that side-channel timing attacks have to be considered seriously in the context of cognitive authentication schemes.

Other “partially observable” solutions that involve a protected challenge channel, as proposed in [9] and [10], require a fairly sophisticated, non-standard and potentially expensive hardware. In contrast, our methods only require a headset, which are commonly available or can be added with little extra cost (e.g. to ATMs).

2 Shoulder-Surfing Safe Login Based on Table Look-ups

Secure PIN-entry with the STL method. STL implements the challenge-response paradigm and comprises three major components: (i) *a protected channel* ensuring secrecy and integrity of challenge values, (ii) *a simple lookup table* - a table of digits from 1 to 9 organized in such a way that each digit i is an immediate neighbor to the other 8 digits from the set $\{1, \dots, 9\}$ (Figure 1(a)) and (iii) a set of *response buttons* (Figure 1(b)). The STL method works as follows. The computer will display the STL table on its screen as shown in Figure 1(a). Let us assume that a user wants to authenticate to a computer using the following PIN: 46548¹. Let us denote PIN digits as $d_0 = 4$, $d_1 = 6$, $d_2 = 5$, $d_3 = 4$ and $d_4 = 8$. At time instant t_0 , the user receives a random challenge (one digit long) c_0 selected from $\{1, \dots, 9\}$, $c_0 = 9$ in our example. The user will receive the challenge over a protected channel (e.g., over *earphones* plugged into the computer). The user looks in the darker area of the STL table (Figure 1(a)) and locates (visually) the PIN digit, $d_0 = 4$. The user then locates (visually) the challenge $c_0 = 9$ in the immediate (one-hop) neighborhood of previously located digit $d_0 = 4$.

¹ With STL method every PIN digit can take one out of 9 values compared with one out of 10 in Mod10 and classical methods; note that $9^5 > 10^4$.

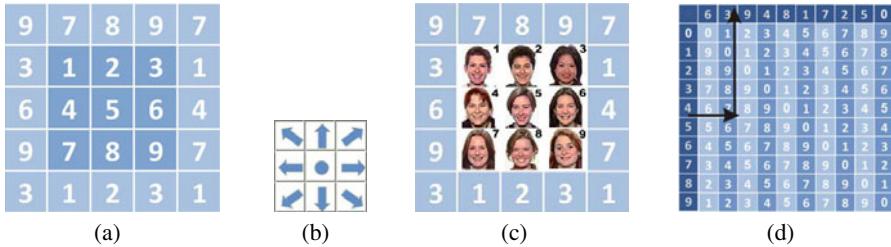


Fig. 1. User interface: (a) In the STL table each digit i is an immediate neighbor to the other 8 digits from the set $\{1, 2, \dots, i-1, i+1, \dots, 9\}$; (b) A user enters his/her response via 8 arrow buttons and one center button; (c) Strengthening Passfaces graphical password system against shoulder-surfing attacks; (d) The Mod10 lookup table

Finally, the user answers the challenge by clicking a response button (Figure 1(b)) that shows the relative position of the challenge c_0 with respect to the corresponding PIN digit d_0 . In our example, the user clicks the “south-west” arrow, that is, he/she responds with $r_0 = \swarrow$. It is easily seen that the response r_0 unambiguously links the challenge with the corresponding PIN digit. This procedure repeats for all the remaining PIN digits. For example, for $d_1 = 6$ the user receives $c_1 = 6$ and responds with $r_1 = \circ$. Note that the STL method does not require any numerical computation on the part of the human user. Moreover, the number of challenge-response rounds equals to the size of the PIN. It is these two features that make the STL method highly usable (Section 3).

Passfaces. While there are many forms of graphical passwords, Passfaces [8] is perhaps the simplest and the most attractive solution in this category. However, Passfaces is particularly vulnerable to shoulder surfing attacks [11]. In Figure 1(c) we show that STL naturally complements Passfaces. Using STL, the threat of observation attacks against Passfaces can be mitigated.

Secure PIN-entry with the Mod10-table method. In order to assist non-mathematically oriented people, we propose to extend the Mod10 method with a simple lookup table. The Mod10 lookup table is shown in Figure 1(d). Let us assume that a user wants to enter PIN digit 4 and that she received random challenge 7. The user first looks up (visually) the digit 4 in the first column of the lookup table. Note that number 4 marks the beginning of the sixth row. Then the user looks for challenge 7 in the sixth row and moves up along the corresponding column (the column nine). The top number in this column, number 3, corresponds to the public response she has to enter back into the system². Note that Mod10-table (Figure 1(d)) does not involve mathematical operations.

3 Usability Evaluation

We carried out experiments in order to study different usability aspects of the Mod10, STL and Mod10-table login methods. The usability test is divided into STL vs Mod10 and a Mod10 vs Mod10-table study. Each study took 90 minutes per user (30 minutes

² Note that if we order the digits in the top row as 0, 1, ..., 9 we get responses that are consistent with the Mod10 additions; hence the name Mod10-table.

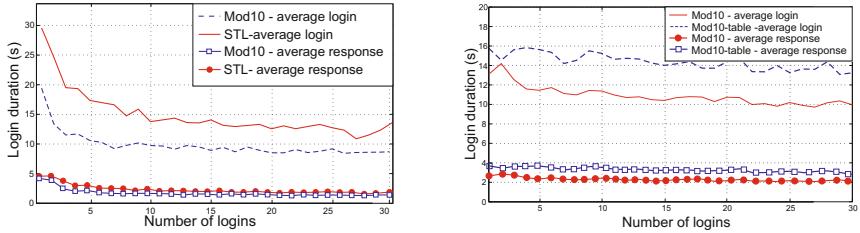


Fig. 2. The average login times and the average user's response time per PIN digit from the experiment with (left) 20 STL and Mod10 users and (right) 38 Mod10 and Mod10-table users

Table 1. Summary of the users' demographics

	Age			Using PC (hours/week)			Using web (hours/week)				
	18-25	26-40	>40	≥ 30	15-30	6-15	≤ 5	≥ 30	15-30	6-15	≤ 5
STL vs Mod10	18	2	0	11	6	2	1	7	7	4	2
Mod10 vs Mod10 table	22	8	8	6	18	4	10	8	5	15	10

per method). The users would take a break, of about half an hour in between the tested methods. In the first usability study, we tested both the STL and the Mod10 methods. In Mod10 vs Mod10-table study we wanted to test whether non-mathematically oriented people find easier to use the lookup table compared to the basic Mod10 method. In both STL vs Mod10 and Mod10 vs Mod10-table study the given tests were randomized. A total of 58 participants took part in the usability study: 20 (13 males, 7 females) in the STL vs Mod10 and 38 (26 males, 12 females) in the Mod10 vs Mod10-table study. Table 1 summarizes user's demographics. The test was voluntary and the users were recruited via flyers. No one of the participants have taken part in any of our tests before.

Implementation and Test Procedure. We implemented the STL, Mod10 and Mod10-table methods as a web application. For each participant, the same test statistics (overall login time, error rates) were collected and stored in a central database. The usability evaluation for each of the PIN entry methods consisted of two phases: A *training phase* and an *authentication phase*. In the training phase participants learned how to use the respective methods (five successful logins per method). The authentication phase served as the actual test authentication methods. The participants were asked to successfully login 30 times per method; there have been no other incentives on the part of the testers (e.g. to achieve faster login times). At the end of each usability test for each login method, the users were asked to complete a post-test questionnaire. The System Usability Scale (SUS) [2] test was used to numerically express the usability of each method. The System Usability Scale (SUS) is a ten-item (Likert) scale giving a global view of subjective assessments of usability [2].

3.1 STL vs. Mod10 Evaluation Study

Login Time. In Figure 2(left), we plot the average login times taken by the 20 participants over 30 successful logins. Already after the first few successful logins, the login time decreases quickly. The overall login times are only 12.5 (std = 7.41) and 9.5 (std = 2.54) seconds on average for STL and Mod10 methods, respectively. Higher login

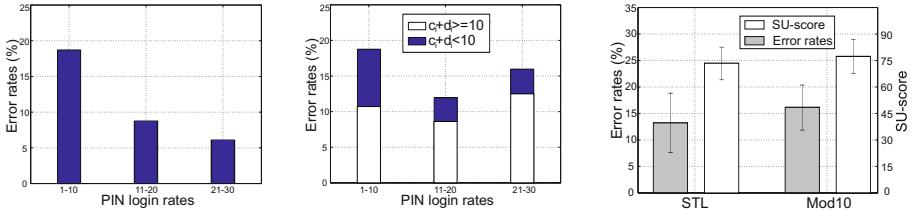


Fig. 3. The average PIN-entry error rates (%) for 20 users using (left) STL and (middle) Mod10 method. (right) The average PIN-entry error rate (%) and SU-score for 20 users.

time with STL can be explained through the size of the PIN (5 digit PIN). The average user response time per PIN digit for both STL and Mod10 are given in Figure 2(left). Towards the end of the testing session, the average user response time for STL and Mod10 is 2.25 (std = 1.17) and 1.8 (std = 0.54) seconds, respectively.

Error Rates. Figure 3 shows average PIN-entry error rates for STL (Figure 3(left)) and Mod10 (Figure 3(middle)) methods over the period of 30 consecutive successful logins. The error rates are shown for 3 equal subsequent periods (10 successful logins per period). In the first period, the error rates are approximately the same for both methods. For the two subsequent periods, Mod10 has a higher error rate than STL. This difference is better seen in Figure 3(right). It is very interesting to observe from Figure 3(middle) that the major source of errors with the Mod10 method are cases in which the sum of the challenge and the respective PIN digit exceeds 10. This type of errors accounts for more than 70% of all errors with Mod10.

Usability score. The SU-scores are shown in Figure 3(right). The average SU-score for STL and Mod10 is 73 and 78 (out of 100). The participants evaluated Mod10 as slightly more usable because of the shorter login time (9.5 vs 12.5 seconds). The majority of participants considered both methods easy-to-use as well as secure (Table 2(left)).

Within User Analysis. Paired t-tests [7] reveal that users achieve significantly higher error rates ($p = 0.0892$) and significantly faster login time ($p = 0.0023$) using the Mod10 method as compared with the STL method. However, users did not consider this method to be significantly more usable than the STL method ($p = 0.1068$).

3.2 Mod10 vs. Mod10-table Evaluation Study

Login Time. In Figure 2(right), we plot average login times for 38 participants over 30 successful logins. Already after the first few successful logins the login time decreases. The overall login times are only 10 (std = 3.92) and 12.5 (std = 3.76) seconds on average for Mod10 and Mod10-table methods, respectively. The average user response time per PIN digit for Mod10 and Mod10-table (Figure 2(right)) is 2 seconds (std = 0.69), and 2.7 seconds (std = 0.72), respectively.

Error Rates. Figure 4 shows average PIN-entry error rates for Mod10 (Figure 4(left)) and Mod10-table(Figure 4(middle)) methods. Similarly to the results of error rates from STL vs Mod10 evaluation, Mod10 method achieves larger error rates due to the fact that Mod10-table requires only a simple table lookup operations.

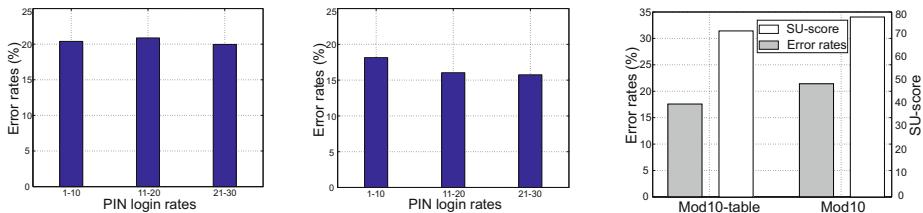


Fig. 4. The average PIN-entry error rates (%) for 38 users using (left) Mod10 and (middle) Mod10-table method. (right) The average PIN-entry error rate (%) and SU-score for 38 users.

Table 2. The table summarizes the responses from users about how acceptable and secure they found using the Mod10 vs Mod10-table and STL vs Mod10 methods (5-strongly agree, 1-strongly disagree), and which method (Mod10 or Mod10-table) they prefer the most

	Using					Feel secure				
	5	4	3	2	1	5	4	3	2	1
STL vs Mod10	10	4	6	0	0	11	5	4	0	0
	9	4	7	0	0	9	9	0	0	0

	18-25	26-40	over 40	Overall
Mod10	19	6	4	29
Mod10-table	3	2	4	9

Usability score. The average SU-score for Mod10-table and Mod10 is 72 and 78 (Figure 4(right)). Thus, in spite of higher error rates, the participants evaluated Mod10 as slightly more usable perhaps because of the shorter login time (10 seconds vs 12.5 seconds). The post-test questionnaire results in Table 2(left) show that majority of participants considered both of the methods easy-to-use and secure. The results in Table 2(right) also indicate that young participants (age 18-25 years) tend to prefer the Mod10 method (87%). However, older participants (over 40 years), are likely to prefer (50%) the login method with a simple lookup table (Mod10-table).

Within User Analysis. Paired t-tests [7] revealed that users achieve significantly higher error rates ($p = 0.0035$), significantly faster login time ($p = 0.000021$) and significantly higher SU-score ($p = 0.01674$) using the Mod10 method as compared with the Mod10-table method. From paired t-tests we conclude that users belonging to the age group 18-25 consider the Mod10 method significantly more usable ($p = 0.0409$) due to the faster login times ($p = 0.001$) achieved with this method despite the higher ($p = 0.001$) error rate. On the other hand, the users belonging to the age group 26-40 and above 40 years do not consider the Mod10 method significantly more usable than the Mod10-table method.

Between User Analysis. For the Mod10 method, unpaired t-tests revealed that users belonging to the age group above 40 take significantly longer time to login compared to the users belonging to the 18-25 group ($p = 0.032$). The means of login times were 9.9756 and 8.9264 seconds, respectively, corresponding to the two age groups. For the Mod10-table method, unpaired t-tests revealed that users belonging to the age group 18-25 years achieve significantly lower error rates to complete the task compared to the users belonging to the group 26-40 years ($p = 0.00117$) and above 40 years ($p = 0.00891$). The means of the error rates are 13.033, 24.663 and 22.994, corresponding to the age groups 18-25, 26-40 and above 40 years.

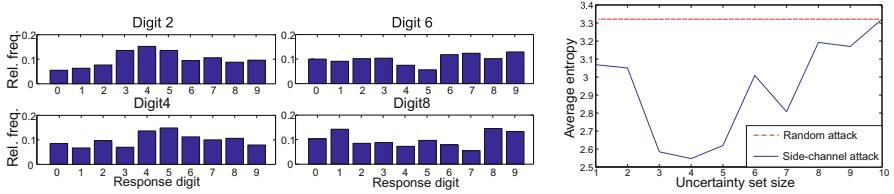


Fig. 5. (left) Relative frequency with which a given response digit appears within $\ell = 4$ fastest response digits, for PIN digits: 2, 4, 6 and 8, (right) Reduction in the average entropy due to the side-channel timing attack

4 Side-Channel Timing Attacks

As we stated before, Mod10, STL and Mod10-table PIN-entry schemes implement the *one-time pad* paradigm. As such they are all perfectly secure against passive observation attacks. However, this is conditionally on the fact that proper mechanisms for preventing side-channel timing attacks are put in place. Side-channel timing attack represents an interesting vector of attacks on cognitive authentication schemes. A classical timing attack is a side channel attack in which an attacker attempts to compromise a given cryptosystem by analyzing the time it takes to execute different cryptographic operations [6]. In this section, we analyze the possibility of reducing the entropy of PINs by simply observing the user's reaction time. We consider a passive attacker capable of recording the user's reaction time during the course of the Mod10 procedure (e.g. by using key-logging malware or a simple camera). The attacker records *the user's response time* (the difference between the moment at which the user receives the challenge value and the moment at which the user enters his/her response. We saw earlier (Figure 3(middle)) that the major source of errors in Mod10 scheme are the cases when the sum of two numbers exceeds 10. Consequently, we hypothesize that these additions (over 10) have longer average response times. To verify this hypothesis, for each user (out of 38) we recorded 30 successful logins and calculated the response time taken for entering a given PIN digit. Since there are only 10 different challenge values and they are generated uniformly at random, each challenge has been generated approximately 3 times on average for the fixed PIN digit. For the fixed PIN digit we count how many users (with this PIN digit) have a given response digit within their ℓ “fastest” response digits³. We plot the corresponding relative frequency in Figure 5(left) for the PIN digits 2, 4, 6, 8 and for $\ell = 4$. As shown, the shortest response time (large bars in Figure 5(left)) for the respective PIN digit occurs in cases in which the users receive challenges from the set {0, 1, or 2}; i.e. for “easy” additions with challenge 0, 1 and 2. Based on these observations, we constructed a set of unique features for each PIN digit. Then, we applied standardized pattern matching techniques (k-nearest neighbor [3]) to a test group of users to classify their PIN digits. The designed algorithm outputs a reduced set of most likely PIN digits. As can be seen from Figure 5(right), with this method we can reduce the entropy from $\log_2 10 \approx 3.3$ bits to 2.55 bits. Similarly, in the STL method the attacker can also observe the correlation between two (or more) equal PIN digits of the respective user and thereby reduce the entropy of the PIN digit.

³ Here we refer to ℓ response digits of the corresponding user, which have shortest response times.

It is important to emphasize that the side-channel timing attack is not specific to Mod10 and STL only. It is common to any cognitive authentication scheme.

5 Conclusion

We made several contributions in this paper. We studied three simple PIN-entry methods – Mod 10, STL and Mod10-table – designed for the partially observable attacker model. All methods are challenge-response protocols that allow a user to login securely in the presence of an adversary who can observe user input (the response values). A notable feature of the Mod10 method is that the user needs to perform a very simple mathematical operation, whereas, in the other two methods, STL and Mod10-table, the user performs a simple table lookup. Our usability evaluation indicates that all methods have reasonably low login times and minimal error rates. Although Mod10 method is slightly faster compared to STL and Mod10-table, it exhibits slightly higher error rates, and was found to be most suitable for younger users. We showed that the threat of side-channel timing attacks has to be considered seriously in the context of cognitive authentication schemes.

Acknowledgment. We would like to thank our shepherd Philippe Golle, and our anonymous reviewers for their thorough reviews and helpful suggestions.

References

1. Backes, M., Drmuth, M., Unruh, D.: Compromising Reflections - or - How to Read LCD Monitors Around the Corner. In: IEEE Symposium on Security and Privacy (May 2008)
2. Brooke, J.: SUS: A Quick and Dirty Usability Scale. In: Usability Evaluation in Industry (1996)
3. Cover, T., Hart, P.: Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory* 13, 21–27 (1967)
4. Golle, P., Wagner, D.: Cryptanalysis of a Cognitive Authentication Scheme (Extended Abstract). In: Proc. IEEE Symposium on Security and Privacy (2007)
5. Hopper, N., Blum, M.: Secure Human Identification Protocols. In: Boyd, C. (ed.) ASI-ACRYPT 2001. LNCS, vol. 2248, p. 52. Springer, Heidelberg (2001)
6. 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)
7. O'Rourke, N., Hatcher, L., Stepanski, E.J.: A Step-by-Step Approach to Using SAS for Univariate and Multivariate Statistics, 2nd edn. SAS Institute Inc. (2005)
8. The Science Behind Passfaces, <http://www.realuser.com/>
9. Kuber, R., Yu, W.: Authentication Using Tactile Feedback. In: Interactive Experiences, HCI, London, UK (2006)
10. Sasamoto, H., Christin, N., Hayashi, E.: Undercover: Authentication Usable in Front of Prying Eyes. In: ACM Conference on Human Factors in Computing Systems (2008)
11. Tari, F., Ant Ozok, A., Holden, S.H.: A Comparison of Perceived and Real Shoulder-surfing Risks Between Alphanumeric and Graphical Passwords. In: SOUPS (2006)
12. Weinshall, D.: Cognitive Authentication Schemes Safe Against Spyware (Short Paper). In: Proc. IEEE Symposium on Security and Privacy (2006)
13. Wilfong, G.T.: Method and Apparatus for Secure PIN Entry. Lucent Technologies, Inc., Murray Hill, NJ, U. S. Patent, Ed. United States (1999)

Using Sphinx to Improve Onion Routing Circuit Construction

(Extended Abstract)*

Aniket Kate and Ian Goldberg

University of Waterloo, ON, Canada
`{akate, iang}@cs.uwaterloo.ca`

Abstract. This paper presents compact message formats for onion routing circuit construction using the Sphinx methodology developed for mixes. We significantly compress the circuit construction messages for three onion routing protocols that have emerged as enhancements to the Tor anonymizing network; namely, Tor with predistributed Diffie-Hellman values, pairing-based onion routing, and certificateless onion routing. Our new circuit constructions are also secure in the universal composability framework, a property that was missing from the original constructions. Further, we compare the performance of our schemes with their older counterparts as well as with each other.

1 Introduction

Goldschlag, Reed and Syverson [2] proposed *onion routing* to achieve low-latency anonymous communication on public networks, which motivated the original Onion Routing project [3] and many other anonymous communication constructions [4,5,6]. Among these, with its hundreds of thousands of users, the second generation onion routing project—Tor [7]—has turned out to be a huge success. However, with its latency times of a few seconds, users find Tor to be very slow for their usual communication over the Internet, and employ it only in situations where their anonymity is indispensable to them. Efficiency is essential for widespread use of anonymity networks; therefore, defining an efficient practical onion routing protocol forms the motivation of this work.

An onion routing (OR) network consists of a set of *onion routers* (OR nodes) that relay traffic, a large set of users and a *directory server* that provides routing information of the OR nodes to the users. A user constructs a *circuit* choosing a small ordered subset of OR nodes, where the chosen nodes route the user’s traffic over the path formed. The key property is that it is difficult for any OR node in a circuit to determine the circuit nodes other than its predecessor and successor. Further, the task must also be difficult for a powerful but not global observer. The user achieves this by sending the first OR node an *onion*—a message wrapped in multiple layers of encryption (one layer per selected node). A user includes the identifier of the next node and a random symmetric session key in each onion layer, and uses nodes’ public keys to encrypt their respective layers. A node

* See [1] for the full version of this paper.

decrypts a received onion using its private key, forwards the remaining onion to the next node, and uses the random symmetric session key for the rest of the session. However, this *single-pass* circuit construction is not *forward secret*; if an adversary corrupts a node and obtains its private key, then the adversary can decrypt all of its past communication. The adversary could then successively compromise all the nodes in a circuit to break the anonymity of a user’s past communications. Although changing the the public/private key pairs for all OR nodes after a predefined interval (*forward secrecy phase*) is a possible solution, it is not scalable. Every system user now has to download a new set of public keys for all the nodes at the start of every forward secrecy phase.

Observing the above issue with forward secrecy, Dingledine, Mathewson and Syverson [5] used an interactive and incremental *telescoping* approach while designing Tor. In the Tor authentication protocol (TAP), which is used to negotiate the session keys in this *multi-pass* circuit construction, a node’s public key is only used to initiate the construction and its compromise does not void the security of the session keys once the randomness used in the protocol is erased. Øverlier and Syverson [8] improved the efficiency of Tor using a half-certified Diffie-Hellman (DH) key agreement [9, §12.6].

However, in Tor, $\Theta(\nu^2)$ messages are required to create a circuit of length ν , as compared to $\Theta(\nu)$ required in a single-pass circuit construction. To solve the scalability issue in single-pass circuit constructions, Kate, Zaverucha and Goldberg [10] suggested the use of an identity-based setting and defined a pairing-based onion routing protocol (PB-OR). Catalano, Fiore and Gennaro [11] suggested the use of a certificateless setting instead and defined two certificateless onion routing protocols (CL-OR and 2-CL-OR). Øverlier and Syverson [8] have also suggested a single-pass circuit construction that provides forward secrecy eventually. However, an extensive comparison between all these schemes is not available yet. In terms of security, none of these practical protocols achieve security in the universal composability (UC) framework [12]. Camenisch and Lysyanskaya [13] presented a framework for UC-secure OR circuit construction, but their protocol is not practical enough for realistic use.

Contributions. In this paper, we present a practical generic onion routing circuit construction protocol that achieves security in the UC model. We apply our protocol to Tor-preDH, PB-OR, CL-OR and 2-CL-OR to define their UC-secure versions. Importantly, the circuit construction messages for these new protocols are significantly smaller than those in the original protocol and there is no addition to a user’s computational cost. We achieve this using *Sphinx*, an efficient message format for mix networks, defined by Danezis and Goldberg [14].

2 Preliminaries

The Sphinx Message Format. Mix message formats have been a point of interest in research on mix networks (see references in [14]). Recently, Danezis and Goldberg [14] proposed Sphinx as the most compact, efficient and UC-secure cryptographic mix message format.

In Sphinx, an adversary is computationally bounded by a security parameter κ . For a prime q of size 2κ bits, let \mathbb{G} be a cyclic group of order q that satisfies the *decisional Diffie-Hellman* (DDH) assumption. Sphinx makes the circuit construction message size independent of the length of the circuit, ν ; we denote the maximum length of a circuit as r . Node identifiers are κ -bit strings. Each node has a public/private key pair. Further, Sphinx assumes a message authentication code (MAC) μ , a pseudo-random generator (PRG) ρ and corresponding random oracle hash functions $h_\mu, h_\rho : \mathbb{G}^* \rightarrow \{0, 1\}^\kappa$, where \mathbb{G}^* is the set of non-identity elements of \mathbb{G} . It also needs a random oracle hash function $h_b : \mathbb{G}^* \times \mathbb{G}^* \rightarrow \mathbb{Z}_q^*$.

Cryptographically, the most elegant feature of the Sphinx message format is its session key derivation technique based on a repeatedly modified random element of \mathbb{G} . We call this technique Sphinx's *blinding logic*. The mentioned random element (α) and its repeated modified forms are called *pseudonyms* since each of these random elements is a temporary public key whose private key is held by the user. In the Sphinx blinding logic, each mix node uses a pseudonym supplied by its predecessor and its own private key to compute the session key with the user. To improve the unlinkability, a pseudonym must not remain the same across the circuit. In the onion routing literature, this is done by including separate random pseudonyms in a construction message for each node in the circuit. In Sphinx's blinding logic, this is achieved using a single repeatedly changing pseudonym. At every node, a blinding factor is extracted from the current pseudonym and the newly computed session key. The pseudonym is then exponentiated with the blinding factor to generate the next pseudonym. In other words, $\alpha_{i+1} = \alpha_i^{h_b(\alpha_i, s_i)}$. The session key is computed by node n_i as $s_i = \alpha_i^{x_i}$, where x_i is the node's private key, and by the user as explained in §3.

To send an anonymous message, a sender first chooses her mix nodes and obtains their public keys. She then computes α_i and s_i and wraps the message in multiple layers of encryption using the PRG ρ to generate ciphertext values β_i . To check the integrity of the message header, she calculates and includes a MAC γ_i at each mixing stage. Upon receiving a message header $(\alpha_i, \beta_i, \gamma_i)$, each mix node n_i extracts session keys using its private key x_i and the pseudonym α_i received from the predecessor. It uses those to verify the MAC γ_i and to decrypt a layer of encryption of β_i . It also extracts the routing information, computes the pseudonym α_{i+1} for the next node and forwards the message to n_{i+1} .

Tor Circuit Construction and Recent Enhancements. In Tor circuit construction [5], a user performs a DH key agreement with each successive node in her circuit over a secure tunnel formed using the already-agreed session keys. This ensures the forward secrecy of the communication immediately after these session keys are deleted. In TAP, a user extends a circuit to node n_i by generating a random $x_i \in_R \mathbb{Z}_q^*$ and sending a DH value (that is, a pseudonym) g^{x_i} encrypted using the (RSA) public key of node n_i . Node n_i decrypts the message and responds by sending g^{y_i} , where $y_i \in_R \mathbb{Z}_q^*$, and a hash of $g^{x_i y_i}$. It is important that the user herself generates and encrypts the DH value g^{x_i} ; if an intermediate adversary OR node (n_j for $0 < j < i$) derives g^{x_i} , it can launch a *man-in-the-middle* attack. In Sphinx's blinding logic, node n_{i-1} uses the received pseudonym

$g^{x_{i-1}}$ to generate and send pseudonym g^{x_i} to node n_i unencrypted. Therefore, it is not possible to directly apply the compact Sphinx message format to Tor.

Overlier and Syverson [8], Kate *et al.* [10], and Catalano *et al.* [11] suggested improvements to Tor circuit construction. These schemes use a *one-way anonymous* key agreement [10] strategy in the public-key cryptography (PKC), identity-based cryptography (IBC) and certificateless cryptography (CLC) settings respectively. Here, a user chooses a random element of \mathbb{Z}_q^* per circuit node and computes an associated pseudonym. A session key is computed using the node’s public key and the random element at the user end, and using the pseudonym received and the node’s private key at the node’s end; the precise session-key computation and the cryptographic assumption vary with the OR circuit construction protocol. Most importantly, unlike Tor, the user does not encrypt the pseudonyms in these schemes, which is a direct result of the inclusion of the private key of an OR node in the session key generation. Therefore, it is possible to incorporate Sphinx’s blinding logic into these schemes.

3 Using Sphinx in OR Circuit Construction

In this section, we first present the generic design of OR circuit construction using the Sphinx methodology. We then implement the generic Sphinx format into three OR circuit constructions. Our design goals and threat model are the same as those of Tor. Refer to the full version of this paper [1] for details.

Generic Design. In Sphinx, a pseudonym α_{i+1} for node n_{i+1} is generated using the pseudonym α_i and the session key s_i generated at node n_i . As discussed above, we can use the Sphinx methodology in an OR circuit construction protocol where a node can create or observe a pseudonym for the next node in the circuit.

To create an OR circuit construction message, we use Sphinx’s mix header creation algorithm ([14, §3.2]) with a generalization of the session key generation. The original Sphinx message format is based on the half-certified DH key agreement [9, §12.6], where a session key s_i is generated as $s_i = y_i^{xb_0b_1\cdots b_{i-1}}$ at the user’s end and as $s_i = \alpha_i^{x_i}$ at node n_i , where (y_i, x_i) is the public/private key pair for n_i , α_i is a pseudonym for node n_i , x is the session-specific randomness and b_0, \dots, b_{i-1} are the blinding factors, $b_i = h_b(\alpha_i, s_i)$. The different OR circuit construction protocols use different session key generation methods, so we generalize this session key generation step. At the user end, we set $s_i = f_U(y_i, xb_0b_1\cdots b_{i-1})$, and at node n_i , $s_i = f_N(x_i, y_i, \alpha_i)$. The other technical details of Sphinx remain exactly the same. Refer to [14, §3.2] for circuit construction message creation and to [14, §3.6] for message processing at a node.

Note that, although Sphinx is defined for single-pass constructions, its blinding logic is also useful in multi-pass constructions, where it can avoid the transfer of pseudonyms in circuit extension messages. However, we concentrate on single-pass constructions as the applicability of Sphinx is more evident there.

Security Analysis. Camenisch and Lysyanskaya [13] design a UC-secure framework for onion routing. They define onion-correctness, onion-integrity and onion-security properties for an OR scheme and prove Theorem 1.

Theorem 1 (Theorem 1 [13]). *An onion routing scheme satisfying onion-correctness, integrity and security, when combined with secure point-to-point secure channels, yields a UC-secure OR scheme.*

Danezis and Goldberg [14] separate a wrap-resistance property from onion-security to simplify the onion-security definition and prove the resulting four security properties of the Sphinx message format using random oracles. We use their security discussion to define the security requirements for our generic OR circuit design. Refer to [1] for details.

We next apply the above generic design to three OR circuit constructions.

Tor with Predistributed DH Values (Tor-preDH). The half-certified DH key agreement scheme [9, §12.6] is a one-pass protocol with unilateral key authentication of the receiver to the sender, assuming that the sender has an authentic copy of the receiver’s public key. Øverlier and Syverson [8] define an enhancement to the Tor circuit construction using this technique.

Let $x_i \in \mathbb{Z}_q^*$ be the private key for node n_i and let $y_i = g^{x_i}$ be its public key, where $g \in \mathbb{G}$ is a chosen generator. In the half-certified DH key agreement scheme, a user generates a random $r_i \in_R \mathbb{Z}_q^*$ and sends a pseudonym $\alpha_i = g^{r_i}$ to node n_i over the already formed circuit (tunnel), if any. The user generates the session key as $s_i = y_i^{r_i}$ and node n_i generates $s_i = \alpha_i^{x_i}$. Øverlier and Syverson used this to present a single-pass protocol (their *second* protocol).

Using the generic Sphinx design, we can not only make their eventual forward secret protocol more efficient but also prove its security in the UC model. Here, except for the entry node, the user is not required to send α_i to node n_i in the circuit. Node n_{i-1} generates the pseudonym $\alpha_i = \alpha_{i-1}^{h_b(\alpha_{i-1}, s_{i-1})}$. All other computation remains the same as the half-certified DH key agreement and the message format remains the same as that of Sphinx.

Pairing-Based Onion Routing. Kate *et al.* [10] observe that the public-key management issue while achieving forward secrecy in single-pass onion routing circuit constructions can be solved using IBC. They develop an anonymous key agreement protocol modifying Sakai-Ohgishi-Kasahara key agreement [15] and use that to define a construction called *pairing-based onion routing* (PB-OR).

We choose three cyclic groups \mathbb{G} , $\hat{\mathbb{G}}$, and \mathbb{G}_T (all of which we shall write multiplicatively) of prime order q and a *bilinear pairing* $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$. We refer the readers to [16] for a detailed discussion of pairings. In the BF-IBE setup, given $(e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T, g \in \mathbb{G}, \hat{g} \in \hat{\mathbb{G}})$, a (possibly distributed) private-key generator (PKG) generates a master key $s \in \mathbb{Z}_q^*$ and an associated public key $y = g^s \in \mathbb{G}^*$, and derives private keys d_i for nodes using their well-known identities and s . A node with identity ID_i receives the private key $d_i = (h_{\text{ID}}(\text{ID}_i))^s \in \hat{\mathbb{G}}^*$, where $h_{\text{ID}} : \{0, 1\}^* \rightarrow \hat{\mathbb{G}}^*$ is a cryptographic hash function.

In PB-OR, a user generates a random $r_i \in_R \mathbb{Z}_q^*$ and sends a pseudonym $\alpha_i = g^{r_i}$ to node n_i over the already-formed circuit (if any). The session key s_i is generated at the user end as $s_i = e(y, h_{\text{ID}}(\text{ID}_i))^{r_i}$ and at the node n_i as $s_i = e(\alpha_i, d_i)$. Using our generic design, α_i can be generated as $\alpha_i = \alpha_{i-1}^{h_b(\alpha_{i-1}, s_{i-1})}$, while the computation of s_i remains the same as that of the original PB-OR, except here $r_i = xb_0b_1 \cdots b_{i-1}$ for an $x \in_R \mathbb{Z}_q^*$ chosen by the user.

Certificateless Onion Routing. Catalano *et al.* [11] recently introduced the concept of certificateless onion routing and presented two protocols (CL-OR and 2-CL-OR) for it. Their motivation is to avoid pairings and to eliminate the interactions between a PKG (or key generation centre—KGC) and nodes in PB-OR using CLC introduced by Al-Riyami and Paterson [17].

In certificateless onion routing, the KGC chooses a random generator $g \in_R \mathbb{G}$, two hash functions $h_{CL} : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ and $h_\pi : \mathbb{G} \times \mathbb{G} \rightarrow \{0, 1\}^\kappa$, and a master key $s \in_R \mathbb{Z}_q$. It then computes $y = g^s$ and publishes $(\mathbb{G}, g, y, h_{CL}, h_\pi)$ as the public key. When a node n_i with identity ID_i asks for its partial private key, the KGC first generates a random $k_i \in_R \mathbb{Z}_q$, computes $\omega_i = g^{k_i}$ and $z_i = k_i + h_{CL}(\text{ID}_i, \omega_i)s$ and returns $d_i = (\omega_i, z_i)$ to node n_i . Each node also generates a random $t_i \in_R \mathbb{Z}_q$ and computes $u_i = g^{t_i}$. The public key for a node n_i with identity ID_i is (ω_i, u_i) and its private key is (z_i, t_i) . In CL-OR, a user generates a random $r_i \in_R \mathbb{Z}_q^*$ and sends the corresponding pseudonym $\alpha_i = g^{r_i}$ to node n_i . The user generates the session key $s_i = (z_{i1}, z_{i2})$ such that $z_{i1} = (\omega_i; y^{h_{CL}(\text{ID}_i, \omega_i)})^{r_i}$ and $z_{i2} = u_i^{r_i}$ and upon receiving pseudonym α_i , node n_i generates $z_{i1} = \alpha_i^{z_i}$ and $z_{i2} = \alpha_i^{t_i}$.

While incorporating the generic Sphinx design, only the computation of the pseudonym α_i changes in the above certificateless key agreement protocol. As above, the pseudonym α_i is generated as $\alpha_i = \alpha_{i-1}^{h_b(\alpha_{i-1}, s_{i-1})}$.

4 Performance Comparison

Message Sizes. Message compactness is an important advantage of using Sphinx. The major savings in the length of a circuit construction message comes from reuse of a pseudonym to which blinding is added at each circuit node.

Following the Sphinx notation, p is the size of a public key element in group \mathbb{G} and r is the maximum length of the circuit. We aim at $\kappa = 128$ -bit security and use the elliptic curve (ECC) setting with points (compressed form) of size $p = 256$ bits, such as provided by Dan Bernstein’s Curve 25519 [18] used by Sphinx. For the finite field setting (\mathbb{F}), as higher values amplify our advantage, we consider a DH modulus of size just $p = 2048$ bits to model 128-bit security. To mitigate a recent attack on Tor by Evans, Dingledine and Grothoff [19], the maximum circuit length for recent versions of Tor is set as 8. Therefore, we set $r = 8$ for our Sphinx-based design. However, while comparing, we give an advantage to the other protocols by using Tor’s default circuit size $\nu = 3$ for them; using $r = 3$ in our design will only increase our advantage. Additionally, see [8] for a discussion of the effect of Tor’s “CREATE_FAST” mechanism.

In the Sphinx-based OR construction, the user sends the tuple $(\alpha_0, \beta_0, \gamma_0)$ to node n_0 . The lengths of the elements in this tuple are p , $(2r - 1)\kappa$ and κ respectively. The total length, therefore, is equal to $p + 2r\kappa$. In the chosen ECC setting, this is equal to 1280 bits, while for the chosen finite field setting, this is equal to 3072 bits. The message size does not depend upon a specific OR design.

In the original Tor-preDH, PB-OR and CL-OR protocols, this cost is equal to $r(p + 2\kappa)$ as each layer of onion in those constructions requires p bits for a pseudonym, κ bits for identity of the nodes and κ bits for message integrity. With $\kappa = 128$ and $\nu = 3$, this length is equal to 1536 bits in the ECC setting

Table 1. Comparison between lengths (in bits) of various single-pass OR circuit construction messages for 128-bit security ($\kappa = 128$)

Scheme	Circuit Size	UC Security Message Size (bits)	\mathbb{F} ($p = 2048$)	ECC ($p = 256$)
Tor-preDH [8]	$\nu = 3$	\times	$\nu(p + 2\kappa)$	6912
PB-OR [10]	$\nu = 3$	\times	$\nu(p + 2\kappa)$	— ^a
CL-OR [11]	$\nu = 3$	\times	$\nu(p + 2\kappa)$	1536
CL05 [13]	$\nu = 3$	✓	$\nu(p + \kappa)$	6528
Sphinx-OR	$r = 8$	✓	$p + 2r\kappa$	3072

^a With the necessity of pairings in the PB-OR protocol, we do not consider the finite field setting for it. ^b As we use an Elgamal ciphertext in ECC, $p' = 2p = 512$.

and 6912 bits in the finite field setting. These values are significantly larger than those in our generic format that can make circuits of any length up to 8.

We also consider Camenisch and Lysyanskaya’s design in [13] that is secure in the UC model. The message length there is $r(p + \kappa)$. In the ECC computation, we use an Elgamal ciphertext of two \mathbb{G} elements of length $p' = 2p = 512$ instead of p . For $\nu = 3$, the message sizes are 1920 bits and 6528 bits respectively. Therefore, our Sphinx-based design achieves the same security guarantees with much smaller messages. Table 1 provides a succinct representation of the above discussion. As Tor generates a circuit in a telescoping form, we do not compare it with the single-pass protocols.

Computational Cost. Compact messages and security in the UC model do not come without some additional computational cost. However, importantly, there is no addition to the computations done by users (possibly hundreds of thousands of them), while the increase is easily manageable for OR nodes. Each node in a circuit has to perform an additional exponentiation in \mathbb{G} as it prepares the pseudonym for the next node. However, one exponentiation in \mathbb{G} costs around 1 ms on a desktop machine. This does not affect the overall circuit construction cost in practice, which is in seconds due to the network latency.

Comparison between the Three OR Constructions. In our full version [1], we also compare the Tor-preDH, PB-OR and CL-OR protocols with Tor as well as with each other in terms of their computational and infrastructural costs. We observed that in multi-pass constructions, Tor-preDH is the most efficient. However, in the absence of a clearly optimal scheme, the choice among the single-pass circuit constructions has to be made based on the size of a prospective anonymity network and availability of a PKG infrastructure. For smaller networks, Tor-preDH and CL-OR are better suited than PB-OR. However, the choice between those two is tricky. In Tor-preDH, the directory server and users have to verify OR nodes’ public key certificates once per forward secrecy phase. In CL-OR, for every circuit construction of length ν a user has to perform ν additional exponentiations and every circuit node has to perform one additional exponentiation. For large anonymity networks, we find PB-OR to be more usable. The public-key downloads saved there are more than compensate for the infrastructure cost incurred by a (distributed) PKG. Further, using the CLC setting, it

may be possible to avoid the public-key scalability and key escrow issues at the same time and it is an interesting future work to design such a scheme.

Acknowledgements. We thank D. Fiore for providing the camera-ready version of his certificateless onion routing paper [11] with D. Catalano and R. Gennaro. We also thank R. Dingledine, G. Zaverucha, and the anonymous reviewers for providing valuable feedback. This work is supported by NSERC, MITACS, and a David R. Cheriton Graduate Scholarship.

References

1. Kate, A., Goldberg, I.: Using Sphinx to Improve Onion Routing Circuit Construction. Technical Report CACR 2009-33 (2009), <http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr2009-33.pdf>
2. Goldschlag, D.M., Reed, M., Syverson, P.: Hiding Routing Information. In: Information Hiding: First International Workshop, pp. 137–150 (1996)
3. Reed, M., Syverson, P., Goldschlag, D.: Anonymous Connections and Onion Routing. IEEE J-SAC 16(4), 482–494 (1998)
4. Dai, W.: PipeNet 1.1 (1998), www.weidai.com/pipenet.txt (accessed November 2009)
5. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: 13th USENIX Security Symposium, pp. 303–320 (2004)
6. Freedman, M.J., Morris, R.: Tarzan: A Peer-to-Peer Anonymizing Network Layer. In: CCS 2002, pp. 193–206. ACM, New York (2002)
7. The Tor Project: (2003), <https://www.torproject.org/> (accessed November 2009)
8. Øverlier, L., Syverson, P.: Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services. In: PETs 2007, pp. 134–152 (2007)
9. Menezes, A., Oorschot, P.V., Vanstone, S.: Handbook of Applied Cryptography, 1st edn. CRC Press, Boca Raton (1997)
10. Kate, A., Zaverucha, G.M., Goldberg, I.: Pairing-Based Onion Routing. In: PETs 2007, pp. 95–112 (2007)
11. Catalano, D., Fiore, D., Gennaro, R.: Certificateless Onion Routing. In: CCS 2009, pp. 151–160 (2009)
12. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS 2001, pp. 136–145 (2001)
13. Camenisch, J., Lysyanskaya, A.: A Formal Treatment of Onion Routing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 169–187. Springer, Heidelberg (2005)
14. Danezis, G., Goldberg, I.: Sphinx: A Compact and Provably Secure Mix Format. In: IEEE Symposium on Security and Privacy, pp. 269–282 (2009)
15. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems Based on Pairing. In: Symposium on Cryptography and Information Security (SCIS 2000), Japan (2000)
16. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for Cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
17. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
18. Bernstein, D.J.: Curve25519: New Diffie-Hellman Speed Records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
19. Evans, N.S., Dingledine, R., Grothoff, C.: A Practical Congestion Attack on Tor Using Long Paths. In: 18th USENIX Security Symposium, pp. 33–50 (2009)

Secure Multiparty AES

(Short Paper)

Ivan Damgård and Marcel Keller

Dept. of Computer Science, Aarhus University, Denmark
`{ivan,mkeller}@cs.au.dk`

Abstract. We propose several variants of a secure multiparty computation protocol for AES encryption. The best variant requires $2200 + \frac{400}{255}$ expected elementary operations in expected $70 + \frac{20}{255}$ rounds to encrypt one 128-bit block with a 128-bit key. We implemented the variants using VIFF, a software framework for implementing secure multiparty computation (MPC). Tests with three players (passive security against at most one corrupted player) in a local network showed that one block can be encrypted in 2 seconds. We also argue that this result could be improved by an optimized implementation.

1 Introduction and Motivation

In secure multiparty computation (MPC), a number of players each supply a private input and then compute an agreed function on these inputs *securely*, i.e., even if an adversary corrupts some of the players, honest players obtain correct results, and the intended outputs is the only new information released about the inputs. Several general feasibility results for MPC are known, for instance, given secure point to point channels, any function can be computed securely against an honest but curious adversary corrupting any minority of the players, and securely against a malicious adversary corrupting strictly less than one third of the players [1, 4].

Although MPC has been a topic in cryptographic research for many years, and despite the obvious potential for applications, implementations have evolved only recently [9, 2, 6]. Some of them have even been used to solve real-world tasks, such as privacy-preserving auctions [3].

In this paper, we present several variants of an MPC protocol for computing AES encryption [11]. We assume that key and plaintext are byte-wise secret shared among the players; the same holds for the outputted ciphertext.

Apart from the general motivation of investigating how far we can take MPC in practice, there is also a more direct motivation for looking at such a “threshold-approach” to symmetric encryption. An example: suppose a set of players hold some secret shared data and wish to communicate this data to an external party. A trivial solution is for each player to send his shares securely to the receiver, who can then reconstruct the data. But this will mean that the receiver must be aware of the fact that the data is secret shared and must apply a non-standard algorithm to get the data. In addition, his work is linear in the number of players. From this point of view it would be a more attractive solution if the players could

cooperate to generate a ciphertext for the receiver in standard form, which would typically be an encryption of an AES key K under the receivers public key, followed by the data encrypted under K . Note that a similar solution used in the opposite direction could be used for a party to supply encrypted inputs to a multiparty computation, even if that party is not aware of the number of players, or the concrete MPC protocol they execute. He only needs to know a public key for the system, where the players share the private key. This could be useful in any application of MPC, e.g. for secure auctions, procurement or benchmarking. In practice, this would mean that parties submitting data to the system can use completely standard client software for sending data securely protected under a public key. Moreover, the back-end of the system can be updated with new MPC protocols or migrate a to a new set of players with no change on the client side, as long as the public key remains the same.

Another application could be the following: Analogously to encrypted hard disks, one could imagine to store data encrypted in a place with weak security compliance (e.g., a cloud), whereas the key is secret shared between different secured machines. Those machines then can run multiparty AES to read and write data together with further MPC to process it. The secret sharing of the key reduces the risk of leakage, as well as the risk of loosing the key. A more naive solution, where one reconstructs the key and encrypts/decrypts data in the normal way, would create a single point of attack from where the entire data-set can be stolen even if one only meant to read a small part.

Whereas choosing a random key K and encrypting it under a public key is easy using known techniques, there is virtually no previous work considering specifically MPC for symmetric encryption (except for an existing 2-player solution, see next section).

Our work on AES exploits the fact that AES is based on arithmetic in $\text{GF}(2^8)$. Therefore, our protocol can be based on any general MPC protocol that is based on Shamir secret sharing [12] and implements secure multiplication and addition in $\text{GF}(2^8)$. With respect to security threshold and type of adversary (passive/active), our protocol will be as secure as the underlying MPC protocol we use. We can, for instance, use the classic passively secure protocol from [1] tolerating a dishonest majority, or the actively secure protocol from [6] tolerating less than one third corrupted players.

The non-trivial problem we need solve is to implement the AES S-box efficiently, since this is the only non-linear part of the algorithm, and essentially requires us to securely compute a multiplicative inverse of an element in $\text{GF}(2^8)$ where 0 should be mapped to 0. The naive solution to this is to raise to the power of 254. We propose several alternative solutions that improve on this by reducing the number of elementary operations, or the number of rounds, or both.

We have implemented our protocol in VIFF, a software framework for implementing secure multiparty computation [6]. Tests for three players running a passively secure protocol on a local network show that an AES block can be encrypted in 2 seconds, and tests also confirm that our methods for reducing the number of rounds lead to better performance when network delays are large

enough to influence speed. Since our implementation uses a general framework based on the high-level interpreted language Python, much better performance can certainly be obtained using a dedicated C implementation. We therefore believe our results demonstrate that MPC for symmetric encryption is definitely a possibility in practice.

2 Related Work

MPC protocols can be divided into two categories. The first consist of protocols computing an arithmetic circuit over a suitable field. These are usually related to a secret-sharing scheme [12]. Other protocols can be used to compute any binary circuit. These are mostly based on Yao’s garbled circuits [13]. An optimized implementation by Pinkas et al. was recently used for secure two-party AES [10]. Their protocol differs from ours, which is of the first category. Their protocol requires one party to know the key, the other to know the cleartext, and outputs the ciphertext to the latter one. Our protocol works for the multiparty case, it takes a secret shared key and cleartext as input and outputs a secret shared ciphertext. The communication complexity of our protocol is smaller, due to the utilization of the arithmetic properties of AES. Our implementation is also faster than that of [10], as detailed later. However, Yao’s garbled circuits lead to constant-round protocols, contrary to ours, in the sense that if one increases the number of AES rounds, our number of rounds increase as well.

Since the original proposal of MPC there have been several improvements to make it more efficient. One of those is pseudorandom secret sharing [5], which allows to generate a secret shared random number without any communication at all. Another improvement is an MPC protocol providing active security which allows preprocessing, i.e., performing some computations without knowing the input to reduce the online time [6]. We will use both techniques in the following.

3 Preliminaries

The Finite Field GF(2⁸). AES treats bytes mostly as elements in GF(2⁸) because there exists a bijective mapping from the set of bytes to the field: $\{0, 1\}^8 \rightarrow \text{GF}(2^8) \cong \text{GF}(2)[x]/(p)$, $a = a_7 \dots a_0 \mapsto \sum_{i=0}^7 a_i \cdot x^i$, where $\text{GF}(2)[x]/(p)$ is the field of polynomials over GF(2) modulo an irreducible polynomial p . Note that GF(2⁸) has characteristic 2, i.e. subtraction is the same as addition.

Secure Multiparty Computation. based on Shamir secret sharing over a field provides the following operations: Addition and multiplication can be done locally, multiplication in general and opening of shared values requires communication. We will refer to the latter two as elementary operations. Throughout the paper, we will use square brackets to denote secret shared values: $[x]$.

Pseudorandom Secret Sharing. allows the distributed generation of random values without communication. For fields with characteristic 2, generation of random bits is also possible. We refer to Section 4.2 of [6] for details.

Bit Decomposition. of an element in $\text{GF}(2^8)$ is required for the S-box of AES. Since $\text{GF}(2^8)$ has characteristic 2, this can be done by masking with a bit-wise random secret shared value. If the random bits are generated using PRSS, the communication cost is one opening. We refer to the full version for details [7].

4 The AES Protocol

AES encryption and decryption are round-based, with each round consisting of some operations on the internal state. This is a matrix of 4×4 bytes, corresponding to a block size of 128 bits. Initial state is the input, final state the output. A typical encryption round looks as follows: SubBytes, ShiftRows, MixColumns, AddRoundKey. The only exceptions are an additional AddRoundKey at the beginning of encryption, and that MixColumns is skipped in the last round.

We now describe how to compute all operations using MPC. Both cleartext and key are assumed to be byte-wise secret shared over $\text{GF}(2^8)$. The internal state and so the output will be as well.

4.1 SubBytes

In SubBytes, an S-box is applied to every byte of the input. Because the S-box is defined arithmetically, we can compute it relatively efficiently with multiparty computation. This is the only part of the protocol requiring communication, everything else can be done locally. The S-box consists of two steps: an inversion on $\text{GF}(2^8)$ and an affine linear transformation on $\text{GF}(2)^8$.

Inversion. The field element represented by the byte is inverted in $\text{GF}(2^8)$, except 0, which is mapped to 0. There are several possibilities of doing this with multiparty computation.

Square-and-multiply. We raise the field element to the power of $\text{ord}(\text{GF}(2^8)) - 1 = 254$ using some square-and-multiply variant. This costs 11 multiplications in 9 rounds per byte, using the addition chain $(1, 2, 4, 8, 9, 18, 19, 36, 55, 72, 127, 254)$. This is optimal regarding the number of multiplications. Since the number of rounds is the lowest possible for the number of multiplications, we will refer to this variant as *square-and-multiply with shortest addition chain and least number of rounds*. Another multiplication chain, $(1, 2, 3, 4, 7, 8, 15, 16, 31, 32, 63, 64, 127, 254)$ requires 13 multiplications in 8 rounds, which is optimal regarding the number of rounds. We will refer to this as *square-and-multiply with least rounds*. Standard square-and-multiply costs 13 multiplications in 13 rounds.

Masked Exponentiation. This method uses the fact that $(x+y)^2 = x^2 + 2xy + y^2 = x^2 + y^2$ for fields with characteristic 2. By a simple induction, it follows that $(x+y)^{2^i} = ((x+y)^{2^{i-1}})^2 = (x^{2^{i-1}} + y^{2^{i-1}})^2 = x^{2^i} + y^{2^i}$ for $i \geq 2$. We exploit this property to split up the computation in a preprocessing and an online phase,

which saves some rounds because the preprocessing operations for all S-boxes of the protocol can be executed in parallel.

In the preprocessing phase, we generate a random shared $[r] \in \text{GF}(2^8)$ and square $[r]$ 7 times. This costs 7 multiplications in 7 rounds if pseudorandom secret sharing is used: $[r^2] = [r] \cdot [r], [r^4] = [r^2] \cdot [r^2], \dots, [r^{128}] = [r^{64}] \cdot [r^{64}]$. To invert $[x]$, we mask $[x]$ by $[r]$, open it, and exponentiate locally:

$$\text{open}([x] + [r]) = (x + r) \quad (x + r)^2, (x + r)^4, (x + r)^8, \dots, (x + r)^{128}.$$

Finally, we unmask the powers of $[x]$ and multiply them to get $[x^{254}]$:

$$(x + r)^{2^i} + [r^{2^i}] = [x^{2^i}] \quad \forall i = 1, \dots, 7, \quad \prod_{i=1}^7 [x^{2^i}] = [x^{\sum_{i=1}^7 2^i}] = [x^{254}].$$

The online operations cost 7 elementary operations (1 opening and 6 multiplications) in 4 rounds.

Masking. Here we exploit that the inversion is a homomorphism with respect to multiplication. The field element $[a] \in \text{GF}(2^8)$ can be masked by multiplying with some shared random number $[r] \in_R \text{GF}(2^8)$. We open the masked value, invert it and unmask the result to get a sharing of the inverted element:

$$\text{open}([a] \cdot [r]) = ar, \quad (ar)^{-1} \cdot [r] = [a^{-1}r^{-1}r] = [a^{-1}].$$

This method would leak whether a is 0 because $ar = 0$ for all $r \in \text{GF}(2^8)$ if $a = 0$. Therefore, if $a = 0$, we add 1 before doing the inversion, and subtract 1 after it. This guarantees that 0 is mapped to 0 without leaking it. Let b be 0 if $a \neq 0$, and 1 if $a = 0$. It can be computed by decomposing a into bits a_i , $a = a_7 \dots a_0$, and then letting $[b] := 1 - \prod_{i=0}^7 (1 - [a_i])$. The inversion is computed as follows:

$$\begin{aligned} \text{open}(([a] + [b]) \cdot [r]) &= (a + b) \cdot r, \quad ((a + b) \cdot r)^{-1} \cdot [r] = [(a + b)^{-1}] \\ [(a + b)^{-1}] - [b] &= \begin{cases} [(a + 0)^{-1} - 0] = [a^{-1}], & a \neq 0, b = 0 \\ [(0 + 1)^{-1} - 1] = [0], & a = 0, b = 1. \end{cases} \end{aligned}$$

If $(a + b)r$ is now 0, we know that $r = 0$. In that case, we just choose another random r and repeat the masking.

The computation of $[b]$ costs 1 opening operation for bit decomposition, and 7 multiplications in 3 rounds afterwards. The computation of $(a + b)r$ costs 1 multiplication and 1 opening, again assuming that the random shared number $[r]$ can be generated locally. Since r might be zero, we require expected $\frac{2}{1-1/256} = 2 + \frac{2}{255}$ elementary operations until $(a+b)r$ is non-zero. The rest can be computed locally, so we get expected $10 + \frac{2}{255}$ elementary operations in $6 + \frac{2}{255}$ expected rounds overall.

Affine Linear Transformation. Here, the byte $a = \sum_{i=0}^7 a_i \cdot x^i$ is considered as a bit vector $(a_0, \dots, a_7) \in \text{GF}(2)^8$, which is multiplied with a fixed invertible matrix and then added to a constant vector. To do so, we decompose the input into bits (cost: 1 opening if PRSS is used), and then we compute the rest locally because the matrix and the vector are fixed.

Communication Cost. The computation of one S-box costs at least 12 elementary operations in 10 rounds or 14 elementary operations in 9 rounds using square-and-multiply and expected $11 + \frac{2}{255}$ elementary operations in $7 + \frac{2}{255}$ rounds using masking. For masked exponentiation, preprocessing requires 7 elementary operations in 7 rounds, and online computation requires 8 elementary operations in 5 rounds, both per S-box.

4.2 Other Operations

ShiftRows, MixColumns, and AddRoundKey consist only of byte permutations and linear operations on $\text{GF}(2^8)$, which can be executed locally. Key expansion uses the same S-box as SubBytes and local operations.

5 Security

The security of our protocol relies mainly on the security of the MPC scheme used. The only information that is revealed additionally to the leakage of the MPC scheme are openings of masked values, i.e. either of $x + r$ or of $y \cdot r$ for a random r and $y \neq 0$. It is easy to see that both openings do not reveal information about x and y , respectively.

It follows that a simulator, e.g., in the UC framework, can generate those values with the same distribution as in the real execution if there exists a simulator for the MPC scheme.

6 Analysis

Since the S-box is the only part which requires communication, it suffices to count the number of S-boxes computed. 16 S-boxes are computed in parallel in every SubBytes operation and thus in every AES round. The key expansion can be computed in parallel with the AES rounds. Putting all together, the total number of elementary operations is the number of S-boxes times the number of elementary operations per S-box, and the the total number of rounds is the number of AES rounds times the number of rounds per S-box.

Table 1 describes the different possibilities for inversion. From the AES specification, one can deduce that the encryption of one block in AES-128 requires 200 S-boxes (including key expansion). So, one can calculate that using inversion by masking, it takes $2200 + \frac{400}{255}$ expected elementary operations in $70 + \frac{20}{255}$ expected rounds. See the full version for the analysis of the other AES flavors [7].

Masked exponentiation only gives an advantage over the other methods if all the preprocessing is done before the encryption of a block. In this way, one can calculate with only $7 / (\text{number of AES rounds})$ rounds per S-box, i.e., 0.7 rounds in the case of AES-128.

Table 1. One S-box in different inversion protocols

	El. operations	Rounds
Masking	$11 + \frac{2}{255}$	$7 + \frac{2}{255}$
Standard square-and-multiply	14	14
S-a-m with shortest add. chain and least rounds	12	10
S-a-m with least rounds	14	9
Masked exponentiation	15	$5 + \frac{7}{\# \text{AES rounds}}$

7 Implementation

Our implementation is based on the Virtual Ideal Functionality Framework (VIFF), a Python-based framework for secure multiparty computation [8]. VIFF was developed to implement efficient MPC for asynchronous networks, i.e., every local computation is executed as soon as the needed input values are present. It provides protocols with passive security as well as protocols secure against active adversaries. Shamir secret sharing is used for protocols with at least three parties, and two-party MPC can be done based on the Paillier cryptosystem.

7.1 Benchmarks

The implementation of the encryption was tested on a local gigabit-network (ping 0.1 ms) with modern hardware: Dual-Core AMD Opteron Processor with 2.4 GHz per core, 2 GB RAM, Red Hat 5.2, Linux Kernel 2.6.18, Python 2.6.1. Using three machines and passive security against one opponent, the encryption of one block with AES-128 took about 2 seconds on average including key expansion when encrypting 10 blocks in parallel. This was achieved using inversion by exponentiation, which turned out to be faster than inversion by masking in the given setting. This is contradictory to our analysis. The reason is that masking needs more local computation (more pseudorandom secret sharing used for bit decomposition), which has a higher impact if the network latency is low and the bandwidth is high. However, the benchmarks behave as expected with a network delay of 40 ms or the bandwidth limited to 800 kbit/s, see the full version [7].

Our method is faster than two-party AES by Pinkas et al. [10] which takes 7 seconds with passive security. Note that their implementation is optimized, whereas ours uses Python, a high-level interpreted language. We observed that local computation is the main bottleneck in our implementation, so this gives the possibility for better results using an implementation in a low-level language with less overhead, such as C.

Moreover, in a setting with four players and active security against one malicious adversary our protocol takes about 7 seconds. This is considerably less than the solution by Pinkas et al. which requires 1148 seconds to encrypt one block with security against a malicious adversary. We used the PRSS-based variant of an actively secure MPC scheme by Damgård et al. [6]. The scheme allows to generate so-called multiplication triples in a preprocessing phase, i.e., before

knowing any input. By using that, the online time can be reduced to less than 4 seconds per block for masked exponentiation, which uses preprocessing also for AES inversion, as described in Section 4.1.

8 Conclusion

We have presented a secure multiparty computation protocol for AES together with benchmarking results of an implementation: roughly 2 seconds per block. Our results can not be applied directly to other algorithms (including ciphers) because we made use of the arithmetic properties of AES, namely of the fact that the S-box is not just a “random” substitution.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10. ACM, New York (1988)
2. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A framework for fast privacy-preserving computations. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 192–206. Springer, Heidelberg (2008)
3. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., 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)
4. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19. ACM, New York (1988)
5. Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 342–362. Springer, Heidelberg (2005)
6. Damgård, I., Geisler, M., Krøigaard, M., Nielsen, J.B.: Asynchronous multiparty computation: Theory and implementation. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 160–179. Springer, Heidelberg (2009)
7. Damgård, I., Keller, M.: Secure multiparty AES (full paper). In: Cryptology ePrint Archive, Report 2009/614 (2009), <http://eprint.iacr.org/>
8. Geisler, M.: VIFF: Virtual ideal functionality framework. Homepage (2007), <http://viff.dk/>
9. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: USENIX Security Symposium, pp. 287–302. USENIX (2004)
10. Pinkas, B., Schneider, T., Smart, N., Williams, S.: Secure two-party computation is practical. In: Cryptology ePrint Archive, Report 2009/314 (2009), <http://eprint.iacr.org/>
11. FIPS Publications. Advanced Encryption Standard. Technical Report FIPS PUB 197, National Institute of Standards and Technology (November 2001)
12. Shamir, A.: How to share a secret. ACM Commun. 22(11), 612–613 (1979)
13. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167. IEEE, Los Alamitos (1986)

Modulo Reduction for Paillier Encryptions and Application to Secure Statistical Analysis

(Extended Abstract)

Jorge Guajardo¹, Bart Mennink^{2,*}, and Berry Schoenmakers^{3,*}

¹ Information and System Security Group
Philips Research, Eindhoven, The Netherlands
jorge.guajardo@philips.com

² Dept. Electrical Engineering, ESAT/COSIC and IBBT
Katholieke Universiteit Leuven, Belgium
bart.mennink@esat.kuleuven.be

³ Dept. of Mathematics and Computer Science
Technische Universiteit Eindhoven, The Netherlands
berry@win.tue.nl

Abstract. For the homomorphic Paillier cryptosystem we construct a protocol for secure modulo reduction, that on input of an encryption $\llbracket x \rrbracket$ with x of bit length ℓ_x and a public ‘modulus’ a of bit length ℓ_a outputs an encryption $\llbracket x \bmod a \rrbracket$. As a result, a protocol for computing an encrypted integer division $\llbracket x \div a \rrbracket$ is obtained. Surprisingly, efficiency of the protocol is independent of ℓ_x : the broadcast complexity of the protocol varies between $O(nk\ell_a)$ and $O(n^2k\ell_a)$, for n parties and security parameter k , and it is very efficient in case of small ℓ_a (in practical cases ℓ_a often is much smaller than ℓ_x). Our protocol allows for efficient multiparty computation of statistics such as the mean, the variance and the median, and it is therefore very applicable to surveys for the benefit of statistical analysis.

1 Introduction

We consider the problem of integer division with remainder in the setting of secure multiparty computation. In its full generality, the problem is to evaluate securely the integer function $(x, y) \mapsto (x \div y, x \bmod y)$, where $x = (x \div y)y + x \bmod y$ and $0 \leq x \bmod y < y$. Whereas integer multiplication commonly allows for secure protocols for which the performance is independent of the bit length of the multiplicands, this is not true for known protocols for integer division. Typically, secure integer division protocols use the binary decomposition of the inputs x and/or y , and consequently these protocols are generally much more elaborate than secure multiplication protocols. To a certain extent, this is to be expected because integer comparison (which generally also requires bitwise-represented inputs) reduces to equality testing given integer division: $x < y$ if and only if $x = x \bmod y$.

* Work done partly while visiting Philips Research Labs.

In this paper we will focus on the computation of $x \bmod a$ for a public modulus a . We observe that in many applications, particularly in secure statistical analysis, division is used with a public modulus only. For example, for the mean $\bar{x} = (x_1 + \dots + x_L) \bmod L$ of a set of L values, it suffices to divide by the publicly known L . A similar observation can be made for the computation of the variance. Hence, efficient protocols for the case of public a are clearly of interest, and we will show how to achieve efficient solutions. Although our results apply to a broad range of approaches in secure multiparty computation, we will present our protocols mostly for the framework based on threshold homomorphic cryptosystems (THCs) [11, 5, 22]. This framework allows n parties, $n \geq 2$, to securely and privately evaluate a given function f : given encrypted inputs $\llbracket x_1 \rrbracket, \dots, \llbracket x_L \rrbracket$, it will be ensured that the output is an encryption $\llbracket f(x_1, \dots, x_L) \rrbracket$, without leaking any further information on the values x_1, \dots, x_L . In general, one may construct a Boolean or arithmetic circuit for f , consisting of basic gates such as NAND gates or addition/multiplication gates, and then evaluate this circuit securely. For performance reasons, however, specific protocols are needed to obtain more practical solutions.

The advantage of our THC-based protocols is that the malicious case can be treated without efficiency loss (asymptotically) compared to the semi-honest case. Our protocols can also be translated to the framework based on verifiable secret sharing (cf. [6]). In this case, however, the malicious case will be (asymptotically) more expensive than the semi-honest case. The technical reason is that some of the particularly efficient zero-knowledge interval proofs used in the THC-based approach do not carry over to the VSS-based approach. Our protocols can be seen as a generalization of the bit decomposition protocols of [23]. As observed in [23], the problem of evaluating $x \mapsto x \bmod 2$ is already non-trivial as it cannot be solved when ElGamal is used as the underlying (additively) homomorphic cryptosystem: an efficient protocol for computing the least significant bit $\llbracket x \bmod 2 \rrbracket$ for given $\llbracket x \rrbracket$ would imply efficient computation of a hard-core bit (of the one-way function $x \mapsto g^x$), contradicting the discrete log assumption. Therefore, we will use a sufficiently strong homomorphic cryptosystem for our protocols, concretely the Paillier cryptosystem. An immediate application of integer division is to securely access arbitrary bits of a given input $\llbracket x \rrbracket$ efficiently. For an ℓ_x -bit integer x , the work to access the i -th least significant bit will be proportional to i , as $x_i = (x \bmod 2^i) \bmod 2$. Our protocols actually simplify considerably for the case that a is a power of 2, such that the overall work is much less than one would need using the bit decomposition protocol of [23].

1.1 Our Contributions

For a set of n participants jointly sharing the decryption key of the Paillier cryptosystem, we construct a protocol which, on input $\llbracket x \rrbracket$ with $x < 2^{\ell_x}$ and a public ‘modulus’ a such that $2^{\ell_a-1} < a \leq 2^{\ell_a}$, outputs an encryption of the modulo reduction of x with respect to a , $\llbracket x \bmod a \rrbracket$. Consequently, this implies a protocol for computing an encrypted integer division $\llbracket x \bmod a \rrbracket$. The efficiency of the protocol relies on the fact that a is known and, in particular, its length

is known. The protocol has a broadcast complexity varying between $O(nk\ell_a)$ and $O(n^2k\ell_a)$ (with corresponding round complexities $O(n)$ and $O(1)$), where k is a security parameter, and the variation depends on the building blocks used (e.g., for random bit generation several protocols are known, which differ in complexities). In [16, Sect. 5], the protocol is proven statistically secure in the framework of Cramer et al. [5].

As an interesting application, this protocol can be used for secure and efficient statistical analysis on encrypted data. In [16, Sect. 8], a protocol for the computation of the variance of L inputs is constructed in detail. Other statistics can be implemented similarly. The possibility to securely evaluate statistics allows for a broad range of applications, like (medical) surveys. In medical surveys, many users release medical data to some institute which analyzes the data and outputs some result (a diagnostic, a result of statistical analysis, etc.). However, medical data are privacy sensitive and users might be unwilling to reveal these data in plaintext. Using secure multiparty computation, the institute is represented by a set of multiparty computation servers and the users can input their medical data *in encrypted form*. The servers then use the modulo reduction protocol for secure statistical analysis. We end by noticing that the protocol can easily be carried over to a client/server setting [16, Sect. 7], and that it has many other practical applications, for instance in the area of secure face recognition [10], packing of encrypted values [2] and auctions [7, 12]. In particular, a modulo reduction protocol allows for easily obtaining packed encrypted values out of one encryption.

1.2 Related Work

The relation with the bit decomposition protocols of [23] has already been discussed. For the unconditional setting using verifiable secret sharing, Algesheimer et al. [1] constructed a modulo reduction protocol which works for encrypted modulus a . The protocol relies on approximating $1/a$ (for which also a protocol by Kiltz et al. [18] can be used). This protocol is only of theoretical interest¹: instead of $x \bmod a$, the value $x \bmod a + ia$ is computed, with $|i| < (n+1)(5 + 2^{4+\ell_x-\ell_a-\ell'})$ for some additional security parameter ℓ' (the number of correctly approximated bits of $1/a$). The value $x \bmod a$ is then computed after $O(n2^{\ell_x-\ell_a-\ell'})$ executions of a comparison protocol, which makes the protocol inefficient. We note that our protocol does not rely on approximations. More comparable to ours is the VSS-based protocol by Damgård et al. [6], which opts for constant rounds. Unlike ours, their scheme does not make use of the form of a . In particular, in the THC-setting their protocol has a broadcast complexity varying between $O(nk\ell_x(\log \ell_x + \ell_a))$ and $O(nk\ell_x(n + \log \ell_x + \ell_a))$, where $\ell_x \geq \ell_a$. In many practical applications the value ℓ_a is even *much smaller* than ℓ_x , as exemplified in Sect. 5. Using ideas of [1], their protocol can also be extended to secret a . We stress that for our purposes the protocol with public a

¹ We note that From and Jakobsen [13, Ch. 8] discuss the efficiency of the protocol of [1]. They conclude that the performance is generally low, particularly for a large number of participants. See also [24, Sect. 4.6].

suffices. In [4], Catrina and Dragulin independently introduce a modulo reduction protocol similar to ours. However, unlike ours, their protocol is constructed for secure computation based on secret sharing and considers modulo reduction by powers of two only. Our protocol works for general a , and in particular relies on efficient ways for generating random values from $[0, a)$ securely. Moreover, [4] provides security against semi-honest adversaries only, while our protocol covers the malicious case.

Although our main concern is the modulo reduction protocol, we also consider related work with respect to statistical multiparty computation, which is used as motivational example. Many works on privacy-preserving statistical analysis (e.g., [9, 18]) focus only on techniques other than THC-based secure multiparty computation. In [17], computation of moments is considered for Paillier encryptions and used to compute statistics like the mean and the variance. The authors circumvent the need for a modulo reduction protocol by applying division on the decrypted moments only. This protocol is not of practical interest: if for instance $\sum_{i=1}^L x_i$ is decrypted rather than $(\sum_{i=1}^L x_i) \text{ div } L$, the protocol unintentionally leaks information about the inputs, namely $(\sum_{i=1}^L x_i) \bmod L^2$. Moreover, the protocol of [17] cannot be integrated as a sub-protocol with encrypted output, while this would be desirable in many applications like packing of encrypted values. In this sense the construction of the modulo protocol offers a new approach for privacy-preserving statistical computation.

2 Preliminaries

Throughout, we denote $[A, B] := \{A, A+1, \dots, B-1\}$. By ‘random’ we implicitly mean ‘uniformly randomly and independently distributed’, and we denote by $x \in_R V$ the event that x is taken at random from V .

PAILLIER CRYPTOSYSTEM. Our protocol relies on the additively homomorphic cryptosystem by Paillier [20], but we consider its generalization and its threshold variant by Damgård and Jurik [8]. On input of a security parameter k , the public key consists of an RSA modulus $N = pq$ of length k , for $p = 2p'+1$ and $q = 2q'+1$ safe primes, and a positive integer s . We define $m := p'q'$. The secret key is a value d coprime to N^s satisfying $d = 0 \bmod m$. The message space is the ring \mathbb{Z}_{N^s} , and a message x is encrypted by taking an $r \in_R \mathbb{Z}_{N^{s+1}}^*$ and computing $c = (N+1)^x r^{N^s} \bmod N^{s+1}$. For the threshold decryption, d is polynomially shared among the n participants, each participant has a share d_i , and at least t participants are required to correctly decrypt a ciphertext. This decryption protocol operates in constant rounds and has broadcast complexity $O(nk)$. A more detailed specification of the cryptosystem can be found in [8]. Encryptions are denoted by $\llbracket x \rrbracket$.

PROOFS OF KNOWLEDGE. Our modulo reduction protocol involves zero-knowledge proofs of knowledge in order to achieve security against malicious adversaries. We use standard Σ -protocols, which can be made non-interactive using

² Otherwise, if the value $x \bmod a$ is computed, the value $x \text{ div } a$ would leak.

the Fiat-Shamir heuristic and are provably secure in the random oracle model. In particular, our protocol involves interval proofs in which a prover shows that a published encryption $\llbracket x \rrbracket$ encrypts a value $x \in [A, B]$. For this, one can use the protocol by Boudot [3] (refined in [19, 15]). This protocol operates in constant rounds and has broadcast complexity $O(k)$.

2.1 Multiparty Computation Gates

The proposed protocol requires several efficient gates, which will be introduced in this section. Using efficient Σ -protocols, these gates handle the malicious case efficiently. We recall that the Paillier cryptosystem is additively homomorphic, which means that given encryptions $\llbracket x \rrbracket, \llbracket y \rrbracket$ and a public a , the encryptions $\llbracket x + y \rrbracket = \llbracket x \rrbracket \llbracket y \rrbracket$ and $\llbracket ax \rrbracket = \llbracket x \rrbracket^a$ can be computed non-interactively.

MULTIPLICATION. Cramer et al. [5] constructed a constant round protocol for n participants to securely compute $\llbracket xy \rrbracket$ given $\llbracket x \rrbracket, \llbracket y \rrbracket$. This protocol has broadcast complexity $O(nk)$.

RANDOM BIT GENERATION. Several multiparty protocols for generating random bits are known, varying between an $O(n^2k)$ broadcast complexity protocol in constant rounds [5], and an $O(nk)$ broadcast complexity protocol in $O(n)$ rounds [23].

COMPARISON GATE. On input of two encrypted bit representations $(\llbracket x_0 \rrbracket, \dots, \llbracket x_{\ell-1} \rrbracket)$ and $(\llbracket y_0 \rrbracket, \dots, \llbracket y_{\ell-1} \rrbracket)$, a comparison gate outputs an encrypted bit $\llbracket [x < y] \rrbracket$. An $O(\lg \ell)$ round complexity protocol [14], as well as a constant round protocol [6] are known, but the latter has a considerably higher hidden constant. Both protocols have broadcast complexity $O(nk\ell)$.

3 Random Bitwise Value Generation

The modulo reduction protocol introduced in Sect. 4 requires a sub-protocol to generate a value $r \in_R [0, a)$ in a bitwise manner. We refer to this gate as the *random bitwise value generation* protocol and we discuss such a protocol in this section. Other protocols for securely generating random values from a restricted domain are known as well [21].

Protocol 1 (Random bitwise value generation). Given a publicly known value a such that $2^{\ell_a-1} < a \leq 2^{\ell_a}$, the following protocol generates an encrypted bit representation $(\llbracket r_0 \rrbracket, \dots, \llbracket r_{\ell_a-1} \rrbracket)$ of r such that $r \in_R [0, a)$. The participants \mathcal{P}_i ($i = 1, \dots, n$) perform the following steps:

1. For $j = 0, \dots, \ell_a - 1$, the participants jointly generate random bit encryptions $\llbracket r_j \rrbracket$ for $r_j \in_R \{0, 1\}$;
2. Using a comparison gate, $\llbracket [r < a] \rrbracket$ is computed and jointly decrypted. If $[r < a] = 0$, the protocol is restarted.

Notice that in case $a \neq 2^{\ell_a}$, the number of restarts of the protocol is $2^{\ell_a}/a < 2$ on average. Using this observation, we conclude that Prot. 1 has broadcast complexity varying between $O(nk\ell_a)$ (with round complexity $O(n)$) and $O(n^2k\ell_a)$ (in constant rounds). Correctness and security are proven in [16, Propositions 1 and 3].

4 Multiparty Modulo Reduction

We consider input $\llbracket x \rrbracket$ with $x \in \{0, 1\}^{\ell_x}$ and a public value a such that $2^{\ell_a-1} < a \leq 2^{\ell_a}$ for some ℓ_a , and construct a protocol for the computation of $\llbracket x \bmod a \rrbracket$. Without loss of generality, we assume that $\ell_a \leq \ell_x$: clearly, if $\ell_a > \ell_x$ then certainly $a > x$, in which case $x \bmod a = x$. The protocol relies on the fact that it is unnecessary to compute the ℓ_x bits of x if the modulus $a \leq 2^{\ell_a}$ is known for some $\ell_a \leq \ell_x$. As in many cases ℓ_a is relatively small compared to ℓ_x (cf. Sect. 5), this reduces the costs. We recall that we have n participants (t, n)-threshold sharing the secret key for Paillier decryption, and that the public key for the cryptosystem is (N, s) . We introduce a security parameter ℓ_s , which we require to satisfy $an2^{\ell_x+\ell_s} < N^s$.

Protocol 2 (Modulo reduction). Given $\llbracket x \rrbracket$ for $x \in \{0, 1\}^{\ell_x}$ and a publicly known value a , the following protocol outputs an encryption $\llbracket x \bmod a \rrbracket$. The participants \mathcal{P}_i ($i = 1, \dots, n$) perform the following steps:

1. The participants jointly generate a random encrypted bit representation $(\llbracket r_0 \rrbracket, \dots, \llbracket r_{\ell_a-1} \rrbracket)$ of r such that $r \in_R [0, a]$, using Prot. 1. In parallel, each participant takes $s_i \in_R \{0, 1\}^{\ell_x+\ell_s}$ and publishes $S_i = \llbracket s_i \rrbracket$ together with an interval proof of knowledge for relation $\{(S_i; s_i) \mid S_i = \llbracket s_i \rrbracket \wedge s_i \in [0, 2^{\ell_x+\ell_s}] \}$;
2. Each participant individually computes

$$\llbracket \tilde{x} \rrbracket = \llbracket x \rrbracket \llbracket r \rrbracket^{-1} \left(\prod_{i=1}^n \llbracket s_i \rrbracket \right)^a = \left[\llbracket x - r + a \sum_{i=1}^n s_i \rrbracket \right];$$

3. Using threshold decryption the participants obtain \tilde{x} , and compute $\bar{x} = \tilde{x} \bmod a$;
4. Using a comparison gate the participants compute $\llbracket c \rrbracket = \llbracket [a - 1 - \bar{x} < r] \rrbracket$;
5. Each participant individually computes

$$\text{mod}(\llbracket x \rrbracket, a) = \llbracket \bar{x} \rrbracket \llbracket r \rrbracket \llbracket c \rrbracket^{-a} = \llbracket \bar{x} + r - ca \rrbracket.$$

Notice that in phase 4 the comparison gates of Sect. 2.1 can be used, as $a - 1 - \bar{x}$ is known in plaintext, and the participants know the encrypted bit representation of r . Correctness and security are proven in [16, Propositions 2 and 4].

5 Efficiency Analysis

The modulo reduction protocol has average broadcast complexity varying between $O(nk\ell_a)$ (in $O(n)$ rounds) and $O(n^2k\ell_a)$ (in constant rounds). The absolute number of rounds highly depends on the gates used. In [6], Damgård et al. also construct a modulo reduction gate, although for verifiable secret sharing. The THC-analogue of this gate is less efficient than the one proposed here. Their protocol has a broadcast complexity varying between $O(nk\ell_x(\log \ell_x + \ell_a))$ and $O(nk\ell_x(n + \log \ell_x + \ell_a))$ (with round complexities $O(n + \ell_x)$ and $O(1)$, respectively). More importantly, in many practical applications the value ℓ_a is rather

small compared to ℓ_x : consider for example a scenario where 100 millionaires want to securely compute an encryption of their average fortune. In this case $\ell_a = 7$, while $\ell_x = 37$ but needs to be extended to 47 to cover billionaires as well³. Note that Damgård et al.'s construction needs to compute the complete bit representation of x , while the idea of the proposed scheme relies on knowledge of the form of a .

Acknowledgments

This work has been funded in part by the European Community's Sixth Framework Programme under grant number 034238, SPEED project - Signal Processing in the Encrypted Domain, in part 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. The work reported reflects only the authors views; the European Community is not liable for any use that may be made of the information contained herein.

References

- [1] 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)
- [2] Bianchi, T., Piva, A., Barni, M.: Efficient pointwise and blockwise encrypted operations. In: MM&Sec 2008, pp. 85–90. ACM, New York (2008)
- [3] 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)
- [4] Catrina, O., Dragulin, C.: Multiparty computation of fixed-point multiplication and reciprocal. In: DEXA 2009, pp. 107–111. IEEE Computer Society, Los Alamitos (2009)
- [5] Cramer, R., Damgård, I., Nielsen, J.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–300. Springer, Heidelberg (2001)
- [6] Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J., 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)
- [7] Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for online auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007)
- [8] Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)

³ For simplicity we assume that the fortune of a millionaire is upper bounded by one billion.

- [9] Du, W., Atallah, M.: Privacy-preserving cooperative statistical analysis. In: ACSAC 2001, pp. 102–112. IEEE Computer Society, Los Alamitos (2001)
- [10] Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 235–253. Springer, Heidelberg (2009)
- [11] Franklin, M., Haber, S.: Joint encryption and message-efficient secure computation. *Journal of Cryptology* 9(4), 217–232 (1996)
- [12] Franklin, M., Reiter, M.: The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering* 22(5), 302–312 (1996)
- [13] From, S., Jakobsen, T.: Secure multi-party computation on integers. Master’s thesis, University of Århus, Århus (2006), <http://www.cs.au.dk/~tpj/uni/thesis/report.pdf>
- [14] Garay, J., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007)
- [15] 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)
- [16] Guajardo, J., Mennink, B., Schoenmakers, B.: Modulo reduction for Paillier encryptions and application to secure statistical analysis. Full version of this paper, available from the authors (2009)
- [17] Kiayias, A., Yener, B., Yung, M.: Privacy-preserving information markets for computing statistical data. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 32–50. Springer, Heidelberg (2009)
- [18] 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)
- [19] 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)
- [20] 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)
- [21] Schoenmakers, B., Sidorenko, A.: Distributed generation of uniformly random bounded integers (October 1, 2007) (unpublished manuscript)
- [22] Schoenmakers, B., Tuyls, P.: Practical two-party computation based on the conditional gate. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 119–136. Springer, Heidelberg (2004)
- [23] Schoenmakers, B., Tuyls, P.: Efficient binary conversion for Paillier encrypted values. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 522–537. Springer, Heidelberg (2006)
- [24] SecureSCM. Secure computation models and frameworks. Technical Report D9.1, SecureSCM (July 2008), <http://www.securescm.org>

On Robust Key Agreement Based on Public Key Authentication

(Short Paper)

Feng Hao

Thales E-Security, Cambridge, UK
feng.hao@thales-esecurity.com

Abstract. We describe two new attacks on the HMQV protocol. The first attack raises a serious question on the basic definition of “authentication” in HMQV, while the second attack is generally applicable to many other protocols. In addition, we present a new authenticated key agreement protocol called YAK. Our approach is to depend on well-established techniques such as Schnorr’s signature. Among all the related protocols, YAK appears to be the simplest so far. We believe simplicity is an important engineering principle.

1 Introduction

There are two categories of authenticated two-party key agreement protocols: Password Authenticated Key Exchange (PAKE) and Authenticated Key Exchange (AKE) [9]. The former realizes authentication based on a shared password, while the latter based on public key certificates [2, 5, 4, 1, 6]. In this paper, we focus on discussing the second category. To better differentiate it from the first category, we will call it Public Key Authenticated Key Exchange (PK-AKE).

2 Past Work

Many PK-AKE protocols claim to be provably secure in a formal model. Among them, the HMQV scheme is perhaps the most well-known example [2]. In this section, we will show two new attacks on HMQV.

The HMQV protocol is modified from MQV [6] with the primary aim for provable security [2]. The most significant change is that HMQV drops some mandated verification steps in MQV, including the Proof of Possession (PoP) check during the CA registration and the prime-order validation check of the ephemeral public key.

Dropping the public key validations is highly controversial, despite that HMQV has formal security proofs. In one attack, Menezes *et al.* demonstrated disclosing the user’s private key without violating the HMQV model definition [8, 7]. This attack indicates a flaw in the original design of HMQV.

In acknowledgement of the missing public key validation, Krawczyk revised HMQV in the submission to IEEE P1363 Standards committee [3]. He added the following validation: Alice checks the term YB^e has the correct prime order

and Bob does the same for XA^d (see [2], p. 548, for the definition of symbols.) This change prevents the small subgroup attack in [8], but decreases the claimed efficiency. However, instead of validating the static and ephemeral public keys separately as in MQV, the revision chooses to optimize efficiency by mixing the two operations together. This causes the problem as below.

We now report a new “invalid public key attack” on HMQV. For illustration, we follow the same symbols used in the original description of HMQV (see [2], p. 548). In both the original and revised versions of HMQV, the CA is only required to check the submitted public key is not 0. The attack works as follows. Assume Bob (attacker) registers a small subgroup element $s \in G_w$ as the public key where $w|p - 1$. Bob chooses an arbitrary value $z \in Z_q$. Let $Y = g^z \cdot s'$ where s' is an element in the same small subgroup G_w . Exhaustively, Bob tries every element s' in G_w such that $YB^e = g^z \cdot s' \cdot s^e = g^z$. In other words, the small subgroup elements s and s' cancel each other out. Suppose \bar{H} works like a random oracle as assumed in HMQV. Then, for each try of s' , the probability of finding $s' \cdot s^e = 1$ is $1/w$. It will be almost certain to find such s' after searching all w elements in G_w (if not then change a different z and repeat the procedure). Following the HMQV protocol, Bob sends $Y = g^z \cdot s'$ to Alice. Alice checks YB^e has the correct prime order and computes the session key $\kappa = H((YB^e)^{x+da}) = H(g^{z \cdot (x+da)})$. Because Bob knows z , he can compute the same session key κ and successfully authenticate himself to Alice.

The fact that an obviously invalid public key is totally undetected by all flows in HMQV is unsettling. This raises a serious question on the basic definition of “authentication” in HMQV – Bob does not even have a private key, yet he is able to successfully pass all authentication checks. In fact, anyone can do the same pre-computation as above and authenticate to Alice as “Bob”. In one attacking scenario, Bob (the attacker) may at any time suddenly repudiate all previous authenticated transactions with Alice by telling the judge that his public key is invalid, so anyone can impersonate him. (Bob’s certificate is publicly available.) In comparison, MQV does not have this problem.

The other attack on HMQV happens when two parties use the same certificate during self-communication [2]. Self-communication is a useful application in practice. For example, a mobile user and the desktop computer may hold the same static private key (registering two public key certificates costs more). Krawczyk formally proved that self-communication is “secure” in HMQV [2]. However, the formal model in [2] only considers the user talking to one copy of self, but neglects the possibility that the user may talk to multiple copies of self at the same time. This deficiency is common among other formal models too [11, 4, 1]. The attack works as follows (also see Figure 1):

1. Alice initiates the connection to a copy of herself by sending g^x . The connection is intercepted by Mallory who pretends to be Alice-1.
2. Mallory starts a separate session by pretending to be Alice-2. He initiates the connection by sending to Alice g^x (this is possible because HMQV does not require the sender to know the exponent).
3. Alice responds to Alice-2 by sending g^y .

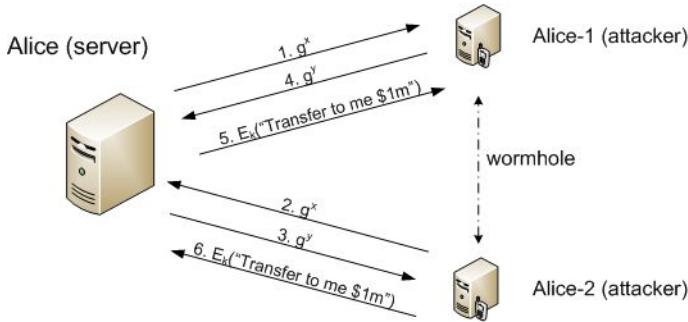


Fig. 1. Wormhole attack on HMQV

4. Mallory replays g^y to Alice as Alice-1.
5. Alice derives a session key and sends an encrypted message to Alice-1, say: “Transfer to me \$1m”.
6. Mallory replays the encrypted message to Alice. (After receiving money from Alice, Mallory disconnects both connections.)

In the above attack, we only demonstrated the attack against the two-pass HMQV (implicit authentication). For the three-pass HMQV (explicit authentication), the attack works exactly the same. Also, we have omitted the identities in the message flows, because they are all identical according to the HMQV specification [2].

This attack is essentially an unknown key sharing attack. Alice thinks she is communicating to a mobile user with the same certificate, but she is actually communicating to herself. The attacker does not hold the private key, but he manages to establish two fully authenticated channels with Alice (server). The same attack also applies to other PK-AKE schemes, including NAXOS [4], KEA+ [5], CMQV [11], MQV [6], and SIG-DH [1] etc, despite that many of them have formal security proofs.

3 The YAK Protocol

In this section, we propose a new PK-AKE protocol called YAK¹. Let G denote a subgroup of Z_p^* with prime order q in which the Computational Diffie-Hellman problem (CDH) is intractable. Let g be a generator in G . The two communicating parties, Alice and Bob, both agree on (G, g) .

3.1 Stage 1: Public Key Registration

In stage 1, Alice and Bob register static public keys from a Certificate Authority (CA). Alice selects a random secret $a \in_R Z_q$ as her private key. Similarly, Bob selects $b \in_R Z_q$ as his private key.

¹ The yak lives in the Tibetan Plateau where environmental conditions are extremely adverse.

CA-Registration. Alice sends to the CA g^a with a knowledge proof for a . Similarly, Bob sends to the CA g^b with a knowledge proof for b .

The sender needs to produce a valid knowledge proof to demonstrate the Proof of Possession (PoP) of the private key. As an example, we can use Schnorr's signature, which is provably secure in the random oracle model [9]. Let H be a secure hash function. To prove the knowledge of the exponent for $X = g^x$, one sends $\{\text{SignerID}, \text{OtherInfo}, V = g^v, r = v - x \cdot h\}$ where SignerID is the *unique* user identifier (also called Distinguished Name [10]), OtherInfo includes auxiliary information to indicate this is a request for certifying a static public key and may include other practical information such as the name of the algorithm etc, $v \in_R Z_q$ and $h = H(g, V, X, \text{SignerID}, \text{OtherInfo})$. The CA checks that X has prime order q and verifies that $V = g^r X^h$ (computing $g^r X^h$ requires roughly one exponentiation using the simultaneous computation technique [9]).

3.2 Stage 2: Key Agreement

Alice and Bob execute the following protocol to establish a session key. For simplicity of discussion, we explain the case that Alice and Bob have different certificates ($a \neq b$) and will cover self-communication later.

YAK-protocol. Alice selects $x \in_R Z_q$ and sends out g^x with a knowledge proof for x . Similarly, Bob selects $y \in Z_q$ and sends out g^y with a knowledge proof for y .

When this round of communication finishes, Alice and Bob verify the received knowledge proof to ensure the other party possesses the ephemeral private key. They also need to ensure the identity (i.e., SignerID) in the knowledge proof must match the one in the public key certificate.

Upon successful verification, Alice computes a session key $\kappa = H((g^y \cdot g^b)^{x+a}) = H(g^{(x+a)(y+b)})$. And Bob computes the same session key: $\kappa = H((g^x \cdot g^a)^{y+b}) = H(g^{(x+a)(y+b)})$.

In YAK, Alice needs to perform the following exponentiations: one to compute an ephemeral public key (i.e., g^x), one to compute the knowledge proof for x (i.e., g^{v_x}), two to verify the knowledge proof for the exponent of $Y = g^y$ (i.e., Y^q and $g^{r_y} Y^{h_y}$) and finally one to compute the session key $(Y \cdot B)^{x+a}$. Thus, that is five in total: $\{g^x, g^{v_x}, Y^q, g^{r_y} Y^{h_y}, (Y \cdot B)^{x+a}\}$.

Among the above operations, some are merely repetitions. To explain this, let the bit length of the exponent be $L = \log_2 q$. Then, computing g^x alone would require roughly $1.5L$ multiplications which include L square operations and $0.5L$ multiplications of the square terms. However, the same square operations need not be repeated for other items with the common base. If we factor this in, it will take $(1+0.5 \times 3)L = 2.5L$ to compute $\{g^x, g^{v_x}, g^{r_y}\}$, and another $(1+0.5 \times 2)L = 2L$ to compute $\{Y^q, Y^{h_y}\}$ and finally $1.5L$ to compute $(Y \cdot B)^{x+a}$. Hence, that is in total $6L$, which is equivalent to $6L/1.5L = 4$ usual exponentiations. This is quite comparable to the 3.5 exponentiations in MQV (which cannot reuse the square terms since the bases are different).

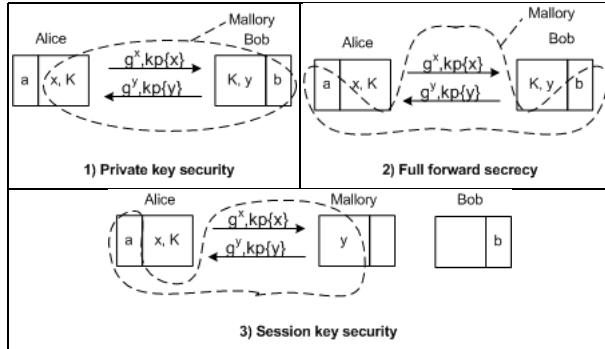


Fig. 2. The oracle diagrams in YAK. Alice is honest.

4 Security Analysis

We formulate the following requirements for the PK-AKE protocol.

1. **Private key security:** An attacker cannot learn any useful information about the user’s static private key even if he is able to learn all session specific secrets in any session.
2. **Full forward secrecy:** Session keys that were securely established in the past uncorrupted sessions will remain incomputable in the future even when both users’ static private keys are disclosed.
3. **Session key security:** An attacker cannot compute the session key if he impersonates a user but has no access to the user’s private key.

The first requirement is generally not covered by a formal model, but we think it is crucially important. For example, both the SIG-DH [1] and (original) HMQV [2] protocols have been formally proven secure in the CK model. Yet attacks reported in [4] and [8] show that in both protocols, an attacker is able to disclose the user’s private key. In the second requirement, we use “full” to distinguish it from the “half” forward secrecy, which only allows one user’s private key to be revealed (e.g., KEA+ [5]). The third requirement has already covered the Key Compromise Impersonation (KCI) attack [6]. The “invalid public key” attack in Section 2 indicates that HMQV does not satisfy this property.

The goal of our design is to make the best use of well-established techniques such as Schnorr’s signature. This strategy allows us to leverage upon the provable results of Schnorr’s signature (see [9]), and thus greatly simplify the security analysis. In the following, we will provide a simple and intuitive analysis, while leaving detailed proofs to a full paper.

First, let us discuss the private key security. Without loss of generality, we assume Alice is honest. As shown in Figure 2 (1), Mallory totally controls Bob’s static and ephemeral private keys. Additionally, he has the extreme power that allows him to learn Alice’s transient secrets in an arbitrary session. The only power that he does not have is the access to Alice’s private key.

A sketch of the proof goes as follows. The knowledge proofs defined in YAK prove that Mallory (the attacker) knows the value of y and b . He also knows Alice's public key g^a . By revealing Alice's transient secrets (i.e., x and K) in a session, he learns x and $K = g^{(a+x)(b+y)}$. But learning K does not give Mallory any information, because he can compute it by himself from $\{x, y, b, g^a\}$. Effectively, Mallory can actually simulate the attack all by himself through defining arbitrary values of x, y , and b . Clearly, he does not learn any useful information about Alice's private key through his own simulations.

Next, we discuss the full forward secrecy requirement. The definition (see Section 2) specifies that the past sessions must be “uncorrupted”, namely the session-specific transient secrets must remain unknown to the attacker. In YAK, this means x, y and K must remain unknown to the attacker. Obviously, knowing K would have trivially broken the past session. Also, if Mallory can learn any ephemeral exponent x or y in the past session in addition to knowing both parties' static private keys, he has possessed the power to trivially compromise any PK-AKE. Therefore, as shown in Figure 2 (2), we assume the attacker knows both Alice and Bob's private keys, but not any transient secrets in the past session.

We explain the YAK's fulfillment of the full forward secrecy under the Computational Diffie-Hellman (CDH) assumption. To obtain a contradiction, we assume the attacker can compute $K = g^{(a+x)(b+y)}$. The attacker knows the values of a and b (see Figure 2 (2)). The ephemeral public keys g^x and g^y are public information. Therefore, Mallory can compute g^{ab}, g^{ay} and g^{bx} . Now, we can solve the CDH problem as follows: given g^x and g^y where $x, y \in_R Z_q$, we use Mallory as an oracle to compute $g^{xy} = K/(g^{ab} \cdot g^{ay} \cdot g^{bx})$. This, however, contradicts the CDH assumption.

Finally, we discuss the session key security requirement. As shown in Figure 2 (3), Mallory does not hold Bob's private key but he tries to impersonate Bob. We assume the powerful Mallory even knows Alice's private key a . The only power he does not have is the access to Alice and Bob's session states. If Mallory can access Alice's session state, he can impersonate anyone to Alice – he just needs to “steal” the session key that Alice computes in the transient memory. Similarly, if Mallory can access Bob's session state, he can impersonate Bob to anybody by waiting until Bob computes the session key and then stealing it.

In this case, the assumed attacker is less powerful than the one described in the “private key security” argument. Previously, the attacker was able to corrupt an arbitrary session of Alice's or Bob's. He however had learned no useful information than what he can simulate. On discussing the session key security, we assume the attacker no longer has access to either user's session state. This change is necessary, and is consistent with the extreme-adversary principle [4]: the only powers that an attacker does not have are those that would allow him to trivially break any PK-AKE protocol.

The YAK protocol satisfies the session key security requirement under the CDH assumption. As shown in Figure 2 (3), Mallory does not possess Bob's static private key, or have access to either Alice or Bob's session state. To obtain

a contradiction, we assume Mallory is able to compute $K = g^{(a+x)(b+y)}$. Bob's public key g^b is public information. Mallory knows Alice's private key a . The knowledge proof in the protocol proves that Mallory also knows the value y . Hence, he can compute g^{ab} , g^{ay} and g^{xy} . Now, we can solve the CDH problem as follows: given g^b and g^x where $x, b \in_R Z_q$, we use Mallory as an oracle to compute $g^{bx} = Z/(g^{ab} \cdot g^{ay} \cdot g^{xy})$. This, however, contradicts the CDH assumption, which shows YAK satisfies the session key security requirement.

5 Self-communication

The user identity is an important parameter in the protocol definition. In the past literature, almost all PK-AKE protocols use the Distinguished Name (DN) in the user's X.509 certificate as the user identity [1, 2, 5, 6, 4]. This practice also carries over to the self-communication mode [2], which causes the "wormhole attack" (see Section 2). In the self-communication mode, the two parties are still distinct entities and hence, naturally require different identities.

To enable self-communication in YAK, we need to ensure the SignerID in the Schnorr's signature remains unique. This is to prevent Bob from replaying Alice's signature back to Alice and vice versa. One solution is to simply attach an additional identifier to the mobile stations using the same certificate. For example, when Alice (server) is communicating to the n th copy of herself (mobile station), Alice uses "Alice" as her SignerID to generate the Schnorr's signature and the n th copy uses "Alice- n " as its SignerID. Thus, Alice- n cannot replay Alice's signature back to Alice and vice versa. This solution is also generically applicable to fix the self-communication problem in past protocols [1, 2, 5, 6, 4].

Though self-communication is considered a useful feature [2], one should be careful to enable this feature only when it is really needed. This is because, when enabled, it may have negative impact on the theoretical security. In Section 4, under the "private key security", we have explained that, under normal operations (using different certificates $a \neq b$), an attacker cannot learn $g^{a \cdot a}$ from a corrupted session. However, if self-communication is enabled in YAK, we essentially allow $a = b$, hence the attacker can learn $g^{a \cdot a}$ from a corrupted session. This implies we would need a stronger assumption than CDH to prove the "session key security". This is undesirable, but to our best knowledge, no PK-AKE protocol is reducible to the CDH assumption with the self-communication enabled. In comparison, in HMQV [2], the attacker can learn $g^{a \cdot a}$ from a corrupted session regardless whether the self-communication is enabled.

6 Conclusion

In this paper, we report two new attacks on the HMQV protocol. In addition, we present a new authenticated key agreement protocol called YAK, and analyze its robustness in an extremely adverse condition: the only powers that an attacker does not have are those that would allow him to trivially break any other

protocols. Overall, YAK demonstrates robust security under the Computational Diffie-Hellman assumption in the random oracle model.

Acknowledgment

We thank Alfred Menezes and Berkant Ustaoglu for their generous advice and invaluable comments. We thank Lihong Yang for helping improve the readability.

References

1. 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)
2. 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), <http://eprint.iacr.org/2005/176.pdf>
3. Krawczyk, H.: HMQV in IEEE P1363. Submission to the IEEE P1363 Standardization Working Group (2006), <http://grouper.ieee.org/groups/1363/P1363-Reaffirm/submissions/krawczyk-hmqv-spec.pdf>
4. 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)
5. Lauter, K., Mityagin, A.: Security Analysis of KEA Authenticated Key Exchange Protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
6. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An Efficient Protocol For Authenticated Key Agreement. Designs, Codes and Cryptography 28(2), 119–134 (2003)
7. Menezes, A.: Another Look At HMQV. J. of Mathematical Cryptology 1(1), 47–64 (2007)
8. Menezes, A., Ustaoglu, 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)
9. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
10. Mitchell, C.: Security For Mobility. The Institution of Electrical Engineers (2004)
11. Ustaoglu, B.: Obtaining A Secure And Efficient Key Agreement Protocol For (H)MQV And NAXOS. Designs, Codes and Cryptography 46(3), 329–342 (2008)

A Formal Approach for Automated Reasoning about Off-Line and Undetectable On-Line Guessing (Short Paper)

Bogdan Groza and Marius Minea

Politehnica University of Timișoara and Institute e-Austria Timișoara^{*}
`bogdan.groza@aut.upt.ro, marius@cs.upt.ro`

Abstract. Starting from algebraic properties that enable guessing low-entropy secrets, we formalize guessing rules for symbolic verification. The rules are suited for both off-line and on-line guessing and can distinguish between them. We add our guessing rules as state transitions to protocol models that are input to model checking tools. With our proof-of-concept implementation we have automatically detected guessing attacks in several protocols. Some attacks are especially significant since they are undetectable by protocol participants, as they cause no abnormal protocol behavior, a case not previously addressed by automated techniques.

1 Motivation and Related Work

As password-based authentication continues to be used in practice and weak passwords are still chosen by users, detecting protocols subject to guessing attacks is a topic of high interest in security. In this paper we address the problem of formalizing a previously introduced approach to detect guessing attacks in a manner suitable for implementation in an automated verification toolset. We use IF (Intermediate Format), a specification language that can be handled by model checkers such as OFMC (Open Source Fixedpoint Model-Checker) [3] and SATMC (SAT-based Model Checker) [2] from the AVISPA toolset.

A previous intention of integrating guessing rules in OFMC exists in [9], which gives a formalization for off-line guessing attacks. In comparison, our contribution proposes a different formalism (with guessing rules based on a different reasoning), which allows us to handle both on-line and off-line attacks. Our guessing rules are implemented at the level of the protocol description language, without requiring the modification of the back-end model checkers. Other concrete implementations of guessing detection rules are by Corin et al. [7], Lowe [13] who used Casper/FDR and Blanchet [5] in ProVerif, a verifier based on Prolog rules. Our implementation is based on IF, a specification language which can be handled by several back-end model checkers, notably OFMC and SATMC, which thus gain the ability of detecting guessing attacks. Other theoretical foundations for

^{*} This work is supported in part by FP7-ICT-2007-1 project 216471, AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures.

reasoning about guessing attacks exist. Abadi et al. [1] use indistinguishability of two terms, deduced by static equivalence, to formalize guessing. Equational theories for the applied pi-calculus are used by Corin et al. in [6], while Baudet [4] uses a constraint solving algorithm for an equational theory.

Our guessing rules are based on the pseudo-randomness properties of one-way functions. We consider two cases of guessing: first, when the adversary knows the image of a one-way function computed on the secret and other known input; second, when the adversary knows the image of a trapdoor function (encryption) with a key that depends on the secret, and can establish relations on its input. As one-way functions are pseudo-random, the output for a wrong secret cannot match any previously known value, thus a correct guess can be verified.

Most prior work addresses only off-line guessing, considering the low-entropy secret large enough to prevent guessing, or that unsuccessful attempts can be blocked. However, in some on-line attacks the protocol behavior is indistinguishable from normal. These attacks are undetectable by participants and especially dangerous. They are also realistic, as one of our case studies, a Norwegian ATM system, illustrates. Undetectable on-line attacks have also been the focus of Ding and Horster [8], but without a formalization or automated detection.

2 Formalization of Guessing Rules

To express the feasibility of guessing, we use, as in [10], the concept of *strongly distinguishing* functions, which cannot give the same output for two different secrets when these are paired with sufficiently many input choices.

Definition 1. Given $\sigma \in \{0,1\}^k$, we call a function $f(\sigma, x)$ *strongly distinguishing* in the first argument after q queries, if given any q distinct values $\{x_1, x_2, \dots, x_q\}$, $\forall s_1 \neq s_2$ the probability that $f(s_1, x_i) = f(s_2, x_i)$ for all $i = 1 \dots q$ is at most 2^{-k} , i.e., $\forall s_1 \neq s_2 . Pr[f(s_1, x_i) = f(s_2, x_i), i = 1 \dots q] \leq 2^{-k}$.

Using strongly distinguishing functions, we have characterized the conditions for an intruder's guess in a *guessing lemma* [10]. However, due to its algebraic rather than symbolic formulation, it cannot be directly implemented in a formal verification tool. Therefore, we will link the concept of *strongly distinguishing* function (in one query) with a *symbolic* protocol description.

Protocol descriptions contain *terms*, which are either *atomic* or *composed*. *Atomic* terms are *variables*, *constants* or *numbers*; *composed* terms are formed by applying *pair*, *crypt*, *inv* and other predefined operators on *atomic* terms. *Facts* are predicates defined over terms, such as *iknows*, *state*, *contains*, etc.

Definition 2. We call a *symbolic protocol description* \mathcal{P} a triple composed of an *initial state*, a set of *transition rules* and a set of *attack states*, i.e., $\mathcal{P} = (InitialState, TransitionRule^*, AttackState^*)$, where: i) the *initial state* is a conjunction of ground facts, ii) a *transition rule* has the form $LHS \Rightarrow RHS$ where *LHS* and *RHS* are conjunctions of facts, and *LHS* may also contain a negated fact and a *condition* (a conjunction of term equalities and inequalities), iii) an *attack state* is a conjunction of facts with a condition (like a *LHS*).

To reason about guessing, we define derivation rules $P \vdash_r T$, denoting that term T can be derived from term set P using rule r . Rule $\vdash_{\text{ihears}} T$ denotes that a term T has been overheard by the intruder during protocol execution.

Denote the set of symbols (constants or variables) appearing in term T by $S(T)$. If $s \in S(T)$ is such a symbol, we also write $T \vdash_{\text{part}} s$.

Let $T \vdash_{s \leftarrow \text{gen}(s')} T'$ denote that term T' is obtained by substituting any occurrence of the symbol s in T with a fresh symbol $s' \notin S(T)$. For instance, $\text{crypt}(s, m) \vdash_{s \leftarrow \text{gen}(s')} \text{crypt}(s', m)$. As a special case, we write $T \vdash_{s \leftarrow \text{igen}(s')} T'$ if s is substituted by a fresh value chosen by the intruder.

Consider a valuation function v defined on atomic terms with algebraic values, and extended to composed terms through function and operator application. We now relate our symbolic reasoning to the algebraic properties of the protocol.

Definition 3. A *symbolic protocol description* \mathcal{P} is called *algebraically dependent* on symbol s , denoted $\mathcal{P} \text{ dep } s$ if for any term T such that $\vdash_{\text{ihears}} T$ and $T \vdash_{\text{part}} s$, and considering T' such that $T \vdash_{s \leftarrow \text{gen}(s')} T'$, for any valuation v with $v(s) \neq v(s')$, we have $v(T) \neq v(T')$.

Given $s \in S(T)$, denote by $O_s^T(\cdot)$ the oracle corresponding to the function obtained by making s a variable in T and keeping other parts of it constant, e.g., $O_s^{\text{crypt}(s, m)}(\cdot)$ is the oracle corresponding to $f(s) = \text{crypt}(s, m)$.

Lemma 1. The *symbolic protocol description* \mathcal{P} is *algebraically dependent* on s , i.e., $\mathcal{P} \text{ dep } s$, if and only if any function f obtained as $O_s^T(\cdot)$ where $s \in S(T)$ and $\vdash_{\text{ihears}} T$ is *strongly distinguishing* in one query.

Lemma 1 relates a *symbolic protocol description* with the algebraic notion of *strongly distinguishing* function. Since injective functions are *strongly distinguishing* in one query, any symbolic protocol description in which a symbol s occurs only in the body of a injective (bijective) function is algebraically dependent on s . In practice, this covers a large class of protocols, since most cryptographic functions are bijective (as are hash functions, if assumed collision-free).

Definition 4. An adversary *observes* an oracle for a secret s if it *hears* a term that contains s . The adversary *controls* an oracle for secret s if by replacing s in a term with a fresh s' (rule $\vdash_{s \leftarrow \text{igen}(s')}$) the adversary *knows* the new term.

$$\vdash_{\text{ihears}} T \wedge T \vdash_{\text{part}} s \Rightarrow \text{observes}(O_s^T(\cdot)) \quad (1)$$

$$\vdash_{\text{ihears}} T \wedge T \vdash_{s \leftarrow \text{igen}(s')} T' \wedge \vdash_{\text{iknows}} T' \Rightarrow \text{controls}(O_s^T(\cdot)) \quad (2)$$

Lemma 2. Consider a *symbolic protocol description* \mathcal{P} such that $\mathcal{P} \text{ dep } s$. If an adversary *observes* and *controls* an oracle for a low-entropy secret s then the adversary can guess the secret s , i.e.,

$$\text{observes}(O_s^f(\cdot)) \wedge \text{controls}(O_s^f(\cdot)) \Rightarrow \text{guess}(s) \quad (3)$$

This first guessing case involves observing and controlling a one-way function (oracle) that is strongly distinguishing in the secret. A second case uses invertible functions. The adversary can also guess if he or she *observes* messages encrypted

with a key computed as a *strongly distinguishing* function on the secret, controls the corresponding decryption oracle, and can establish a relation to one or several parts of the encrypted messages. We formalize this case as follows:

Definition 5. We call *s-dependent* an encryption or decryption oracle that uses a key containing s . An adversary that *hears* the encryption of some message with a key that contains s is said to *observe* an *s-dependent* encryption oracle. Moreover, we say that he *controls* the corresponding *s-dependent* decryption oracle if by replacing s in the encryption key with a fresh s' known to him the adversary can decrypt arbitrary messages encrypted with the new key, i.e.,

$$\vdash_{\text{ihears}} \{M\}_K \wedge K \vdash_{\text{part}} s \Rightarrow \text{observes}(O_s^{\{M\}_K}(\cdot)) \quad (4)$$

$$\{M\}_K \vdash_{\substack{s \leftarrow \text{igen}(s') \\ M \leftarrow \text{gen}(M')}} \{M'\}_{K'} \wedge \vdash_{\text{iknows}} M' \Rightarrow \text{controls}(O_s^{\{M\}_{K-1}}(\cdot)) \quad (5)$$

Here, $\{M\}_K$ is the encryption of message M with key K . To keep relation (5) simple, we've left implicit that the adversary must overhear the term $\{M\}_K$ and the encryption key must contain s , i.e., $\vdash_{\text{ihears}} \{M\}_K \wedge K \vdash_{\text{part}} s$ as a premise. This is of course needed for the question of controlling the oracle to make sense.

To express a relation between encrypted inputs we employ a derivation rule $\text{Fact} \vdash_{\text{concat}}^M T$ to produce all distinct messages M that satisfy a property $\text{Fact}(M)$, by concatenating them into term T . For example, $(\vdash_{\text{ihears}} M) \vdash_{\text{concat}}^M T$ yields a term T that is the concatenation of all distinct terms for which $\vdash_{\text{ihears}} M$ holds. Similarly, $(\vdash_{\text{ihears}} \{M\}_K \wedge K \vdash_{\text{part}} s) \vdash_{\text{concat}}^M T$ produces the concatenation of all distinct messages that are encrypted with a key that contains s . Also, let $T \vdash_{\text{split}} \langle T', T'' \rangle$ denote that T' and T'' are derived by splitting T into disjoint subsets of terms (at least one of them non-empty).

The second guessing rule provides powerful capabilities: to find a relation between two terms (the *relates* fact) the adversary can use any available operators: *pair*, *crypt*, etc., as well as his Dolev-Yao abilities, *fake*, *overhear*, etc. Thus, for deciding *relates* the adversary can perform any transition allowed by the *symbolic protocol description* \mathcal{P} . The following definition models this intuition.

Definition 6. An adversary can *relate* two terms T' and T'' of a *symbolic protocol description* \mathcal{P} if by adding T' to the adversary knowledge he can derive T'' (denoted $T' \vdash_{DY(\mathcal{P})} T''$) using *all* his abilities over \mathcal{P} .

$$T' \vdash_{DY(\mathcal{P})} T'' \Rightarrow \text{relates}(T', T'') \quad (6)$$

Lemma 3. Let \mathcal{P} be a *symbolic protocol description* such that $\mathcal{P} \text{ dep } s$. If the adversary *observes* one or more *s-dependent* encryption oracles for which he or she *controls* the corresponding decryption oracles and can *relate* parts of the encrypted messages then the adversary can guess the secret, i.e.,

$$\begin{aligned} & \text{observes}(O_s^{\{M\}_K}(\cdot)) \wedge \text{controls}(O_s^{\{M\}_{K-1}}(\cdot)) \vdash_{\text{concat}}^M T \\ & \wedge T \vdash_{\text{split}} \langle T', T'' \rangle \wedge \text{relates}(T', T'') \Rightarrow \text{guess}(s) \end{aligned} \quad (7)$$

3 Implementation and Experimental Results

Our formalization of the guessing calculus makes it amenable to an implementation where states are sets of terms, and transitions are given as rewrite rules, as in the IF protocol specification language. Derivations such as \vdash_{ihears} , \vdash_{part} , $\vdash_{s \leftarrow \text{gen}(s')}$, \vdash_{split} yield corresponding IF facts. These are combined into rules to establish the relations *observes* and *controls*, and ultimately, guessing.

We use an adversary model with standard Dolev-Yao abilities: the adversary can fake new messages, intercept sent messages or overhear them. Moreover, the adversary has the standard computational abilities: he can encrypt and decrypt if he knows the corresponding key, and he can pair and decompose messages.

Based on this model we want to express rules for the adversary's ability to *observe* and *control* oracles. To decide whether a composed term represents an oracle, we need to determine if it contains the secret to be guessed. By overhearing such a term, the adversary *observes* the oracle. Further, to decide *controls*, we start from terms containing the secret, construct new terms in which the secret is replaced by a different value and test if the adversary knows them, and thus *controls* the oracle for the function derived from the term.

For secret containment (the derivation \vdash_{part} in our theory) we define the `containsSec` fact, which is true for all terms containing the secret. For secret replacement (derivation $\vdash_{s \leftarrow \text{gen}(s')}$), we define the `replaceSec` fact which replaces any secret from the `guessableSecrets` set with a *replacement* secret.

With these helper facts defined, the *observes* and *controls* abilities are easily derived. *Observing* an oracle is modeled as `ihears(T).containsSec(T, SList)`, where `SList` is the list of guessable secrets, while *controlling* an oracle is specified as `replaceSec(T, Tnew).iknows(TNew)` (where pairing with `.` means fact conjunction in IF). Explicit *observes* and *controls* predicates are not necessary; for efficiency, the above expressions are directly embedded into the guessing rules.

Guessing multiple secrets. To enable guessing in such scenarios, secrets already guessed must be used in subsequent guesses. However, this cannot be expressed by a simple chaining of the guesses, since adding new knowledge to the intruder cannot be done dynamically in the attack condition. Our simple and effective solution expresses the guessing rule (based on the *observes* and *controls* abilities) as transition of the protocol itself. As a result, any guessed value is added to `iknows`. Being protocol-independent, this rule can be inserted in any protocol specification and enables chaining multiple guesses.

Distinguishing detectable from undetectable on-line attacks. As a first intuition, if guessing takes place after a participant has reached a final state, then guessing goes undetected for that participant. This intuition is wrong, as the same participant may have another instance still running. To distinguish undetectable from detectable on-line guessing attacks, we need to express that all participant instances have successfully completed. We can do this by adding the PIDs of all started instances to a set, adding their termination to the intruder knowledge and checking the match in the attack condition. Alternatively, simply matching the count of started and finished instances suffices.

MS-CHAP v2	
1. $A \rightarrow B : A$	$(a,1) \rightarrow i: a$
2. $B \rightarrow A : N_B$	$i \rightarrow (b,1): a$
3. $A \rightarrow B : N_A, H(k_{AB}, N_A, N_B, A)$	$(b,1) \rightarrow i: Nb(2)$
4. $B \rightarrow A : H(k_{AB}, N_A)$	$i \rightarrow (a,1): Nb(2)$
	$(a,1) \rightarrow i: Na(3).h(kab.Na(3).Nb(2).a)$
	$i \rightarrow (b,1): Na(3).h(kab.Na(3).Nb(2).a)$
	$(b,1) \rightarrow i: h(kab.Na(3))$
	$i \rightarrow (a,1): h(kab.Na(3))$
	$i \rightarrow (i,1): h(kab_rpl.Na(3))$
	$i \rightarrow (i,1): kab.snull$
	$i \rightarrow (i,17): kab$
NTLMv2-Session	
1. $B \rightarrow A : N_B$	$i \rightarrow (i,1): h(kab.Na(3))$
2. $A \rightarrow B : N_A, H(k_{AB}), H'(N_A, N_B)$	$i \rightarrow (i,1): h(kab_rpl.Na(3))$
3. $B \rightarrow A : H(k_{AB}, H'(N_A, N_B)), H'(N_A, N_B)$	$i \rightarrow (i,17): kab$

Fig. 1. MS-CHAP and NTLMv2-Session protocols and OFMC attack trace

MS-CHAP and NTLM. These are two simple, well known protocols from Microsoft, vulnerable but still frequent in practice even today. MS-CHAP is used for remote user authentication and has two versions. NTLM is used with SMB to access remote printers, files etc. and has three versions: NTLMv1, NTLMv2 and NTLMv2-Session. Figure 1 presents MS-CHAP v2 and NTLM v2-Session.

We have augmented the MS-CHAP v2 protocol model with guessing rules. As expected, OFMC found the attack in Figure 1; a similar attack can be traced for NTLM. The intruder acts as man-in-the-middle. Guessing is possible because the intruder hears $h(kab.Na(3))$, and knowing $Na(3)$ can compute $h(kab_rpl.Na(3))$ for arbitrary replacements kab_rpl of kab . By Lemma 2 this means that the intruder *observes* and *controls* the oracle $O_s^f(\cdot)$, where $f = h(s, Na(3))$. The last three trace steps are intruder reasoning; they reflect the fact that additions to intruder knowledge are modeled in the same way as message receipts.

The guessing attacks on MS-CHAP and NTLM are known and simple, but the results serve as basic proof that our approach can automate their detection. The role of an automatic verification tool is to be used on large, complex systems or services that cannot be handled by hand and where such a protocol is only a small foundational component. Thus, the ability of detecting flaws in such a protocol becomes crucial in the discovery of new flaws in the overall system.

The Norwegian ATM. A second test for our implementation was a Norwegian ATM protocol (Fig. 2), known as flawed [11]. A bank issues ATM cards with the user PIN encrypted as $E_{BKey}(PIN)$ with a secret bank key $BKey$. The question is if an adversary, having a stolen card, can guess the user PIN from the encrypted value. One can argue that testing on-line against an ATM with all possible PINs is not feasible since the ATM will lock, e.g., after the first three wrong guesses. However, imagine that the adversary has a card issued by the same bank, with $E_{BKey}(PIN_{Adv})$ on it and that each card owner may change its PIN at an ATM. Now, the adversary can break the PIN on the stolen card by using an on-line attack in which he legally changes its own PIN and then verifies the encrypted value from his card against the one from the stolen card. This is a simplified example, since the PIN is usually not encrypted alone, but with some card-specific information. However, it justifies the concern for on-line attacks.

Card Issuing Stage: 1. $\text{Bank} \rightarrow \text{User} : [\text{DES}_{B\text{Key}}(\text{PIN})]_{16}, \text{PIN}$
PIN Change Procedure: 1. $\text{User} \rightarrow \text{ATM} : [\text{DES}_{B\text{Key}}(\text{PIN}_{\text{old}})]_{16}, \text{PIN}_{\text{old}}, \text{PIN}_{\text{new}}$
 2. $\text{ATM} \rightarrow \text{User} : [\text{DES}_{B\text{Key}}(\text{PIN}_{\text{new}})]_{16}$

First attack trace	Second attack trace
1. $i \rightarrow (i,3) : \text{stolenPIN}_{\text{rpl}_{B\text{Key}, \text{rpl}}}$	1. $i \rightarrow (i,1) : \text{PIN}(1)_{B\text{Key}}.\text{PIN}(1)$
2. $i \rightarrow (i,3) : \text{BKey}.\text{stolenPIN}$	2. $i \rightarrow (i,2) : \text{PIN}(1)_{B\text{Key}}.\text{PIN}(1).\text{stolenPIN}_{B\text{Key}}$
3. $i \rightarrow (i,17) : \text{stolenPIN}$	3. $i \rightarrow (i,2) : \text{stolenPIN}$

Fig. 2. The Norwegian ATM protocol (modified) and OFMC attack traces

The first attack trace produced by OFMC (Fig. 2) was rather unexpected. If an adversary obtains $\text{DES}_{B\text{Key}}(\text{PIN})$, he can compute $\text{DES}_k(m)$ for any key and message, thus he *observes* and *controls* the oracle $O^{\text{DES}(\cdot)(\cdot)}$ and can perform guessing. This is an off-line attack and the trace represents intruder deductions: controlling the oracle with replaced values (1), the intruder deduces both PIN and *BKey* (2), and thus the PIN (3). In practice this is impossible because every PIN would match for some DES key; moreover, only 16 bits of the result are stored, yielding a huge number of potential values for *BKey* and PIN. To test our calculus, we restricted the adversary from trying replacements for *BKey*. Thus, the adversary no longer controls the oracle $O^{\text{DES}(\cdot)(\cdot)}$.

OFMC found the second trace, where the adversary, being issued a legal card (1) uses the PIN change procedure against the ATM in order to *control* the oracle $O^{\text{DES}_{B\text{Key}}(\cdot)}$, where *BKey* is constant. Matching the terms for legal and stolen cards in the intruder knowledge (2) produces the PIN (3). The attack is realistic assuming that the DES encryption is done directly on the PIN.

The Lomas et al. protocol [12] is an interesting case study for guessing attacks. The protocol is illustrated in Figure 3. With respect to our theory it is relevant as it fits the second guessing case (Lemma 3). Lowe [13] found an attack by choosing the constant 0 as timestamp, which allows a replay attack that lets the adversary recover the nonce from two different responses of the server, thus allowing the verification of a correct password. Using OFMC we found a different attack, based again on the weakness of the timestamp. We used a nonce encrypted with *pwdA* as the arbitrary timestamp, in order to avoid Lowe's attack. Quite unexpectedly, OFMC produced the following attack trace: instead of replaying the message from session 1 (Lowe's attack) the intruder lets the protocol run normally between *A* and *B* and from this run he obtains $\{Na1, k \oplus Na2\}_{pwdA}$. Now the intruder initiates a new protocol session, impersonating *A* and sending $\{A, B, Na1', Na2', Ca, \{Na1, k \oplus Na2\}_{pwdA}\}_{pkS}$ in step 1 to the server. Indeed $\{Na1, k \oplus Na2\}_{pwdA}$ besides the length (which is

- | | |
|--|---|
| 1. $A \rightarrow S : \{A, B, Na1, Na2, Ca, \{Ta\}_{pwdA}\}_{pkS}$ | 5. $S \rightarrow B : \{Nb1, k \oplus Nb2\}_{pwdB}$ |
| 2. $S \rightarrow B : A, B$ | 6. $B \rightarrow A : \{Rb\}_k$ |
| 3. $B \rightarrow S : \{B, A, Nb1, Nb2, Cb, \{Tb\}_{pwdB}\}_{pkS}$ | 7. $A \rightarrow B : \{f(Rb), Ra\}_k$ |
| 4. $S \rightarrow A : \{Na1, k \oplus Na2\}_{pwdA}$ | 8. $B \rightarrow A : \{f(Ra)\}_k$ |

Fig. 3. The Lomas et al. protocol

mainly an implementation issue) is just a random value (indistinguishable from a nonce) encrypted with the correct password of A . Further on, the server answers in step 4 with $\{Na1', k \oplus Na2'\}_{pwdA}$, but now $Na1'$ is known to the adversary as he has forged the message in step 1 and thus he can make a correct guess. This attack is not based on a replay, as the message in step 1 was never received by S before. To the best of our knowledge, this attack is new.

4 Conclusions

We have formalized rules for detecting guessing attacks, linking their underlying algebraic properties to the context of symbolic protocol descriptions. Stated as conditional rewrite rules in the description language IF, they can be added to any protocol model, and used with the usual Dolev-Yao intruder deductions. Thus, guessing attacks can be automatically detected without change to the model-checker back-ends. Our implementation automatically distinguishes between detectable and undetectable on-line attacks and can guess multiple secrets.

Using OFMC we have found attacks on several protocols; of these, the attacks on the simplified Norwegian ATM (using the PIN change procedure) and on the Lomas et al. protocol are new to the best of our knowledge. We believe this shows our automation of guessing attack detection to be practically relevant, especially in its support for undetectable on-line attacks. Future work involves integrating this proof-of-concept implementation into a stable release of the AVANTSSAR verification toolset or potentially with a high-level specification language.

Acknowledgments. We thank Sebastian Mödersheim and Roberto Carbone for valuable answers on using the OFMC and SATMC model checkers.

References

1. Abadi, M., Baudet, M., Warinschi, B.: Guessing attacks and the computational soundness of static equivalence. In: Aceto, L., Ingólfssdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 398–412. Springer, Heidelberg (2006)
2. Armando, A., Compagna, L.: SAT-based model-checking for security protocols analysis. International Journal of Information Security 7(1), 3–32 (2008)
3. Basin, D.A., Mödersheim, S., Viganò, L.: OFMC: A symbolic model checker for security protocols. Internat. Journal of Information Security 4(3), 181–208 (2005)
4. Baudet, M.: Deciding security of protocols against off-line guessing attacks. In: 12th ACM Conf. on Computer and Communications Security, pp. 16–25 (2005)
5. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE Computer Security Foundations Workshop, pp. 82–96 (2001)
6. Corin, R., Doumen, J.M., Etalle, S.: Analysing password protocol security against off-line dictionary attacks. In: 2nd W. Secur. Issues Petri Nets, pp. 47–63 (2004)
7. Corin, R., Malladi, S., Alves-Foss, J., Etalle, S.: Guess what? Here is a new tool that finds some new guessing attacks. In: W. Issues Theory Sec., pp. 62–71 (2003)

8. Ding, Y., Horster, P.: Undetectable on-line password guessing attacks. *Operating Systems Review* 29(4), 77–86 (1995)
9. Drielsma, P.H., Mödersheim, S., Viganò, L.: A formalization of off-line guessing for security protocol analysis. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS (LNAI), vol. 3452, pp. 363–379. Springer, Heidelberg (2005)
10. Groza, B., Minea, M.: A calculus to detect guessing attacks. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 59–67. Springer, Heidelberg (2009)
11. Hole, K.J., Moen, V., Klingsheim, A.N., Tande, K.M.: Lessons from the Norwegian ATM system. *IEEE Security and Privacy* 5(6), 25–31 (2007)
12. Lomas, T.M.A., Gong, L., Saltzer, J.H., Needham, R.M.: Reducing risks from poorly chosen keys. In: 12th ACM Symp. on Oper. Sys. Princip., pp. 14–18 (1989)
13. Lowe, G.: Analysing protocols subject to guessing attacks. *Journal of Computer Security* 12(1), 83–98 (2004)

Signatures of Reputation

(Extended Abstract)

John Bethencourt¹, Elaine Shi², and Dawn Song¹

¹ UC Berkeley

{bethenco,dawsong}@cs.berkeley.edu

² PARC

eshi@parc.com

Abstract. Reputation systems have become an increasingly important tool for highlighting quality information and filtering spam within online forums. However, the dependence of a user’s reputation on their history of activities seems to preclude any possibility of anonymity. We show that useful reputation information can, in fact, coexist with strong privacy guarantees. We introduce and formalize a novel cryptographic primitive we call *signatures of reputation* which supports monotonic measures of reputation in a completely anonymous setting. In our system, a user can express trust in others by voting for them, collect votes to build up her own reputation, and attach a proof of her reputation to any data she publishes, all while maintaining the unlinkability of her actions.

1 Introduction

In various forms, reputation has become a ubiquitous tool for improving the quality of online discussions. For example, a user may mark a product review on Amazon or a business review on Yelp as “useful”, and these ratings allow others to more easily identify the best reviews and reviewers. Most web message boards also include a means of providing feedback to help highlight quality content, an early example being Slashdot’s “karma” system.

Unfortunately, in all such systems, a user is linked by their pseudonym to a history of their messages or other activities. In many online communities (e.g., a support group for victims of abuse), users may hope that the use of a pseudonym allows them to remain anonymous. However, recent work has shown that very little prior information about an individual is necessary to match them to their pseudonym [1,2,3]. Building a truly private forum requires abandoning the notion of persistent identities.

We raise the question of whether it is possible to gain all the utility of existing reputation systems while maintaining the unlinkability and anonymity of individual user actions, thus avoiding the histories of activity which threaten privacy. Such a system would enable a number of intriguing applications. For example, we might imagine an anonymous message board in which every post stands alone – not even associated with a pseudonym. Users would rate posts based on whether they are helpful or accurate, collect reputation from other users’ ratings, and annotate or sign new posts with the collected reputation.

Other users could then judge new posts based on the author’s reputation while remaining unable to identify the earlier posts from which it was derived. Such a forum would allow effective filtering of spam and highlighting of quality information while providing an unprecedented level of user privacy.

Our approach. To build toward this goal, we propose *signatures of reputation* as a new cryptographic framework enabling the counter-intuitive combination of reputation and anonymity. In a conventional signature scheme, a signature is associated with a public key and convinces the verifier that the signer knows the corresponding private key. Based on the public key, a verifier could then retrieve the reputation of the signer. Through signatures of reputation, we aim to eliminate the middle step of identifying the signer: instead, verification of the signature directly reveals the signer’s reputation. With such a tool, a user may apply their reputation to *any* data that they wish to publish online, without risking their privacy. By formally defining this setting, we hope to spur further research into techniques for its realization.

As a first step, we introduce a construction for signatures of reputation that supports *monotonic* aggregation of reputation. That is, we assume that additional feedback cannot decrease a user’s reputation. While a user’s misbehavior cannot damage reputation they have already accumulated, such a system is sufficient to prevent more casual attackers who, for example, wish to post spam without taking the time to obtain reputation first. Although some existing reputation systems are monotonic (e.g., Google’s PageRank algorithm), one would ultimately hope to support non-monotonic reputation as well. We leave this as a primary open problem for future work.

In our construction, the reputation feedback takes the form of cryptographic “votes” that users construct and send to one another, and a user’s reputation is simply the number of votes they have collected from distinct users. Each user stores the votes they have collected, and to anonymously sign a message with their reputation, the user constructs a non-interactive zero-knowledge (NIZK) proof of knowledge which demonstrates possession of some number of votes.

The ability of a reputation system to limit the influence of any single user is crucial in enabling applications to control abuse. To this end, our construction ensures that each user can cast at most one valid vote for another user (or up to k for any fixed $k \geq 1$). Enforcing this property is a major technical problem due to the tension with the desired unlinkability properties. In Section 4, we give a high-level overview of the techniques our construction uses to address this and other technical challenges. For details of the construction, proofs, and additional material, we refer the reader to the full version of this paper [4].

Related work. While we are not aware of any work directly comparable to our proposed signatures of reputation, others have explored the conflict between reputation and unlinkability [5,6,7]. E-cash schemes also attempt to maintain the unlinkability of individual user interactions, and in several cases [8,9,10] they have been applied for reputation or incentive purposes. The work of Androulaki et al. [10] is particularly close to ours in its aims. However, this and all other e-cash based approaches are incapable of supporting the type of abuse resistance provided by our scheme because they allow a single user to give multiple coins

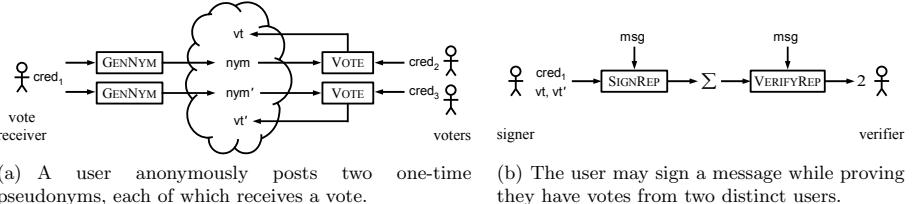


Fig. 1. One-time pseudonyms, votes, and signatures of reputation

to another, inflating their reputation. In our scheme, it is possible to prove that a collection of votes came from *distinct* users. This ability to prove distinctness while maintaining the mutual anonymity of both voters and vote receivers is the key technical achievement of our construction.

Anonymous credentials schemes [11,12,13] may also be considered an effort toward the goal of “trust without identity”. There are two key distinctions, however. First, anonymous credentials are concerned with the setting of access control based on trust derived from explicit authorities, whereas this work aims to support trust derived from a very different source: the aggregate opinions of other users. Second, like e-cash based approaches, existing anonymous credential schemes lack a mechanism for proving that votes or credentials come from distinct users while simultaneously hiding the identities of those users.

Finally, our setting superficially resembles that of e-voting (e.g., [14]), in that it allows the casting of votes while maintaining privacy properties. However, e-voting schemes are designed for an election scenario in which the candidates have no need to receive votes and prove possession of votes anonymously, among other differences, and cannot be used to achieve the properties we require.

2 Defining Signatures of Reputation

We now introduce our formulation of signatures of reputation within the vote counting scenario, then define the algorithms which constitute such a scheme.

Overview. In the system illustrated in Fig. 1, we refer to each user as a *vote receiver*, *voter*, *signer*, or *verifier* depending on their role in the specific algorithm being discussed. To ensure *receiver anonymity*, a vote receiver invokes the **GENNYM** algorithm to compute a “one-time pseudonym” called a *nym*, which they attach to some content that they publish and wish to receive credit for. A voter can then use the **VOTE** algorithm on a *nym* to produce a vote which hides their identity, even from the recipient (referred to as *voter anonymity*). The voter posts the vote online where the recipient can later retrieve it. After collecting some votes, a signer runs the **SIGNREP** algorithm on a given message to construct a signature of reputation, which must not reveal the signer’s identity (*signer anonymity*). We also ensure that a malicious signer cannot inflate its reputation (*reputation soundness*).

To participate in the system, each user must contact a *registration authority* (RA) which generates the user's private credentials, just as the key generating server does within IBE schemes. Although our construction requires trust in the RA for both privacy and reputation soundness, it need only be trusted when registering users and may thereafter go offline. As with typical IBE schemes, it is also possible to reduce the trust necessary in the RA by distributing it amongst multiple parties [15]. Devising a scheme which maintains privacy in the presence of a malicious RA is an interesting problem for future work. On the other hand, relying on the honesty of the RA for reputation soundness seems inevitable, since a malicious RA could always register additional phony users (i.e., Sybil identities) to arbitrarily create votes and inflate reputations.

At this point, one might raise the concern that, if each user has received a unique number of votes, the reputation value itself is identifying. Clearly, there is an inherent tradeoff between the precision of a measure of reputation and the anonymity of a user with any specific value, as pointed out by Steinbrecher [6]. The solution is to use a sufficiently coarse-grained reputation. When producing a signature in our construction, a user may prove any desired *lower bound* on their reputation instead of revealing the actual value. In this way, our construction allow users to implement their own policies for the precision of their reputations. For example, one policy would be to always round down to a power of two.

Algorithms. We now list and define the algorithms that constitute a scheme for signatures of reputation. All but VERIFYREP may be randomized.

SETUP(1^λ) \rightarrow (**params**, **authkey**): The SETUP algorithm is run once on security parameter 1^λ to establish the public parameters of the system **params** and a key **authkey** for the registration authority.

GENCRED(**params**, **authkey**) \rightarrow **cred**: To register a user, the registration authority runs GENCRED and returns the user's credential **cred**.

GENNYM(**params**, **cred**) \rightarrow **nym**: The GENNYM algorithm produces a one-time pseudonym **nym** from a user's credential.

VOTE(**params**, **cred**, **nym**) \rightarrow **vt** or \perp : Given the credentials **cred** of some user and a one-time pseudonym **nym**, VOTE outputs a vote from that user for the owner of **nym**, or \perp in case of failure (e.g., if **nym** is invalid).

SIGNREP(**params**, **cred**, V , **msg**) \rightarrow Σ or \perp : Given the credentials **cred** of some user, the SIGNREP algorithm constructs a signature of reputation Σ on a message **msg** using a collection of c votes $V = \{vt_1, vt_2, \dots, vt_c\}$ for that user. The signature corresponds to a reputation $c' \leq c$, where c' is the number of distinct users who generated votes in V . The SIGNREP algorithm outputs \perp on failure, specifically, when V contains an invalid vote or one whose recipient is not the owner of **cred**.

VERIFYREP(**params**, **msg**, Σ) \rightarrow c or \perp : The VERIFYREP algorithm checks a purported signature of reputation on **msg** and outputs the corresponding reputation c , or \perp if the signature is invalid.

The most basic property required of the above algorithms is *correctness*; we omit this definition for brevity. In the following section, we explore the other desired properties.

3 Privacy and Security Properties

The full version of this paper provides rigorous definitions for the four privacy and security properties [4]; here, we describe them at an intuitive level and discuss some of the subtleties in defining them appropriately.

Signer anonymity. First, we would like to ensure that a user may produce signatures of reputation anonymously. Furthermore, it should be impossible to determine whether two different signatures were produced by the same user. This may be defined by the following game. The challenger begins by generating the public parameters and a list of user credentials $\text{cred}_1, \dots, \text{cred}_n$. An adversary \mathcal{A} is given access to all the credentials and may use them to generate pseudonyms and votes before eventually printing a message msg , $1 \leq i_0, i_1 \leq n$, and two sets of votes V_0, V_1 . The challenger flips a coin $b \in \{0, 1\}$ and returns $\Sigma_b = \text{SIGNREP}(\text{params}, \text{cred}_{i_b}, V_b, \text{msg})$ to \mathcal{A} , which then prints a guess b' . We say that \mathcal{A} has won the game if $b = b'$ and $\text{VERIFYREP}(\text{params}, \text{msg}, \Sigma_0) = \text{VERIFYREP}(\text{params}, \text{msg}, \Sigma_1)$. That is, the value of b should affect neither the reputation values of the resulting signatures nor their validity. If the advantage (that is, the probability of winning the game minus one-half) of every PPT \mathcal{A} is negligible in the security parameter, we say that the scheme is *signer anonymous*.

Receiver anonymity. Complementing the ability to produce a signature of reputation anonymously is the ability to receive the necessary votes anonymously. In this case, we require that a pseudonym generated by the GENNYM algorithm reveal nothing about its owner in the absence of that user's credential. An adversary \mathcal{A} playing the corresponding game will select two users $1 \leq i_0, i_1 \leq n$ and must guess which produced the challenge $\text{nym}^* = \text{GENNYM}(\text{params}, \text{cred}_{i_b})$. Since we allow users to identify their own pseudonyms, we cannot provide all the credentials to \mathcal{A} in this case. Instead, we provide \mathcal{A} with access to an oracle which will reveal individual credentials on demand (a “corrupt” query) or use them to produce pseudonyms, votes, and signatures as requested. Then, to win the game, we require that \mathcal{A} not corrupt either i_0 or i_1 . We also require that \mathcal{A} not request a signature from i_0 or i_1 using a vote that was cast for nym^* , since the reply would immediately reveal b (the signer is i_b iff the reply is not \perp). If the advantage of every PPT \mathcal{A} in this game is negligible in the security parameter, the scheme is *receiver anonymous*.

Astute readers may note that we have not properly defined what it means for a vote to have been “cast for nym^* ”, since we have no information about how the adversary may have constructed it. To resolve this definitional issue, in the full version of this paper, we define opening algorithms which reveal the creator of a pseudonym and the voter and recipient of a given vote. To operate, they require a special opening key which may be generated during setup, just as in group signature schemes. However, while this tracing is an explicit feature of group signatures, here we use it only to establish a “ground truth” for definitional purposes. In an actual implementation, the opening key would not be generated.

Voter anonymity. We wish to define the voter anonymity property to encompass the strongest form of unlinkability compatible with the general semantics of the scheme, as we did in the case of receiver anonymity. Doing so is more subtle in

this case, however, due to the necessity of detecting duplicate votes. Because we require a SIGNREP algorithm to demonstrate the number of votes from *distinct* users, such an algorithm can be used by a vote receiver to determine whether two votes cast for any of their pseudonyms were produced by the same voter (duplicates). That is, the receiver can try to use the two votes to produce a signature and then check the reputation of the result with VERIFYREP.

In defining voter anonymity, we allow precisely this type of duplicate detection, but nothing more. While initially this may seem like an “exception” to the unlinkability of votes, in actuality, it is not only inevitable,¹ but also unlikely to be a practical concern. Although a vote receiver must be able to detect duplicate votes, we can still avoid the voting histories we originally set out to eliminate. In particular, our definition ensures that in the following cases it is not possible to determine whether two votes were cast by the same user (i.e., to link the votes):

1. A user cannot link a vote for one of their pseudonyms with a vote for a pseudonym of another user, nor can they link two votes for distinct pseudonyms of another user (or two different users).
2. A *colluding group* of users cannot link votes between their pseudonyms, provided the pseudonyms correspond to different credentials. Furthermore, they are not able to link the *numbers* of duplicates they have observed. For example, if a user determines that they have received two votes from one user and three votes from another, they will have no way of matching these totals up with those of another colluding user.

In the corresponding game, \mathcal{A} selects $1 \leq i_0, i_1 \leq n$ and nym and is given $\text{vt}^* = \text{VOTE}(\text{params}, \text{cred}_{i_b}, \text{nym})$ as a challenge. As before, they are given access to the oracle and must make a guess b' . In this case, we require that if \mathcal{A} requests through the oracle that the user corresponding to nym produce a signature using vt^* , then votes from both i_0 and i_1 must be included. Otherwise, the number of distinct votes in the resulting signature would directly reveal b . Additionally, we disqualify \mathcal{A} if they both corrupt the user corresponding to nym and request a vote on nym from either i_0 or i_1 . This is necessary because the status of such a vote as a duplicate of vt^* (or lack thereof) would reveal b . If every PPT \mathcal{A} has negligible advantage in this game, the scheme is *voter anonymous*.

Reputation soundness. To define the soundness of a scheme for signatures of reputation, we use a computational game in which an adversary \mathcal{A} must forge a signature of reputation Σ on some message msg . We disqualify \mathcal{A} if Σ was the reply to one of its oracle queries, and we require that Σ have reputation strictly greater than what it could if the adversary had used the scheme normally. The value of the best such legitimately obtainable reputation will depend on several things: the number of users the adversary has corrupted (since the adversary may use their credentials to produce votes), the number of votes received from honest users via oracle queries, and how those votes were distributed amongst

¹ Allowing proofs of vote distinctness while eliminating the ability to identify duplicates could only be possible if the notion of discrete votes is abandoned. This approach would require *all* votes in the system to be aggregated into a indivisible block before they can be used to produce signatures, a vastly impractical solution.

the corrupted users. More precisely, let ℓ_1 be the *number of corrupted users* and ℓ_2 be the *greatest number of distinct honest users that voted for a single corrupt user*. Then we require that $\text{VERIFYREP}(\text{params}, \text{msg}, \Sigma) > \ell_1 + \ell_2$ for the adversary to succeed. If, for every PPT \mathcal{A} , the probability of winning this game is negligible in the security parameter, then the scheme is *sound*.

In some applications, a weaker version of soundness may suffice and may be desirable for greater efficiency. One natural way to relax the definition is to specify an additional security parameter $\varepsilon \in (0, 1)$ as a multiplicative bound on the severity of cheating we wish to prevent. Specifically, we say that a scheme is ε -*sound* if it satisfies the above definition, but using the requirement that $(1 - \varepsilon) \cdot \text{VERIFYREP}(\text{params}, \text{msg}, \Sigma) > \ell_1 + \ell_2$.

4 Highlights of Our Construction

Our scheme for signatures of reputation can produce sound signatures of reputation c of size $O(c)$ or ε -sound signatures of size $O(\frac{1}{\varepsilon} \log c)$. In the full version of this paper, we detail the construction and prove that it satisfies all of the properties discussed in the previous section [4]. In this section, we describe some of the scheme’s technical features and underlying ideas.

Assumptions. Our constructions rely on a bilinear map (symmetric or asymmetric) between prime order groups. Its privacy and security properties are based on the relatively standard DLINear and SDH assumptions, BB-HSDH and BB-CDH [11], and a new constant-size, non-interactive, computational assumption called SCDH, which we prove hard in generic groups. Additionally, the ε -sound variant of our scheme requires the random oracle model.

Nested NIZKs. Throughout our construction, we make extensive use of the Groth-Sahai scheme for non-interactive zero-knowledge (NIZK) proofs [16], which can be used to efficiently demonstrate possession of signatures, ciphertexts, and their relationships while maintaining unlinkability properties. One unique (to our knowledge) feature of our construction is the use of *nested NIZKs*, that is, NIZKs which prove knowledge of other NIZKs and demonstrate that they satisfy the verification equations. This situation arises because a user’s credentials contain a signature from the registration authority, and a user includes a NIZK proof of the validity of this signature when they cast a vote. When a signer later uses the vote, they include this NIZK within a further NIZK to demonstrate the validity of the votes while maintaining signer anonymity.

Proving distinctness. We give signers the ability to prove the distinctness of their votes through the following mechanism. Each user credential contains (among other components) a “voter key” v and a “receiver key” r . A valid vote must contain a certain deterministic, injective function of these keys: $f(v, r)$. Thus, duplicate votes can be detected when $f(v_1, r) = f(v_2, r)$. To receive votes anonymously, a user includes in each `nym` an encryption of their receiver key $E(r)$ under their own public key. Using a homomorphism, the voter can use this ciphertext to compute $E(f(v, r))$ and place it within the vote; later, the receiver will decrypt this to obtain $f(v, r)$. To maintain signer anonymity when using a series of votes $U_1 = f(v_1, r), U_2 = f(v_2, r), \dots$ to sign a message, the signer blinds the

votes with a (single) exponent to produce a list U_1^s, U_2^s, \dots , which is included in the signature of reputation along with proof of knowledge of the exponent. Note that U_1^s, U_2^s, \dots will be distinct if the original values were.

Short signatures. To reduce the size of the signatures, we employ a sampling technique. Specifically, we can achieve ε -soundness while only including a random subset of the votes of size $O(\frac{1}{\varepsilon})$, *independent* of the original number of votes. To ensure the sample is random, we require the signer to first commit to the entire list of votes, then use the commitment as a challenge specifying which must be included. To efficiently demonstrate that the correct votes were included, we compute the commitment using a Merkle hash tree and include the corresponding off-path hashes with each vote, resulting in a final signature of size $O(\frac{1}{\varepsilon} \log c)$.

References

1. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: IEEE Symposium on Security and Privacy (2008)
2. Backstrom, L., Dwork, C., Kleinberg, J.M.: Wherefore art thou r3579x? In: International World Wide Web Conference (2007)
3. Arrington, M.: AOL proudly releases massive amounts of user search data. In: TechCrunch News (August 2006)
4. Bethencourt, J., Shi, E., Song, D.: Signatures of reputation: Towards trust without identity, <http://www.cs.berkeley.edu/~bethenco/sigrep-full.pdf>
5. Steinbrecher, S.: Enhancing multilateral security in and by reputation systems. In: FIDIS/IFIP Internet Security and Privacy Summer School (September 2008)
6. Pingel, F., Steinbrecher, S.: Multilateral secure cross-community reputation systems for internet communities. In: Furnell, S.M., Katsikas, S.K., Liou, A. (eds.) TrustBus 2008. LNCS, vol. 5185, pp. 69–78. Springer, Heidelberg (2008)
7. Steinbrecher, S.: Design options for privacy-respecting reputation systems within centralised internet communities. In: Intl. Information Sec. Conf, SEC (2006)
8. Belenkiy, M., Chase, M., Erway, C., Jannotti, J., Kupcu, A., Lysyanskaya, A., Rachlin, E.: Making p2p accountable without losing privacy. In: WPES (2007)
9. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Balancing accountability and privacy using e-cash. In: Security and Cryptography for Networks, SCN (2006)
10. Androulaki, E., Choi, S.G., Bellovin, S.M., Malkin, T.: Reputation systems for anonymous networks. In: Privacy Enhancing Technologies (2008)
11. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
12. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: PKC (2009)
13. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Non-interactive anonymous credentials. In: TCC (2008)
14. 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)
15. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 295. Springer, Heidelberg (1999)
16. 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)

Intention-Disguised Algorithmic Trading

(Short Paper)

William Yuen¹, Paul Syverson², Zhenming Liu¹, and Christopher Thorpe¹

¹ Harvard University

{yuen,zliu,cat}@seas.harvard.edu

² Naval Research Laboratory

syverson@itd.nrl.navy.mil

1 Introduction

Large market participants (LMPs) must often execute trades while keeping their intentions secret. Sometimes secrecy is required before trades are completed to prevent other traders from anticipating (and exploiting) the price impact of their trades. This is known as “front-running”. In other cases, LMPs with proprietary trading strategies wish to keep their positions secret even after trading because their strategies and positions contain valuable information. LMPs include hedge funds, mutual funds, and other specialized market players.

However order information is leaked, or why it is sought, front-runners who exploit the LMP’s order information extract value from markets at the expense of the LMP. Thus, hedge funds and other firms take great pains to hide their intentions, even generating “noise” trades to hide their intended positions from other traders [2]. We present trading schemes that disguise an LMP’s intentions and positions from *any* other entity, including the brokers that the LMP interacts with.

Various studies [13,12] have shown abnormal price behavior and significant negative price impact from information leakage prior to a block trade execution. Thorpe and Parkes [16,17] discuss cryptographic and security research on exchanges and how information can be exploited in financial markets. But, existing research generally proposes new infrastructures or protocols, for which adoption is notoriously difficult. We take a simpler approach. Our specific contributions are: (1) to propose a general model underlying the design of trading strategies that leak no information, (2) to study major scenarios in the market and design associated algorithms *that require no changes to the existing trading infrastructure*, and (3) to prove those algorithms leak no information in those scenarios. These algorithms can serve as building blocks for more challenging real-world scenarios beyond our present scope. Though our approach is algorithmic, we are not concerned with volume-weighted algorithmic trading. See [4], [5] and [11] for a review of the literature and for insights into the study of automated trading.

We next discuss existing trading infrastructure, define three types of adversaries, and present ways they can extract information from orders placed by the LMP. In Section 3, we describe the model for information leakage and address the needed properties for an efficient trading strategy. Section 4 introduces different trading strategies that disguise the intention and holdings of the LMP from

exploiters. We evaluate their defensive performance against each of the three types of “exploiters”. Given the available space, our presentation gives only the basic ideas. Detailed mathematical explanations of various information leaks, as well as theorem statements and proofs, can be found in the full paper [18].

2 Preliminaries

Existing trading infrastructure and exploiters

Brokers include brokers, dealers, and broker-dealers. *Shares* are units of any security, including equities, bonds, currencies, or derivatives. One can long or short any of the shortable securities represented through brokers. Each transaction is for a nonzero integer number of shares; although LMPs typically trade in increments of at least 100 shares. When a trade is executed, the symbol and quantity is publicly reported by the exchange. Typically, only the broker involved in the trade knows the identity of the LMP and whether the LMP was the buyer or seller. From weakest to strongest, the categories of exploiters are:

1. *Curious Observers* are able to see trades printed as they are executed and/or the prices and sizes of orders (requested trades) as they are quoted. With sufficient intelligence and experience, curious observers may be able to guess the identity and intention of the LMP.
2. *Individual Curious Brokers* are able to see trade orders by the LMP before they are executed. A corrupt or careless broker can leak the LMP’s intentions for exploitation by broker insiders or external agents. Using multiple brokers, the LMP can limit the information a single curious broker can extract.
3. *Colluding Curious Brokers* are able to see trade orders by the LMP and can share their information with each other. If all brokers used by the LMP are curious and collude, any benefits resulting from splitting trades across different brokers would be lost. However, all brokers used by the LMP must collude in order to yield complete knowledge.

In all our strategies the LMP places a set of orders for an asset d at one or more brokers in order to yield a net purchase or sale of d . Using minimal resources, we want to prevent reasonably capable exploiters from guessing the net order. Strategies must also stay completely effective even when exploiters are aware that the LMP is using them. We focus primarily on the following scenarios:

- *Multiple brokers with one trader* ($nB1T$): There is only one trader represented in the market, the LMP. This trader can interact with many brokers.
- *Single broker with multiple traders* ($1BmT$): Only one broker handles trades, for the LMP and possibly for other market participants.
- *Multiple brokers with multiple traders* ($nBmT$): Multiple traders and multiple brokers can trade simultaneously, the most general scenario.

Two motivating trading strategies and why they leak information

To hide the net order we first consider a simple approach where the LMP uses two brokers and places an order with each one so that neither broker individually

Table 1. An example of disguising true trading intentions using two brokers

Stock	Shares through Broker A	Shares through Broker B	Net LMP volume traded
ATK	+500k	-500k	0
SXL	-300k	+400k	+100k

learns about the net order. This simple strategy still allows brokers to extract some knowledge. For example in Table 1, Broker *A* observes that a large ATK trade has gone through Broker *B* when Broker *B* prints the block trade after execution. Broker *A* is not sure whether the LMP is involved in the trade with Broker *B*, or the sign of the LMP’s trade through Broker *B*. But, *A* knows that his large client has traded one of three net positions: $500k$, $500k + 500k = 1M$, or $500k - 500k = 0$. Similarly, broker *A* knows his client’s net trade for SXL is $100k$, $-700k$, or $-300k$ shares. Because the number of possible cases is low, Broker *A* can analyze each scenario and deduce the best exploitation strategy. For example, a block trade price that is closer to the bid than to the offer is more likely to be seller-initiated [10].

Another simple strategy is to use a single broker but multiple registered traders. An LMP might create several registered trading agents that are not known to be associated with the LMP but trade on its behalf. Thus we could produce the exact same structure as Table 1 except that now instead of a single trader using Broker *A* and Broker *B*, we have Trader *A* and Trader *B*, both responsible for the same book of the LMP, trading through a single broker. The broker cannot tell from what he sees if he is dealing with one LMP shopping two blocks or two LMPs. This defends against collusion, unlike the two-broker system. But, if the broker links the two pseudonymous traders together, then he will know everything about their intentions going forward. We can combine the two solutions so that each pseudonymous trader is splitting orders across multiple brokers. This gains both the advantages and the overhead of both approaches.

3 Defining “Information Leak”

A rigorous definition of “information leak” is needed to understand both the potential threats from exploitation for the LMP and the desired properties of the trading strategies we are seeking. Here we provide just a sketch of such definitions and refer the reader to the full paper [18]. We propose in this section three types of information leak (or rather its absence) so as to formalize the notion: *zero information leak*, ϵ -*information leak*, and *full space strategy*.

Our inspiration is Goldwasser et al.’s [9] notion *zero knowledge*. Roughly, transmitting a piece of information is zero-knowledge if the universe of computations the recipient can perform does not change after receiving the information.

Definition 1. (Efficient algorithms for zero information leak) *Let $(\Omega, \mathcal{F}, \Pr)$ be a probability space that represents all possible intended positions of an LMP and the corresponding a priori distribution over these positions. Let ω be a random*

sample from Ω . A trading algorithm \mathcal{A} is said to be perfect-zero-knowledge with respect to exploiters if the following two conditions hold:

- \mathcal{A} can generate an execution plan in polynomial time (wrt a reasonable representation of Ω) that ends with the LMP holding exactly ω shares.
- The exploiters are able to generate the distribution on the random variable M on their own without seeing the signal ω .

A natural relaxation of zero information leak is to allow ϵ information leak. The definition is essentially the same as this except that exploiters can generate a random variable with a statistical difference¹ from M of at most ϵ . One may think of the difference between perfect zero knowledge and statistical zero knowledge [7] to understand the motivation for this relaxation in security definition.

Finally, we propose another way to ensure sufficient noise that an adversary is unable to eliminate any possible values from Ω . Specifically we require that $\Pr[\omega | M = p] > 0$ for all q and all ω such that $\Pr[\omega] > 0$.

Definition 2. (Efficient algorithms for full space strategy) Let $(\Omega, \mathcal{F}, \Pr)$ be a probability space that represents all possible intended positions of an LMP and the corresponding prior over these positions. Wolog, assume that $\Pr[\omega] > 0$ for any ω . Let ω be a random sample from Ω . A trading algorithm \mathcal{A} is said to give a full space strategy with respect to exploiters if the following two conditions hold:

- \mathcal{A} can generate an execution plan in polynomial time (w.r.t. a reasonable representation of Ω) that ends with the LMP holding exactly ω shares.
- For any message M observed by the exploiters, $\Pr[\omega | M] > 0$ for any $\omega \in \Omega$.

Although there are more refined notions of knowledge, e.g., that quantify the exact number of bits leaked by a system [8], it is unclear how the amount of leaked information relates to the financial cost of the information. A single leaked bit information can have great value (the sign of an order issued by an insider), but other times even a large information leak may be harmless.

4 Trading Strategies

In this section, we design and analyze trading strategies to counter various adversaries in various markets, and in progressively more challenging scenarios.

Multiple brokers with one trader (nB1T)

In order to defend against the three types of exploiters mentioned, we first build our strategies using a single trader and n orders placed with n different brokers. We call this the *nB1T platform*. We start with nB1T strategies for the LMP against *curious observers* (the weakest). The following sign flipping game is closely related to a trading strategy that leaks no information:

¹ The statistical difference between two discrete random variables X and Y is defined as $\sum_i |\Pr[X = i] - \Pr[Y = i]|$

Definition 3. (Sign Flipping Game) Given an interval $[-q, q]$, find a set of numbers $T = \{t_1, t_2, \dots, t_n\}$ such that $\sum_i t_i = q$ and

- (1) for any integer $x \in [-q, q] \cap \mathbb{Z}$ there exists a set of numbers $a_1, a_2, \dots, a_n \in \{-1, 1\}$, $t_i \in \mathbb{Z}$ such that $x = a_1 \cdot t_1 + a_2 \cdot t_2 + \dots + a_n \cdot t_n$,
- (2) The number n is a function of q . The value of n should be as small as possible.

Intuitively, for our nB1T strategy, n in the sign flipping game is the number of brokers the LMP interacts with, and $\Omega = [-q, q]$ is the range of net position the LMP wants to hold. By buying or selling volume t_i with broker i , he can construct every possible desired net trading volume, x , bounded between $-q$ and q . Unsigned traded volumes $T_L = \{|a_i t_i|\}$ are printed among other traded volumes W_0 that do not involve the LMP. Observer identification of T_L from $T_L \cup W_0$ depends on market liquidity and other factors. An LMP is always able to set a larger q at the cost of higher transaction costs. When the security parameter q is fixed, a natural goal is to minimize the number of brokers used.

Now, suppose the LMP wishes to buy $x \in [-q, q]$ shares (negative x notated as selling) of a product. She would then be able to execute a sequence of orders t_1, t_2, \dots, t_n to each of the brokers such that $x = t_1 + t_2 + \dots + t_n$.

From an observer's point of view, he only sees the sequence $|t_1|, |t_2|, \dots, |t_n|$. If he does not have information of the LMP's intention a priori, the observer can only attempt to extract knowledge by going through all combinations of the signs for all t_i . Therefore, the LMP's strategy should make the following set as large as possible: $S = \{a_1|t_1| + a_2|t_2| + \dots + a_n|t_n| : a_1, \dots, a_n \in \{-1, 1\}\}$. A necessary requirement for a zero-information-leak trading strategy is that $[-q, q] \subseteq S$. Our first goal is to construct $T = \{t_1, t_2, \dots, t_n\}$ with minimum possible n such that S fully covers $[-q, q]$. We can find a T with $|T| = \lceil \log_2 q \rceil + 2$ that satisfies the first requirement of the sign flipping game. In fact this is nearly optimal in that any set T that satisfies the first requirement of the sign flipping game will have $|T| \geq \lceil \log_2 q \rceil + 1$. Further, there exist on the nB1T platform both efficient strategies that leak zero information and full space strategies against curious individual brokers. Proofs of these and related results are in the full paper [18].

The above strategies no longer work against *curious individual brokers who do not collude*. For example, in our analysis [18] of efficient strategies for the sign flipping game, curious broker b_n , knowing q and seeing the sign a_n of an order of size $q/2$, would know that the LMP is intending to buy from the range $[-q, 0]$ if $a_n = -1$ or $[1, q]$ if $a_n = 1$. If instead we are less efficient, splitting trades across more brokers, or less complete, making some of the intermediate values unreachable, then we can prevent any one broker from knowing this much about the LMP's position. We will revisit this observation below.

There are also efficient ϵ -information-leak strategies for the curious broker market. When $1/\epsilon$ is a constant or a polynomial in n , the strategy has a $\Omega(\text{poly}(n))$ expansion. See the full paper for rigorous statements and details.

Countering collusion

The above nB1T strategic platform does not yield strong defense against curious colluding brokers: they can share knowledge with each other, including the

identity of the LMP and the sets a_i and t_i . If colluders know the total number of brokers used and can find all of them, the value x can be trivially extracted.

Even if n is not known or not all n brokers collude, certain possible values for x can be eliminated: Suppose, for example, the LMP uses two sets of brokers $R = \{b_1, b_2, \dots, b_n\}$ and $R' = \{b'_1, b'_2, \dots, b'_n\}$, and that $R \cap R' = \emptyset$. Let $B = R \cup R'$. Suppose brokers $B_c \subset B$ collude and share the information $T_c \subset T$ and $A_c \subset A$ with each other, and let J be the set of indices corresponding to colluding brokers. With enough colluders they can learn significant information. For example, if $\sum_{j \in J} a_j t_j > \sum_{i \notin J} |a_i t_i|$, colluding brokers would know that $1 \leq x \leq q$.

To maximize the collusion resistance for a given n , it is clearly optimal to split q uniformly across all n brokers. In other words, every broker is used to trade q/n shares, either buying or selling. (Let some brokers be allowed to receive no order when n is odd to hide a zero position.) This of course leaks n (easily countered by randomization). Also, note that, even if n is known, the colluding brokers B_c can never learn more than their proportion of the LMP's position.

Single broker with multiple traders (1BmT)

To defend against broker collusion, we now examine utilizing m registered trading agents (hereafter referred to as traders) by the LMP to create the desired net position. The mathematics behind this 1BmT platform is very similar to the nB1T strategy: Simply substitute m traders placing orders at one broker in place of one trader at n brokers (where $m = n$). The same theorems hold for 1BmT as for nB1T. In practice, the additional redundant positions held by the traders add ongoing carrying and transaction costs. Also, changing brokers, especially in a developed market, is generally easier than changing registered traders.

We next consider strategies against the *curious individual broker*, assuming he is unable to identify the traders associated with the LMP. Suppose the LMP places a set of orders $\{a_k t_k\}$ at the broker via m different traders. Let $W_0 = \{w_1, w_2, \dots, w_z\}$ be the normal market interest seen by the broker; i.e., the set of orders the broker receives from clients not affiliated with the LMP. The broker thus sees total market interest $W_t = \{a_k t_k\} \cup W_0$. In a very liquid market, $\exists w_i \in W_0 \ni |w_i| = |a_k t_k|$ for $k = 1, \dots, m$. In this case, the broker cannot identify any $a_k t_k$ from W_t , and the 1BmT platform does not leak information to him. This is not so when liquidity is low and the broker knows the LMP is employing 1BmT, however. For example, if there is no corresponding surge in activity in the overall market or at other brokers, he can infer that all market interests may originate from the LMP. Furthermore, if $\nexists w_i \in W_0 \ni |w_i| = |a_k t_k|$ for some k , $a_k t_k$ can be identified as originating from the LMP. Thus, elements in the set S can be eliminated, similar to the nB1T platform under collusion. These potential information leaks on the 1BmT platform in an illiquid market motivate our next strategy platform.

Multiple brokers with multiple traders (nBmT)

We can extend the above strategies by using n brokers (with index j) and m traders (with index i), with the security parameter q remaining the same. In general form, the LMP uses the set of traders $\{d_1, d_2, \dots, d_m\}$, each of the trader

d_i places orders with a subset of brokers $\{b_{i1}, b_{i2}, \dots, b_{in}\}$. In total, a maximum of $n \cdot m$ orders are placed with a maximum of $n \cdot m$ unique brokers. In practice, some of the b_{ij} 's are the same broker. One possibility is to split the net order x that the LMP wishes to place into m different orders, $\{a_1t_1, a_2t_2, \dots, a_mt_m\}$, for m traders as in the sign flipping game in Definition 3. Each trader d_i can then place its individual single order a_it_i , with broker b_{i1} . In this case, the total number of orders placed is $m = \lceil \log_2(q) \rceil$.

Curious observers cannot see the identities of the traders. Thus, nBmT would look the same as nB1T to curious observers. So, as under nB1T, the external observers cannot extract the trade order made by each trader, thus cannot extract any knowledge about the LMP. Another variant (nBmT2) of this strategy is to divide x into m sets of orders $\{a_1t_1, a_2t_2, \dots, a_mt_m\}$ for m traders according to the sign flipping game. This is detailed in our full paper [18].

We now study the performance of nBmT against *curious individual brokers*. Each trader d_i places its order a_it_i at a different broker. Let W_i be the set of orders each broker b_{i1} receives from his clients not affiliated with the LMP, or normal market activity. Broker b_{i1} , sees total interest $W_{ti} = a_it_i \cup W_i$, and he cannot identify a_it_i from W_i since he does not know that d_i is affiliated with the LMP. This case is different from 1BmT because the market activities W_v at other brokers $b_{v1}, v \neq i$, are also increasing due to the activity of the LMP in nBmT. Thus, even in a low liquidity environment, broker b_{i1} cannot determine whether the increase in $|W_{ti}|$ is due to the activity of the LMP (the presence of a_it_i), or due to increased general market volume (an increase in $|W_i|$).

Collusion does not benefit brokers if traders $\{d_i\}$ are not revealed to be affiliated with the LMP. Colluding brokers do not know which orders are affiliated with the LMP and therefore would act at worst as a single broker in the 1BmT scenario. Thus, the nBmT strategy can guard against total broker collusion.

5 Conclusions and Future Work

We have examined the problem of placing orders while hiding intention. We presented models of information leakage, and based on these models, we derived three classes of strategies against curious observers, individual curious brokers, and colluding curious brokers.

Though not our current focus, we believe transaction costs of these strategies can sometimes be reasonable, such as when the notional share price is high and/or the bid-offer is tight. We estimate these costs in [18]. We hope this class of intention-disguised algorithmic trading can reduce the profitability of and incentive for exploiting trade information, and alter market behavior as a whole. To that end, understanding these costs, and reducing them, is important.

Open Questions and Future Research

We believe that either finding the lower bound of the brokers that need to be used (in terms of $f(n)$) or finding a better strategy using fewer brokers may be possible. Furthermore, the sign of a trade with any one broker may be inferred by an observer using a trade direction algorithm such as that developed by Ellis,

Michaely and O'Hara [6] or Peterson and Sirri [15]. Our strategies are unaffected, assuming that all trades are filled in one round. However, realistically, such trades may take multiple rounds. On the other hand, in practice, there are also often other market participants trading, thus creating cover noise against identifying the trades initiated by the LMP. Even in an extremely illiquid market with no other active trading participants, the orders being worked by brokers are not synchronous in practice. Therefore, even if a broker with malicious intention is able to deduce the signs of other brokers, he cannot front run confidently that he has seen all the relevant trades initiated by the LMP.

Acknowledgement. Zhenming Liu is supported in part by NSF CCF-0634923.

References

1. Brain, S.: A front-running smile? Traders Magazine (May 2005), <http://www.wisdom.weizmann.ac.il/~oded/zk-tut02.html>
2. Chacko, G.: Personal Communication (August 2009)
3. Di Crescenzo, G.: Privacy for the stock market. In: Proc. Financial Cryptography and Data Security (2002)
4. Domowitz, I., Yegerman, H.: The cost of algorithmic trading: a first look at comparative performance. In: Algorithmic Trading: Precision, Control, Execution (March 2005)
5. Domowitz, I., Yegerman, H.: Measuring and interpreting the performance of broker algorithms. In: ITG Inc. Research Report (August 2005)
6. Ellis, K., Michaely, R., O'Hara, M.: The accuracy of trade classification rule: evidence from NASDAQ. Journal of Financial and Quantitative Analysis (2000)
7. Goldreich, O.: Zero-knowledge: a tutorial. Accessed through, <http://www.wisdom.weizmann.ac.il/~oded/zk-tut02.html>
8. Goldreich, O., Petrank, E.: Quantifying knowledge complexity. In: 32nd IEEE Symposium on Foundations of Computer Science (1996)
9. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. In: 17th Annual ACM Symposium of Theory of Computing (1985)
10. Harris, L.: Trading and exchanges: market microstructure for practitioners. Oxford University Press, Oxford (2003)
11. Kearns, M., Nevyvaka, Y., Papandreou, A., Sycara, K.: Electronic Trading in Order-Driven Markets: Efficient Execution. In: IEEE Conference on Electronic Commerce, CEC (2005)
12. Keim, D.B., Madhavan, A.: The upstairs market for large-block transactions: analysis and measurement of price effects. The Review of Financial Studies (1996)
13. Kumar, R., Sarin, A., Shastri, K.: The behavior of option Price Around Large Block Transactions in the Underlying Security. The Journal of Finance (1992)
14. Madhavan, A.: VWAP Strategies. In: Investment Guides, Transaction Performance (Spring 2002)
15. Peterson, M., Sirri, E.: Evaluation of biases in execution cost estimates using trade and quote data. Journal of Financial Markets (2002) (forthcoming)
16. Thorpe, C., Parkes, D.C.: Cryptographic securities exchanges. In: Financial Cryptography and Data Security (2007)
17. Thorpe, C., Parkes, D.C.: Cryptographic combinatorial securities exchanges. In: Financial Cryptography and Data Security (2009)
18. Yuen, W., Syverson, P., Liu, Z., Thorpe, C.: Intention-Disguised Algorithmic Trading. Harvard School of Engineering and Applied Sciences Tech. Report TR-01-10

When Information Improves Information Security^{*}

(Short Paper)

Jens Grossklags¹, Benjamin Johnson², and Nicolas Christin²

¹ Center for Information Technology Policy, Princeton University

² CyLab, Carnegie Mellon University

jensg@princeton.edu,

{johnsonb,nicolasc}@andrew.cmu.edu

Abstract. This paper presents a formal, quantitative evaluation of the impact of bounded-rational security decision-making subject to limited information and externalities. We investigate a mixed economy of an individual rational expert and several naïve near-sighted agents. We further model three canonical types of negative externalities (weakest-link, best shot and total effort), and study the impact of two information regimes on the threat level agents are facing.

Keywords: Game Theory, Security Economics, Bounded Rationality, Limited Information.

1 Introduction

Users frequently fail to deploy, or upgrade security technologies, or to carefully preserve and backup their valuable data [10,12], which leads to considerable monetary losses to both individuals and corporations every year. A partial interpretation of this state of affairs is that *negative externalities* impede end-users' investments in security technologies [11,14]. Negative network externalities occur when the benefit derived from adopting a technology depend on the actions of others as is frequently the case in the context of network security. For example, users who open and respond to unsolicited advertisements increase the load of spam for all participants in the network, including participants who are making the effort to adopt secure practices. Similarly, choosing a weak password for a corporate VPN system can facilitate compromises of many user accounts, possibly including those of individuals with strong passwords if trust relationships inside the VPN exist.

In other words, a rational user facing negative externalities could make the decision *not* to invest in security primitives given that their personal investment may only marginally matter if other users are adopting insecure practices, or if the perceived cost of a security breach significantly exceeds the cost of investing in security [9].

* We thank John Chuang for his helpful comments to an earlier version of this paper. This work is supported in part by CyLab at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office, by the National Science Foundation under ITR awards ANI-0331659 (100x100) and CCF-0424422 (Team for Research in Ubiquitous Secure Technology), and by a University of California MICRO project grant in collaboration with DoCoMo USA Labs.

In prior work, we addressed different canonical types of interdependencies, but we focused our analysis on users capable of gathering all relevant information and correctly understanding all implications of interconnectedness [4,5]. This paper extends our prior analysis by relaxing several restrictive assumptions on users' rationality and information availability.

First, we anticipate the vast majority of users to be *non-expert*, and to apply approximate decision-rules that fail to accurately appreciate the impact of their decisions on others [1]. In particular, in this paper, we assume non-expert users to conduct a simple self-centered cost-benefit analysis, and to neglect externalities. Such users would secure their system only if the vulnerabilities being exploited can cause significant harm or a direct annoyance to them (e.g., their machines become completely unusable), but would not act when they cannot perceive or understand the effects of their insecure behavior (e.g., when their machine is used as a relay to send moderate amounts of spam to third parties). In contrast, an advanced, or expert user fully comprehends to which extent her and others' security choices affect the network as a whole, and responds rationally.

Second, we address how the security choices by users are mediated by the information available on the severity of the threats the network faces. We assume that each individual faces a randomly drawn probability of being subject to a direct attack. Indeed in practice, different targets, even if they are part of a same network, are not all equally attractive to an attacker: a computer containing payroll information is, for instance, considerably more valuable than an old "boat anchor" sitting under an intern's desk. Likewise, a machine may be more attractive than another due to looser restrictions in the access policies to the physical facility where the machine is located.

As an initial step, we study the strategic optimization behavior from the perspective of a sophisticated user in an economy of inexperienced users, using three canonical security games that account for externalities [4]. This approach results in a decision-theoretic model [2,3,13]. We present the mathematical formulation and analysis methodology in the following section, which is based on our recent work focusing exclusively on the weakest-link externality [6].

2 Model and Analysis

Basic Model: Consider a game in which each of N network users is responsible for choosing security investments for their individual node. Each player begins the game with an initial endowment M , and suffers a maximum loss of L if a security breach occurs. The risk of a security breach is determined by an exogenous probability $p_i \in [0, 1]$, which for this paper we assume to be uniformly distributed. Security risks can be mitigated in two distinct ways. We denote by *protection* those security investment strategies which benefit the public network (such as installing antivirus software or firewalls), and we denote by *self-insurance* those strategies which benefit only the contributing user (such as keeping private data backups) [4]. The cost of full protection investment is denoted b and the cost of full self-insurance is denoted c . Each player chooses a protection investment level $e_i \in [0, 1]$ and a self-insurance investment level $s_i \in [0, 1]$. The manner in which protection strategies jointly affect players in the network is determined by an aggregate protection function $H(e_1, \dots, e_N)$, of which we will consider three types:

weakest link ($H(e_1, \dots, e_n) = \min_j e_j$), *best shot* ($H(e_1, \dots, e_n) = \max_j e_j$), and *total effort* ($H(e_1, \dots, e_N) = \frac{1}{N} \sum_j e_j$). The utility for player i is given by

$$U(i) = M - p_i L(1 - H(e_1, \dots, e_N))(1 - s_i) - b e_i - c s_i. \quad (1)$$

Bounded rationality and limited information: We consider distinct approaches to relax assumptions on user rationality and information availability.

First, we distinguish users based on their treatment of network interdependencies. We say a player is *naïve* if she does not take network interdependency into account in her decisions, i.e., she operates under the assumption that $H(e_1, \dots, e_N) = e_i$. Whereas a player is of the type *expert* if she correctly perceives and understands the interdependent nature of the game. That is, she properly considers the role of H into her payoff function.

Second, we distinguish between the level of information users have about others' risks. We say that a player has *complete information* if she knows all the risk factors associated to the other players. In contrast, a player has *incomplete information* if she knows her own risks but not the risks of other players. Clearly, a naïve player does not use the external risk information (because she is not aware of its effects), but an expert player does. To provide a decision framework for an expert with limited information, we assume that the distribution on risk parameters is known to all players. Thus, an expert with limited information can still conduct an expected cost-benefit analysis to make a strategic decision. This setup allows us to study the extent to which the complete information is beneficial to the players and to the network.

Methodology: The overarching aim of our analysis is to understand how expertise and knowledge affect player payoffs. To this end, our study considers a single expert agent in a field of $N - 1$ naïve agents, subject to one of the two information conditions, and the incentives of the utility function defined above.

We begin by determining, for each of the three games, (best shot, weakest link, and total effort), and for each of the two information conditions, (complete and incomplete), the payoff-maximizing strategy for an expert in that game. These strategy conditions involve the parameters b, c, L, M, N, p_i and in the case of complete information p_j for $j \neq i$.

We next compute an average or expected payoff for the expert as a result of playing this strategy. The expected value is taken with respect to the various probabilities p_i and p_j , which we assume are each drawn independently from the uniform distribution on $[0, 1]$. This final expected payoff is a function of parameters b, c, L, M , and N . The required computations often require us to consider selected parameter orientations as separate cases; but there are a small number of separate cases (at most 6) for each game, and the expected payoff functions and case conditions can be recorded neatly in tables.

We compute these payoff functions for an expert with complete information, for an expert with incomplete information, and also for one of the many naïve players. These functions tell the whole story in terms of how a player in this game, with a given level of knowledge and expertise, will fare in a given parameter configuration of this game. Unfortunately, the functions involve five free variables, and it remains to distill the information for further interpretive analysis.

To begin this part of the process, we fix the parameters related to the initial endowment, M , and the total maximum loss, L by setting $M = L = 1$. The assumption $M = L$ says that an agent can lose her entire endowment if a completely unprotected attack occurs, and may be considered as simply an interpretative statement about the initial endowment. The assumption $L = 1$ generates a relatively simple scaling effect – while it does affect the gross payoffs linearly, the assumption does not, for example, affect the configuration of other parameters that yield the minimum or maximum payoff. After incorporating these assumptions, what remains is a payoff function that depends only on the cost of protection, b , the cost of self-insurance, c , and the number of players, N .

The final step is to isolate the parameter conditions that yield interesting and substantive results. It turns out that there is only a narrow range of parameter configuration in which a substantial payoff difference exists between various agent types, and so we focus our attention on those cases.

3 Results

Due to limited space in this version of the paper, we exemplify this methodology by focusing on the case of an expert with limited information in the best shot game. Complete analytical results for all three games, – strategies and expected payoffs, together with all accompanying derivations – may be found in our companion technical report [8] and in our in-depth discussion of the weakest-link externality [6].

3.1 Analytical Results

Consider the factors influencing the decisions of an expert with incomplete information in the best shot game. First, note that because of the linear, monotonous nature of the utility function given in Eqn. (1), only three strategies are potentially utility-maximizing for player i : passivity ($e_i = 0, s_i = 0$), full protection ($e_i = 1, s_i = 0$), and full insurance ($e_i = 0, s_i = 1$). Any other strategy can be shown to result in sub-optimal payoffs [4,8].

Now, if player i protects, her payoff is $M - b$; if she insures, her payoff is $M - c$, and if she does neither, then her payoff is $M - p_i L(1 - Pr_{\neg i}^*)$ where $Pr_{\neg i}^*$ is the probability that one of the $N - 1$ naïve players protects. Because we know the strategy for naïve player j is to protect if and only if $b \leq p_j L$ and $b \leq c$, we may compute $Pr_{\neg i}^*$ as

$$\begin{cases} 0 & \text{if } c < b \\ 1 - \left(\frac{b}{L}\right)^{N-1} & \text{if } b \leq c \end{cases} .$$

Hence the payoff for the expert when she does nothing is

$M - p_i L$ when $c < b$, and $M - p_i L \left(\frac{b}{L}\right)^{N-1}$ when $b \leq c$. These values can be found in the lower portion of Table 1, which also records initial payoffs as the results of strategy choices under the complete information condition.

The next step is determine what our expert should do given known values for the parameters. Consider the case $b \leq c$. Here the expert with incomplete information would never choose to insure because protection is cheaper for the same result. On the other hand, the assumptions that $b \leq L \leq 1$ imply that the inequality $M - p_i L \left(\frac{b}{L}\right)^{N-1} \leq M - b$

Table 1. Best shot security game: Payoffs for different strategies under different information conditions

Case	Information Type	Payoff Passivity	Payoff Self-Insurance	Payoff Protection
$c < b$	Complete	$M - p_i L$	$M - c$	$M - b$
$b \leq c$ and $\max_{j \neq i} p_j < b/L$	Complete	$M - p_i L$	$M - c$	$M - b$
$b \leq c$ and $b/L \leq \max_{j \neq i} p_j$	Complete	M	$M - c$	$M - b$
$c < b$	Incomplete	$M - p_i L$	$M - c$	$M - b$
$b \leq c$	Incomplete	$M - p_i L (b/L)^{N-1}$	$M - c$	$M - b$

is tautological. Hence the expert will also never protect. We find that in the parameter case $b \leq c$, an expert with incomplete information will always choose to be passive. This result can be found in Table 2 under the appropriate parameter case and information condition.

Table 2. Best shot security game: Conditions to select protection, self-insurance or passivity strategies

Case	Information Type	Conditions Passivity	Conditions Self-Insurance	Conditions Protection
$c < b$	Complete	$p_i < c/L$	$p_i \geq c/L$	Never
$b \leq c$ and $\max_{j \neq i} p_j < b/L$	Complete	$p_i < b/L$	Never	$p_i \geq b/L$
$b \leq c$ and $b/L \leq \max_{j \neq i} p_j$	Complete	Always	Never	Never
$c < b$	Incomplete	$p_i < c/L$	$p_i \geq c/L$	Never
$b \leq c$	Incomplete	Always	Never	Never

Finally, to determine an expected payoff for the expert with incomplete information in the best shot game with $b \leq c$, we compute the probability that she protects (over her draw of p_i) times the expected payoff for protection, plus the probability that she insures times the expected payoff for insuring, plus the probability that she is passive times her expected payoff for passivity. The end result is $M - \frac{L}{2} (\frac{b}{L})^{N-1}$. This value is recorded in Table 3, along with expected total payoffs for other parameter cases and player information conditions.

Tables and derivations for the other two canonical games can be found in the companion technical report [8].

3.2 Applications

We highlight two applications from our study. A more thorough discussion can be found in the technical report [8].

Our first result is that the range of security parameters conducive to an environment in which limited information plays a significant role is somewhat restricted, and becomes more restricted as the number of players increases. Consider the three graphs

Table 3. Best shot security game: Total expected game payoffs

Case	Information Type	Total Expected Payoff
$c < b$	Complete	$M - c + c^2/2L$
$b \leq c$	Complete	$M - b(1 - b/2L)(b/L)^{N-1}$
$c < b$	Incomplete	$M - c + c^2/2L$
$b \leq c$	Incomplete	$M - L/2(b/L)^{N-1}$
$c < b$	Naïve	$M - c + c^2/2L$
$b \leq c$	Naïve	$M - b + b^2/2L$

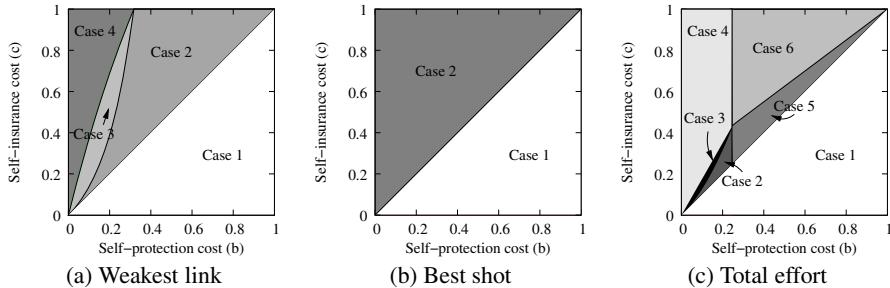


Fig. 1. Strategy boundaries in the incomplete information scenario for the expert player as a function of protection and self-insurance costs. (Here the number of players is fixed at $N = 4$, and the initial endowment and the potential loss are fixed at $M = L = 1$.) For the relevant analytic results used to construct these graphs, please refer to the technical report [8].

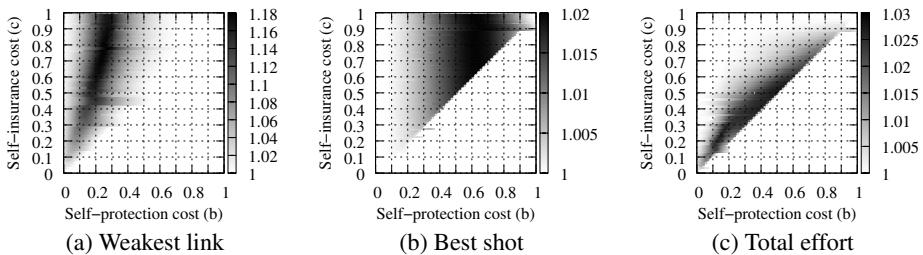


Fig. 2. Heat plots showing the extent of payoff discrepancy between the expert with complete information and the expert with incomplete information as a function of protection and self-insurance costs. (As in Fig. 1, $N = 4$, and $M = L = 1$.) For the relevant analytic results used to construct these graphs, please refer to the technical report [8].

in Figure 1. These graphs show dividing lines between various dominant strategies as function of the two types of investment costs – protection and self-insurance. Comparing these to the graphs in Figure 2, the direct relationship is clear between the parameter conditions that determine certain dominant strategies and the parameter conditions under which additional information has an effect on an expert player's payoff. From these graphs we can tell that information plays a substantial role in the smaller-sized cases,

but in the other cases (i.e., N large), the additional information has little to no effect. Another striking observation (not shown in these figures) is that for every game, the size of the interesting parameter cases shrinks toward zero as we increase the number of players.

Our second result is that, even in the parameter ranges in which information plays a role, that role is limited. In all games, the maximum percent increase in expected payoff as a result of additional information is around 18%. This occurs in the weakest link game with exactly four players, and under fixed costs of protection and self-insurance that are set to produce maximum information impact. So having information about costs and risks of others is not a dominant factor – in other words, using educated guesses to predict these values does not hurt the bottom line too much. In contrast with this observation, the expected payoff of the naïve player is greatly reduced as a result of his imperfect situational perception. So if network users do not understand the interdependent nature of their security threats, their payoff is greatly reduced.

3.3 Value of Information

The degree of darkening in the graphs of Figure 2 allows us to visualize the payoff discrepancy as a result of incomplete information, and is something we might refer to as the value of information. It is nontrivial to arrive at a definitive answer for this quantity’s best measure and is in fact the subject of a full related paper [7], but we consider the following ratio definition as a first step towards the goal of reasonably quantifying the value of information in this context.

$$\frac{\text{Expected payoff of an expert agent in the complete information environment}}{\text{Expected payoff of an expert agent in the incomplete information environment}}$$

4 Conclusions

In our work we emphasize that security decision-making is shaped by the structure of the task environment as well as the knowledge and computational capabilities of the agents. In our model, decisions are made from three distinct security actions (self-protection, self-insurance or passivity) to confront the security risks of weakest-link, best shot and total effort interdependencies [4,14]. In these environments, we investigate the co-habitation of a single fully rational expert and $N - 1$ naïve agents. The naïve agents fail to account for the decisions of other agents, and instead follow a simple but reasonable self-centered rule-of-thumb. We further study the impact of limited information on the rational agent’s choices.

We find that in general, the naïve agents match the payoff of the expert when self-insurance is cheap, but not otherwise. Even with limited information, the sophisticated agent can generally translate her better structural understanding into decisions that minimize wasted protection investments, or an earlier retreat to the self-insurance strategy when system-wide security is (likely) failing.

To analyze the impact of the different information conditions we have proposed a new mathematical formalization. We measure the value of complete information as the ratio of the payoff in the complete information environment to the payoff in the

incomplete information environment. Our analysis of Figure 2 is a first step in that direction, however, we defer a more formal analysis to a companion research paper [7].

Finally, a system designer is not only interested in the payoffs of the network participants given different information realities (e.g., due to frequent changes in attack trends). He is also concerned with how well-fortified the organization is against attacks. To that effect we plan to include a more thorough presentation of the parameter conditions that cause attacks to fail due to system-wide protection, and when they succeed (due to coordination failures, passivity, and self-insurance).

References

1. Acquisti, A., Grossklags, J.: Privacy and rationality in individual decision making. *IEEE Security & Privacy* 3(1), 26–33 (January–February 2005)
2. Cavusoglu, H., Raghunathan, S., Yue, W.: Decision-theoretic and game-theoretic approaches to IT security investment. *J. Mgt. Info. Sys.* 25(2), 281–304 (Fall 2008)
3. Gordon, L., Loeb, M.: The economics of information security investment. *ACM Transactions on Information and System Security* 5(4), 438–457 (November 2002)
4. Grossklags, J., Christin, N., Chuang, J.: Secure or insure? A game-theoretic analysis of information security games. In: Proc. WWW 2008, Beijing, China, pp. 209–218 (April 2008)
5. Grossklags, J., Christin, N., Chuang, J.: Security and insurance management in networks with heterogeneous agents. In: Proc. ACM EC 2008, Chicago, IL, pp. 160–169 (July 2008)
6. Grossklags, J., Johnson, B.: Uncertainty in the weakest-link security game. In: Proc. GameNets 2009, Istanbul, Turkey, pp. 673–682 (May 2009)
7. Grossklags, J., Johnson, B., Christin, N.: The price of uncertainty in security games. In: Proc (online) WEIS 2009, London, UK (June 2009)
8. Grossklags, J., Johnson, B., Christin, N.: When information improves information security. Tech. rep., UC Berkeley & Carnegie Mellon University, CyLab (February 2009), <http://www.cylab.cmu.edu/research/techreports/tr-cylab09004.html>
9. Herley, C.: So long, and no thanks for the externalities: The rational rejection of security advice by users. In: Proc. NSPW 2009, Oxford, UK (September 2009)
10. Kabooza. Global backup survey: About backup habits, risk factors, worries and data loss of home PCs (January 2009), <http://www.kabooza.com/globalsurvey.html>
11. Kunreuther, H., Heal, G.: Interdependent security. *Journal of Risk and Uncertainty* 26(2-3), 231–249 (March 2003)
12. NCSA/Symantec. Home user study (October 2008), <http://staysafeonline.org/>
13. Schechter, S., Smith, M.: How much security is enough to stop a thief? In: Proc. IFCA FC 2003, Gosier, Guadeloupe, pp. 122–137 (January 2003)
14. Varian, H.: System reliability and free riding. In: Camp, L., Lewis, S. (eds.) *Economics of Information Security, Advances in Information Security*, vol. 12, pp. 1–15. Kluwer Academic Publishers, Dordrecht (2004)

BetterThanPin: Empowering Users to Fight Phishing (Poster)

Teik Guan Tan

Data Security Systems Solutions Pte Ltd
teikguan@ds3global.com

Abstract. The BetterThanPin concept is an online security service that allows users to enable almost any Cloud or Web-based account (e.g. Gmail, MSN, Yahoo, etc) to be protected with “almost” 2-factor authentication (2FA). The result is that users can now protect their online accounts with better authentication, without waiting for the service or cloud provider.

Keywords: 2-factor Authentication, Cloud Security.

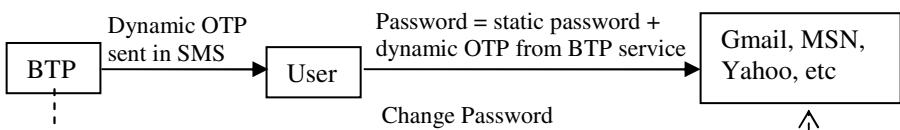
1 The Problem

The strength of authentication security for online accounts is solely dependent on what the online provider is willing to implement. Basically, if the online provider does not implement strong 2-factor authentication, there's probably nothing the user can do about it, besides not using it. Popular online services such as Gmail, MSN and Facebook which have hundreds of millions of subscribers will find it cost-prohibitive and economically unviable to implement strong authentication to their applications simply to satisfy a small percentage (but sizeable in absolute terms) number of users.

2 The Idea

The rationale behind BetterThanPin is to convert any online account authentication from the standard UserID-Password to the more secure UserID-Password+dynamic password, without needing to change anything on the online service. In the BetterThanPin system, an online Change-Password process is run to regularly update the expected login password for the online account to include the both the user's chosen password plus dynamic password as the expected login password.

The 2nd factor tokens we expect to be available for obtaining the dynamic password include Hardware/Software dynamic password tokens, as well as dynamic passwords transmitted via Email or SMS to mobile phones. Users can choose the type of token depending on convenience, costs and level of security needed. The dynamic passwords can be changed on an hourly, daily or weekly basis.



In the example above, a user using SMS as the 2nd factor authentication will receive an SMS daily, containing dynamic password from the BetterThanPin service.

Certification Intermediaries and the Alternative (Poster)

Pern Hui Chia

Centre for Quantifiable Quality of Service in Communication Systems (Q2S)*

Albano and Lizzeri showed that if quality is endogenous, the existence of a certification intermediary will improve product quality [1]. If quality is exogenous, an intermediary will also improve welfare by not certifying unsafe products; however, it is optimal for a monopolistic intermediary to disclose only minimal information necessary to induce trade [3]. Indeed, many certification schemes today specify only whether a product (website, software) has met a minimal set of requirements. When the criteria are lenient, costs for certification will be indifferent, causing the separating equilibrium to diminish and thus not providing a reliable signal. Edelman showed empirically that TRUSTe-certified websites were more likely to be untrustworthy compared to non-certified websites [2].

The role of certification intermediaries is hence an interesting issue. An important work, following the surge of 3rd party mobile applications, is to analyze the **different certification schemes on mobile platforms**: independent software testing is required for Symbian, Java and Windows Mobile certification; Apple, however, solely determines which software can be marketed in the iTunes Appstore; on the other hand, Android applications can be distributed with self-signed certificates. Such an analysis should take into account of the incentives of technology intermediaries and low user awareness for certification schemes.

There are reasons to doubt if strict enforcement of certification would always be appropriate. After all, the verdict is intrinsically centralized and often based on minimal criteria. A plausible strategy is to engage users as alternative intermediaries. Sourcing for grassroots scrutiny can benefit from many eyeballs and be driven by social capital such as reputation and reciprocity. A hurdle is the limited user capability to evaluate security; innovations to enable simple and objective assessment by ordinary users will be helpful. Analyzing the economics of **users as alternative certification intermediaries** (incentive-design, evolution of cooperation, free-riding and reliability tradeoff) can be very interesting.

References

1. Albano, G.L., Lizzeri, A.: Strategic Certification and Provision of Quality. *International Economic Review* (2001)
2. Edelman, B.: Adverse selection in online ‘trust’ certifications. In: Proc. WEIS (2006)
3. Lizzeri, A.: Information Revelation and Certification Intermediaries. *Rand Journal of Economics* (1999)

* Q2S, Centre of Excellence, appointed by The Research Council of Norway, is funded by the Research Council, Norwegian University of Science and Technology (NTNU) and UNINETT. <http://www.q2s.ntnu.no>

SeDiCi: An Authentication Service Taking Advantage of Zero-Knowledge Proofs

Slawomir Grzonkowski

Digital Enterprise Research Institute
National University of Ireland, Galway
IDA Business Park, Galway, Ireland
slawomir.grzonkowski@deri.org

Transmission of users' profiles over insecure communication means is a crucial task of today's ecommerce applications. In addition, the users have to create many profiles and remember many credentials. Thus they retype the same information over and over again. Each time the users type their credentials, they expose them to phishing or eavesdropping attempts. These problems could be solved by using Single Sign-on (SSO). The idea of SSO is that the users keep using the same set of credentials when visiting different websites. For web-applications, OpenID¹. is the most prominent solution that partially implements SSO. However, OpenID is prone to phishing attempts and it does not preserve users' privacy [1].

To address phishing and eavesdropping, we developed SeDiCi, a secure SSO. This technology takes advantage of Zero-Knowledge Proof (ZKP) authentication that is based on our previous work [2]. The technology also supports REST-based API that enables taking advantage of the service by mobile phones, web-applications and other client applications. To provide interoperability with other systems, SeDiCi stores data using semantic web standards such as FOAF. Thus, the users are able to use their profiles and social networks from other services.

Acknowledgement

The work presented in this paper was supported (in part) by the Lion project supported by Science Foundation Ireland under Grant No. RSF0844 and Enterprise Ireland under Grant No. REI 1005. A method and apparatus for authenticating a user is a pending patent. The author would like to thank to Lukasz Korczynski for his help with the implementation.

References

1. Adida, B.: Bea mAuth: two-factor web authentication with a bookmark. In: CCS 2007: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 48–57. ACM, New York (2007)
2. Grzonkowski, S., Zaremba, W., Zaremba, M., McDaniel, B.: Extending web applications with a lightweight zero knowledge proof authentication. In: CSTST 2008: Proceedings of the 5th International Conference on Soft Computing as Transdisciplinary Science and Technology, pp. 65–70. ACM, New York (2008)

¹ OpenID 2.0 spec: http://openid.net/specs/openid-authentication-2_0.html

Poster Abstract: Security in Commercial Applications of Vehicular Ad-Hoc Networks*

Pino Caballero-Gil, Jezabel Molina-Gil,
Cándido Caballero-Gil, and Candelaria Hernández-Goya

Department of Statistics, O.R. and Computation, University of La Laguna, 38271 La Laguna, Tenerife, Spain
`{pcaballe,jmmolina,ccabgil,mchgoya}@ull.es`

Abstract. This work proposes a combined protocol for incentive-based cooperation and strong authentication in Vehicular Ad-hoc Networks.

1 Introduction

A Vehicular Ad-hoc NETwork (VANET) is a special type of ad-hoc network used to provide wireless communications that may be: Vehicle-TO-Vehicle (V2V), Vehicle-TO-Infrastructure (V2I) or Infrastructure-TO-Vehicle (I2V). Its major drawback is the required complex networking management system and security protocols, so both cooperation and strong node authentication are needed.

Many schemes to stimulate cooperation may be found in the bibliography. Buttyan and Hubaux introduced in [1] the use of virtual credit. Li and Wu proposed in [2] a receipt counting reward. Different schemes depending on the type of packets are here defined so that they assign incentives to vehicles according to their contribution in packet forwarding, trying to achieve fairness and stimulate participation. In particular, leaders of groups are encouraged to look for forwarding nodes in their groups. With respect to authentication, based on privacy requirements, Identity-Based cryptography [3] is proposed for I2V while in V2I communications a challenge-response scheme using a secret-key approach based on random key trees is defined. Also to provide privacy, ring signatures are used for V2V communications between groups, while secret-key authentication is the proposed solution inside groups in order to save communications. These ingredients are mixed in a cooperation and authentication protocol for VANETs.

References

1. Buttyan, L., Hubaux, J.P.: Security and Cooperation in Wireless Networks. Cambridge Univ. Press, Cambridge (2007)
2. Li, F., Wu, J.: FRAME: An Innovative Incentive Scheme in Vehicular Networks. In: Proc. of IEEE International Conference on Communications (2009)
3. 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)

* Research supported by the Spanish Ministry of Education and Science and the European FEDER Fund under TIN2008-02236/TSI Project, and by the Agencia Canaria de Investigación, Innovación y Sociedad de la Información under PI2007/005 Project.

Domain Engineering for Automatic Analysis of Financial Applications of Cryptographic Protocols

(Poster)

Lilia Georgieva

Department of Mathematical and Computer Sciences, Heriot-Watt University,
Edinburgh, UK
lilia@macs.hw.ac.uk

Abstract. Our aim is to develop a general framework for study and automatic analysis of properties of cryptographic protocols in order to establish design guidelines for business applications which would ensure correctness.

The rapid growth of e-commerce has led to increasing need for secure online transactions, authentication, and credential management. The design of security protocols to address these needs difficult and error-prone. Despite advances in cryptology, security, database systems and database mining, there is no infrastructure for handling sensitive data.

We investigate how formal methods and tools for automatic verification can address these needs. We study the framework of domain engineering [1] which is suitable for business process modelling. We formalise the desired properties in typed first-order logic and apply model checking to establish correctness. Desired goals of security protocols which can be modelled and are of relevance to financial services include authenticity (correct identification of the sender), integrity (the sent data is identical to the received data), secrecy (non disclosure of a secret message to the attacker), anonymity.

The common approach for formal logic based protocol verification involves (i) formal specification of the properties of the protocol steps; (ii) formal specifications of the protocol assumptions; (iii) formal specifications of the protocol goals; and (iv) applications of the axioms and inference rules of the logic to assumptions and protocol steps to derive the goals.

We model cryptographic protocols in typed first-order logic and use the model checker Alloy [2] with the aim to design a more general framework for automatic analysis of cryptographic protocols in which more properties of the cryptographic protocols can be modelled.

References

1. Bjorner, D.: A triptych software development paradigm: Domain, requirements and software towards a model development of a decision support system for sustainable development. In: Olderog, E.-R., Steffen, B. (eds.) Correct System Design. LNCS, vol. 1710, pp. 29–60. Springer, Heidelberg (1999)
2. Jackson, D.: Software abstractions: Logic, language, and analysis (2006)

hPIN/hTAN: Low-Cost e-Banking Secure against Untrusted Computers

Shujun Li¹, Ahmad-Reza Sadeghi², and Roland Schmitz³

¹ University of Konstanz, Germany

² Ruhr-University of Bochum, Germany

³ Stuttgart Media University, Germany

Abstract. We propose hPIN/hTAN, a low-cost token-based e-banking protection scheme when the adversary has *full* control over the user's computer. Compared with existing hardware-based solutions, hPIN/hTAN depends on neither second trusted channel, nor secure keypad, nor computationally expensive encryption module.

Due to the rapid progress of the Internet, e-banking has become more and more popular all over the world and security is considered as one of the most serious issues of e-banking. The earliest and simplest defense protecting e-banking systems is user authentication based on static PINs. Since static PINs are prone to identity theft, two-factor user authentication such as PIN/TAN has been widely adopted to make e-banking more secure. However, PIN/TAN cannot resist man-in-the-middle (MitM) attack, whose aim is to manipulate transactions. In the strongest form of MitM attacks, the user's computer is under the *full* control of the adversary, who can observe and tamper with all the communications between the user and the e-banking server. The wide spread of malware over the Internet renders such advanced MitM attacks possible in reality.

In this poster, we propose hPIN/hTAN, the first (to the best of our knowledge) hardware-based solution against MitM attacks that depends on neither second trusted channel nor secure keyboard nor computationally expensive encryption (such as PKC). Instead, hPIN/hTAN bases its security only on proper use of a cryptographic hash function and active involvement of human attention.

The hPIN/hTAN includes two specific protocols – hPIN and hTAN, which protect the login process and online transactions, respectively. The involved parties include a human user, a trusted USB-token issued by the bank to the user, an untrusted terminal computer, and the e-banking server. The USB-token is equipped with a trusted display and shares a secret with the server.

The core of the hPIN protocol is a random code shown on the trusted display of the USB-token, which makes it possible for the user to input a transformed PIN on the untrusted computer without leaking the PIN. After user authentication to the USB-token, the hPIN protocol achieves mutual authentication between the USB-token and the server. In the hTAN protocol, the user verifies the transaction data *simultaneously* via the display of the USB-token while typing them on the keyboard of the untrusted computer. Then the USB-token and the server perform a transaction verification process based on the shared secret.

Author Index

- Algasinger, Michael 328
Anderson, Ross 336
Barth, Adam 192
Bartlett, Peter L. 192
Bethencourt, John 400
Bisht, Prithvi 272
Björkqvist, Mathias 160
Bonneau, Joseph 98
Bußmeyer, Daniel 320
Caballero-Gil, Cándido 427
Caballero-Gil, Pino 427
Cachin, Christian 160
Čagalj, Mario 351
Camenisch, Jan 66
Canard, Sébastien 82
Catrina, Octavian 35
Chen, Xiaofeng 304
Chia, Pern Hui 425
Chothia, Tom 20
Christianson, Bruce 4
Christin, Nicolas 416
Cranor, Lorrie Faith 3
Csaba, György 328
Damgård, Ivan 367
De Cristofaro, Emiliano 143
Dingledine, Roger 238
Dubovitskaya, Maria 66
Edelman, Benjamin 175
Edwards, Jonathan W. 51
Gardner, Ryan W. 312
Garera, Sujata 312
Georgieva, Lilia 428
Goldberg, Ian 359
Gouget, Aline 82
Gröbert, Felix 320
Grossklags, Jens 416
Groza, Bogdan 391
Grzonkowski, Sławomir 426
Guajardo, Jorge 375
Haas, Robert 160
Hao, Feng 383
Hernández-Goya, Candelaria 427
Hilgers, Christian 328
Hubaux, Jean-Pierre 2
Hu, Xiao-Yu 160
Jaeger, Christian 328
Järvinen, Kimmo 207
Johnson, Benjamin 416
Just, Mike 98
Kaminsky, Dan 289
Karger, Paul A. 51
Kasper, Timo 343
Kate, Aniket 359
Keller, Marcel 367
Kesdogan, Dogan 114
Kiayias, Aggelos 257
Kim, Jangseong 304
Kim, Kwangjo 304
Kolesnikov, Vladimir 207
Kurmus, Anil 160
Li, Shujun 429
Liu, Zhenming 408
Matthews, Greg 98
Maurer, Ueli 1
McIntosh, Suzanne K. 51
Mennink, Bart 375
Minea, Marius 391
Mitchell, John C. 192
Molina-Gil, Jezabel 427
Moore, Tyler 175, 222
Moran, Tal 222
Murdoch, Steven J. 336
Mu, Yi 304
Neven, Gregory 66
Ngan, Tsuen-Wan “Johnny” 238
Paar, Christof 343
Palmer, Elaine R. 51

- Patterson, Meredith L. 289
Pawlitzek, René 160
Perković, Toni 351
Pimenidis, Lexi 114

Rubin, Aviel D. 312
Rubinstein, Benjamin I.P. 192
Rührmair, Ulrich 328

Sadeghi, Ahmad-Reza 207, 429
Sassaman, Len 289
Saxena, Amitabh 35
Saxena, Nitesh 351
Schäge, Sven 129
Schmitz, Roland 429
Schneider, Thomas 207
Schoenmakers, Berry 375
Schwenk, Jörg 129, 320
Shi, Elaine 400
Silbermann, Michael 343
Sistla, A. Prasad 272
Smirnov, Vitaliy 20
Song, Dawn 192, 400
Stajano, Frank 4

Stutzmann, Martin 328
Sundararajan, Mukund 192
Syverson, Paul 408

Tan, Teik Guan 424
Thorpe, Christopher 408
Tian, Haibo 304
Toll, David C. 51
Tsudik, Gene 143

Venkatakrishnan, V.N. 272
Vukolić, Marko 160

Wallach, Dan S. 238
Weber, Samuel 51
Wegener, Christoph 320
Wendolsky, Rolf 114
Westermann, Benedikt 114
Wong, Ford-Long 4
Wu, Qianhong 304

Yuen, William 408
Yung, Moti 257

Zhang, Fangguo 304