

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Alfred Kobsa

*University of California, Irvine, CA, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*TU Dortmund University, Germany*

Madhu Sudan

*Microsoft Research, Cambridge, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max Planck Institute for Informatics, Saarbruecken, Germany*

George Danezis (Ed.)

# Financial Cryptography and Data Security

15th International Conference, FC 2011  
Gros Islet, St. Lucia, February 28 - March 4, 2011  
Revised Selected Papers



Springer

Volume Editor

George Danezis  
Microsoft Research Cambridge  
Roger Needham Building  
7 J J Thomson Avenue, Cambridge CB3 0FB, UK  
E-mail: gdane@microsoft.com

ISSN 0302-9743  
ISBN 978-3-642-27575-3  
DOI 10.1007/978-3-642-27576-0  
Springer Heidelberg Dordrecht London New York

e-ISSN 1611-3349  
e-ISBN 978-3-642-27576-0

Library of Congress Control Number: 2011944281

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

© Springer-Verlag Berlin Heidelberg 2012

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

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

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

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# Preface

This volume contains the proceedings of the 15th International conference on Financial Cryptography and Data Security, held at the Bay Gardens Beach Resort, St. Lucia, February 28–March 4, 2011.

Financial cryptography and data security (FC) is a well-established international forum for research, advanced development, education, exploration and debate regarding information assurance in the context of finance and commerce. The conference covers all aspects of securing transactions and systems.

This year we assembled a diverse program featuring 26 papers and a panel on “The Future of Banking Security and Financial Transactions for the 21st Century.” The conference was opened by Jolyon Clulow, from Tesco Bank, with a keynote address on “What I Learnt When Trying to Build a Bank” and a closing talk by Markus Jakobsson from PayPal on “Why Mobile Security Is Not Like Traditional Security.”

The program was put together through a standard peer-review process by a technical Program Committee selected by the Program Chair. This year we received 56 full-length and 9 short submission. Each submission received at least three reviews from members of the Program Committee or outside experts – the vast majority of paper received four reviews. A further week-long discussion led to the selection of 16 full-length papers and 10 submissions as short papers to be included in the final program, on the basis of excellence, novelty and interest to the FC community.

The overall acceptance rate for full papers was 28%, while 40% of submissions were accepted at least as a short paper. The acceptance rate for full papers places FC in league with other competitive venues in computer security and cryptography. Yet, the generous time given to short papers ensures new ideas and promising work in progress have an opportunity to be heard.

This conference was made possible through the dedicated work of our General Chair, Steven Murdoch from the University of Cambridge, and our Local Arrangements Chair, Fabian Monrose from the University of North Carolina at Chapel Hill. The Program Chair would like to thank especially the Program Committee members and external reviewers for their expertise and dedication, both for selecting papers for the program as well as providing feedback to improve all submissions. Finally, the members of the International Financial Cryptography Association (IFCA) board should be acknowledged for keeping the FC conference going through the years. This year’s conference was made more affordable due to the generosity of our sponsors.

## Organization

The 15th International conference on Financial Cryptography and Data Security (FC 2011) was organized by the International Financial Cryptography Association (IFCA).

## Organizers

## General Chair

Steven Murdoch University of Cambridge, UK

## **Local Arrangements Chair**

Fabian Monrose University of North Carolina Chapel Hill, USA

## Program Chair

George Danezis Microsoft Research, UK

## Program Committee

Ross Anderson	University of Cambridge, UK
Tuomas Aura	Helsinki University of Technology, Finland
Lucas Ballard	Google, USA
Adam Barth	UC Berkeley, USA
Elisa Bertino	Purdue University, USA
Kevin Butler	University of Oregon, USA
Srdjan Capkun	ETH Zurich, Switzerland
Veronique Cortier	CNRS / LORIA, France
Ernesto Damiani	University of Milan, Italy
Claudia Diaz	K.U. Leuven, Belgium
Roger Dingledine	The Tor Project, USA
Orr Dunkelman	Weizmann Institute of Science, Israel
Simone Fisher-Hubner	Karlstad University, Sweden
Craig Gentry	IBM T.J. Watson Research Center, USA
Dieter Gollmann	Technische Universität Harburg, Germany
Rachel Greenstadt	Drexel University, USA
Jean-Pierre Hubaux	Ecole Polytechnique Federale de Lausanne, Switzerland
Markus Jakobsson	Indiana University, USA
Jaeyeon Jung	Intel Research, USA
Stefan Katzenbeisser	Technische Universität Darmstadt, Germany
Angelos Keromytis	Columbia University, USA

## VIII Organization

Arjen Lenstra	Ecole Polytechnique Federale de Lausanne, Switzerland
Helger Lipmaa	Cybernetica AS, Estonia
Evangelos Markatos	FORTH, Greece
David Molnar	Microsoft Research, USA
Tyler Moore	Harvard University, USA
David Naccache	Ecole normale superieure, France
Thomas Ristenpart	University of Wisconsin, USA
Peter Ryan	Universite du Luxembourg, Luxembourg
Ahmad-Reza Sadeghi	Ruhr-University Bochum, Germany
Rei Safavi-Naini	University of Calgary, Canada
Nigel Smart	University of Bristol, UK
Jessica Staddon	Google, USA
Angelos Stavrou	George Mason University, USA
Paul Syverson	Naval Research Laboratory, USA
Nicholas Weaver	International Computer Science Institute, USA
Moti Yung	Google, USA

## External Reviewers

Sadia Afroz	Seda Guerses	Qun Ni
Mina Askari	James Heather	Onur Ozen
Mira Belenkiy	Hans Hedbom	Daniel Page
Josh Benaloh	Mathias Humbert	Aanjhan Ranganathan
Stefan Berthold	Murtuza Jadliwala	Joel Reardon
Igor Bilogrevic	Dimitar Jetchev	Christian Rechberger
Joppe Bos	Ghassan Karame	Alfredo Rial
Christina Brzuska	Emilia Kasper	Eleanor Rieffel
Dario Catalano	Dalia Khader	Mark Ryan
Liqun Chen	Michael Kirkpatrick	Nashad Safa
Richard Chow	Lara Letaw	Juraj Sarinay
Daniel Colascione	Harshana Liyanage	Thomas Schneider
Jean-Sebastien Coron	Hans Loehr	Steffen Schulz
Boris Danev	Roel Maes	Ben Smyth
Maria Dubovitskaya	Claudio Marforio	Tomas Toft
Ruchith Fernando	Catherine Meadows	Ashraful Tuhin
Aurelien Francillon	Mohamed Nabeel	Zhe Xia
Julien Freudiger	Kris Gagne	Davide Zanetti
Georg Fuchsbauer	Martin Narayan	Ge Zhang

## Sponsors

Gold	Office of Naval Research Global
Silver	Research in Motion
Bronze	East Caribbean Financial Holding
	Google
In Kind	Lime
	WorldPay
	Financial Services Technology Consortium BITS

# Table of Contents

## Finacial Cryptography and Data Security (FC 2011)

Collective Exposure: Peer Effects in Voluntary Disclosure of Personal Data .....	1
<i>Rainer Böhme and Stefanie Pötzsch</i>	
It's All about the Benjamins: An Empirical Study on Incentivizing Users to Ignore Security Advice .....	16
<i>Nicolas Christin, Serge Egelman, Timothy Vidas, and Jens Grossklags</i>	
Evaluating the Privacy Risk of Location-Based Services .....	31
<i>Julien Freudiger, Reza Shokri, and Jean-Pierre Hubaux</i>	
Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance .....	47
<i>Jeremy Clark and Urs Hengartner</i>	
Malice versus AN.ON: Possible Risks of Missing Replay and Integrity Protection .....	62
<i>Benedikt Westermann and Dogan Kesdogan</i>	
Absolute Pwnage: A Short Paper about the Security Risks of Remote Administration Tools .....	77
<i>Jay Novak, Jonathan Stribley, Kenneth Meagher, and J. Alex Halderman</i>	
A Protocol for Anonymously Establishing Digital Provenance in Reseller Chains (Short Paper) .....	85
<i>Ben Palmer, Kris Bubendorfer, and Ian Welch</i>	
Impeding Individual User Profiling in Shopper Loyalty Programs.....	93
<i>Philip Marquardt, David Dagon, and Patrick Traynor</i>	
Beyond Risk-Based Access Control: Towards Incentive-Based Access Control .....	102
<i>Debin Liu, Ninghui Li, XiaoFeng Wang, and L. Jean Camp</i>	
Authenticated Key Exchange under Bad Randomness .....	113
<i>Guomin Yang, Shanshan Duan, Duncan S. Wong, Chik How Tan, and Huaxiong Wang</i>	

Oblivious Outsourced Storage with Delegation .....	127
<i>Martin Franz, Peter Williams, Bogdan Carbunar,     Stefan Katzenbeisser, Andreas Peter, Radu Sion, and     Miroslava Sotakova</i>	
Homomorphic Signatures for Digital Photographs .....	141
<i>Rob Johnson, Leif Walsh, and Michael Lamb</i>	
Revisiting the Computational Practicality of Private Information Retrieval .....	158
<i>Femi Olumofin and Ian Goldberg</i>	
Optimal One Round Almost Perfectly Secure Message Transmission (Short Paper) .....	173
<i>Mohammed Ashraful Alam Tuhin and Reihaneh Safavi-Naini</i>	
A New Approach towards Coercion-Resistant Remote E-Voting in Linear Time .....	182
<i>Oliver Spycher, Reto Koenig, Rolf Haenni, and Michael Schläpfer</i>	
An Attack on PUF-Based Session Key Exchange and a Hardware-Based Countermeasure: Erasable PUFs .....	190
<i>Ulrich Rührmair, Christian Jaeger, and Michael Algasinger</i>	
Peeling Away Layers of an RFID Security System .....	205
<i>Henryk Plötz and Karsten Nohl</i>	
Might Financial Cryptography Kill Financial Innovation? – The Curious Case of EMV .....	220
<i>Ross Anderson, Mike Bond, Omar Choudary,     Steven J. Murdoch, and Frank Stajano</i>	
hPIN/hTAN: A Lightweight and Low-Cost E-Banking Solution against Untrusted Computers .....	235
<i>Shujun Li, Ahmad-Reza Sadeghi, Sören Heisrath,     Roland Schmitz, and Junaid Jameel Ahmad</i>	
Certified Lies: Detecting and Defeating Government Interception Attacks against SSL (Short Paper) .....	250
<i>Christopher Soghoian and Sid Stamm</i>	
Proximax: Measurement-Driven Proxy Dissemination (Short Paper) ....	260
<i>Damon McCoy, Jose Andre Morales, and Kirill Levchenko</i>	
BNymble: More Anonymous Blacklisting at Almost No Cost (A Short Paper) .....	268
<i>Peter Lofgren and Nicholas Hopper</i>	

Towards Secure Bioinformatics Services (Short Paper) . . . . .	276
<i>Martin Franz, Björn Deiseroth, Kay Hamacher, Somesh Jha,     Stefen Katzenbeisser, and Heike Schröder</i>	
Quo Vadis? A Study of the Evolution of Input Validation Vulnerabilities in Web Applications . . . . .	284
<i>Theodoor Scholte, Davide Balzarotti, and Engin Kirda</i>	
Re-evaluating the Wisdom of Crowds in Assessing Web Security . . . . .	299
<i>Pern Hui Chia and Svein Johan Knapskog</i>	
Mercury: Recovering Forgotten Passwords Using Personal Devices . . . . .	315
<i>Mohammad Mannan, David Barrera, Carson D. Brown,     David Lie, and Paul C. van Oorschot</i>	
<b>Author Index . . . . .</b>	<b>331</b>

# Collective Exposure: Peer Effects in Voluntary Disclosure of Personal Data

Rainer Böhme<sup>1</sup> and Stefanie Pötzsch<sup>2</sup>

<sup>1</sup> Department of Information Systems, University of Münster, Germany  
[rainer.boehme@wi.uni-muenster.de](mailto:rainer.boehme@wi.uni-muenster.de)

<sup>2</sup> Department of Computer Science, Technische Universität Dresden, Germany  
[stefanie.poetzsch@tu-dresden.de](mailto:stefanie.poetzsch@tu-dresden.de)

**Abstract.** This paper reports empirical evidence for peer effects in privacy behavior using field data from online social lending. Our content analysis and regression models show that individuals copy observable behavior of others in decisions on a) how much to write about oneself, b) whether to share custom pictures, c) what personal data to disclose, and d) how identifiable to present oneself. We frame this finding in the theory of descriptive social norms and analyze moderating effects, such as similarity of context, social proximity, and mimicry of success factors. The presence of peer effects in disclosure behavior can explain the formation and change of apparent social norms and attitudes towards privacy.

## 1 Financial Privacy and Human Behavior

Information technology has created an age of perfect memory, which raises issues of information privacy and informational self-determination. With legal and technical means available that in principle empower individuals to control the distribution of their personal data, researchers of all disciplines still lack a good understanding of *how* individuals make use of this control.

Scholars of the *economics of privacy* (see [1] for a survey) assume rational individuals who consider all costs and benefits when making a decision to disclose personal data. Works in this tradition largely draw on analytical economic models to study the efficiency of privacy regimes in specific market situations [2–4]. Yet it remains doubtful if individual decisions to disclose personal data can be explained sufficiently well with models of rational agents.

Instead, it has been suggested to approach the subject with theories borrowed from *social psychology* [5] and *behavioral economics* [6]. In this spirit, a number of behavioral biases affecting the decision to disclose personal data has recently been identified empirically: a general discrepancy between stated preferences and behavior [7, 8], present-based biases and discounting [9], anchor effects [10], social norms [11], perceived control [12], and contextual primes [13]. All these results have been obtained from laboratory experiments. While experiments are the method of choice for exploring causality under controlled conditions, they oftentimes suffer from small samples and questionable ecological validity.

The contribution of this paper is to complement the laboratory studies with new evidence from field data. More specifically, we explain voluntary disclosure of personal data with *peer effects*, that is, the tendency of individuals to mimic other peoples' disclosure behavior. Our data are loan applications by real users of *Smava.de*, the largest German platform for *online social lending*.

Also known as “P2P lending”, “Ebay for loans”, or “crowd-sourcing of finance”, online social lending has grown rapidly over the past couple of years [14]. Drawing on concepts of (offline) micro-finance, the idea of social lending is to provide a marketplace for unsecured loans between individuals: an online platform lets borrowers advertise loan applications to lenders, who decide in which loan they invest. Each lender funds only a small share of the financed amount so that the credit risk is shared in loan-specific pools of lenders. Lenders receive interest as a compensation for taking risk, whereas the platform operators typically charge fixed (i. e., risk-free) fees. Market mechanisms differ between platforms, a fact that led to research in mechanism design [15].

Online social lending is an ideal data source for the study of behavioral aspects of financial privacy. By their very nature, loan applications contain a lot of personal details, which enable lenders to assess the associated risk [16]. Data requirements and sharing arrangements already raise privacy concerns in the traditional banking industry [17]. These issues are further exacerbated in online social lending where personal data of loan applications does not remain within heavily regulated institutions, but is accessible to all Internet users [18]. Another feature of this data source is that loan applicants disclose their personal data to this audience *voluntarily*. (In fact, financial regulators require the platform operator to collect additional personal data, which is not disclosed to the public though.) This enables us to look for patterns that explain the influence of peers, i. e., other applicants, on the decision to disclose personal data.

There is a clear link between peer orientation and the notion of *herd behavior*. The latter is an active field of research in finance, often entangled with the question of rationality [19]. At some level of sophistication, models can be found that explain herding and resulting bubbles as rational action. For a first cut on the topic of peer effects in personal data disclosure, we spare us the discussion of whether peer effects are rational or not (the former requires a model of competition between borrowers). We rather see our contribution in the description and rigorous measurement of the phenomenon based on longitudinal field data.

This paper is organized straight. The following Section 2 develops seven hypotheses and introduces the data and analysis method. Results are presented in Section 3, and then discussed in Section 4.

## 2 Approach

The design of online social lending websites, including *Smava.de*, was inspired by other online marketplaces. A list of current loan applications is displayed right on the homepage, and details of each application are accessible to everybody by following a single link. Likewise, an archive of completed loan applications—successful and unsuccessful loans alike—is only a few clicks away. Common sense

suggests that new applicants who complete their own loan application seek inspiration for filling the various text fields with information about themselves and their project. Most likely, they will take recent loan application on the platform as examples. This way, we expect to see serial similarities in the longitudinal analysis of all loan applications, which can be interpreted as peer effects.

Aside from common sense, peer effects in disclosure decisions can also be derived from established behavioral theory, notably the influence of descriptive social norms [20]. Following descriptive as opposed to injunctive norms is facilitated if the disclosure decision is made rather heuristically than systematically [21]. The finding that data disclosure is extremely sensitive to contextual cues [13, 22] supports the assumption of heuristic decisions and thus the dominance of descriptive social norms. Moreover, determinants of *self-disclosure* have been studied in social psychology for decades [23], however with focus on relationship building and therapy rather than on the desire or need to protect one's privacy. Studies of self-disclosure typically include disclosure of personal thoughts and feelings, unlike our study, which defines personal data primarily as facts about individuals. These theoretical considerations lead us to the following hypotheses.

## 2.1 Hypotheses

We postulate four hypothesis on the existence of positive peer effects in voluntary disclosure of personal data:

**Hypothesis 1.** *The total lengths of all descriptions associated with a loan application is positively correlated with the lengths of descriptions of recent loan applications.*

**Hypothesis 2.** *A loan application is more likely illustrated with a custom project picture if recent loan applications include a custom project picture.*

**Hypothesis 3.** *The probability of disclosure of personal data of a specific type increases with the disclosure of personal data items of the same type in recent loan applications.*

**Hypothesis 4.** *Borrowers present themselves more identifiable in loan applications if the borrowers in recent loan applications appear more identifiable.*

The hypotheses so far predict that peer effects exist and are observable by using different indicators of disclosure as dependent variable. Length of description (H1) and provision of a custom picture (H2) were chosen as objective indicators. The divulgence of specific personal data items (H3) and the overall identifiability (H4) operationalize our research question better. Testing the latter requires subjective decisions by expert coders in a content analysis (see Sect. 2.2).

In addition, we postulate three hypotheses on factors that moderate the strength of peer effects.

**Hypothesis 5.** *The peer effects predicted in Hypotheses 1–4 are reinforced for recent loan applications which are similar to the newly drafted loan application.*

We test this hypothesis by measuring the additional explanatory power of loan applications in the same category.<sup>1</sup>

**Hypothesis 6.** *The peer effects predicted in Hypotheses 1–4 are reinforced for recent loan applications which share borrower characteristics with the borrower of the newly drafted loan application.*

We test this hypothesis by measuring the additional explanatory power of loan applications of which a) the borrowers' credit grades match and b) borrowers are of the same sex.

**Hypothesis 7.** *The peer effects predicted in Hypotheses 1–2 are reinforced if recent loan applications were successful.*

We test this hypothesis by measuring the additional explanatory power of directly preceding loans which had been completely funded at the time when the new application was drafted.

Hypotheses 5–7 are motivated by different definitions of the peer group. They were formulated to shed more light on the individuals' motivation to select specific loan applications as examples: similarity of context (H5), perceived social-psychologic proximity (H6), and mimicry of apparent success factors (H7).

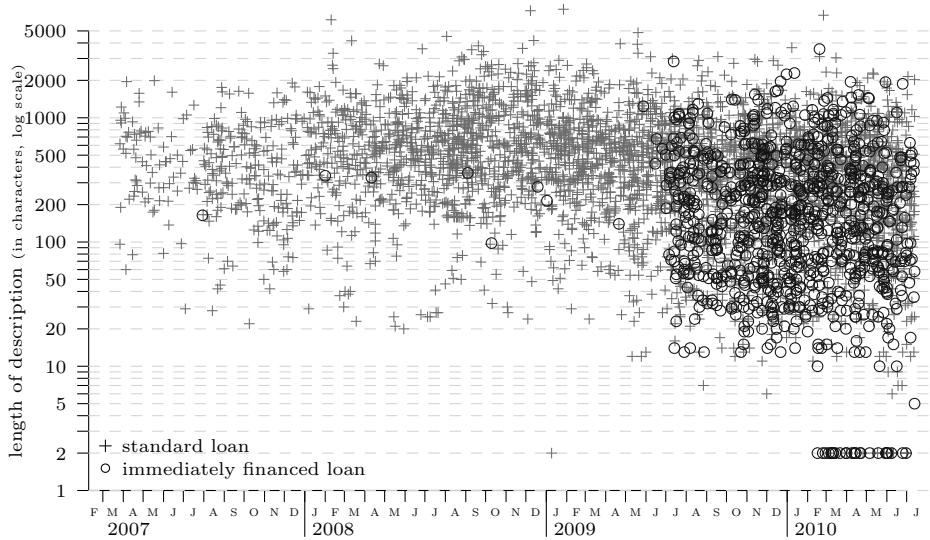
## 2.2 Data

This study is a secondary data analysis in a broader research effort on privacy in online social lending [18, 24]. Our data consists of 4701 loan applications posted on the largest German social lending site *Smava.de* between March 2007 and July 2010, representing a total asked amount of 41.9 million euro (about US\$ 55 million). German borrowers are said to be particularly privacy-aware, reflecting a long tradition of comprehensive data protection regulation as well as high public interest and participation in debates on privacy issues.

*Smava.de* lets potential borrowers propose the basic credit conditions (amount, interest rate, and maturity of 36 or 60 months), checks their identity and publishes on its website verified demographic information (age, gender, state) along with a credit grade, a rough debt service-to-income ratio, an assignment to one of the 19 categories, as well as a user-provided project description and optional user-provided pictures. Lenders can review this information and contribute to its funding in step sizes of 250 euros. When the loan is fully funded or after two weeks, whatever is earlier, the (partial) loan is granted via a commercial bank, who partners with *Smava.de* to comply with the local banking regulations. Borrowers and lenders can appear on the platform under self-chosen nick names, however their full identity is known to and verified by *Smava.de*.

---

<sup>1</sup> *Smava.de* defines the following categories: debt restructuring; liquidity; home, gardening & do-it-yourself; cars & motorbikes; events; education & training; family & education; antiques & art; collection & rarity; electronics; health & lifestyle; sports & leisure; travel; pets & animals; volunteering; commercial; business investment; business extension; miscellaneous.



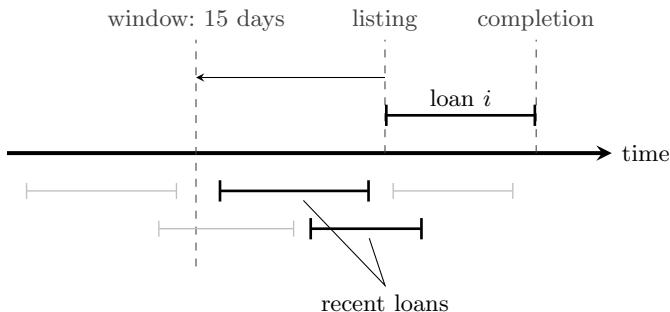
**Fig. 1.** View on the data: variation in the length of the user-provided description for 4701 loan applications published on the largest German social lending platform *Smava.de* between March 2007 and July 2010. Loans which got immediate funding were excluded from the analysis. This reduces the sample size to 3786 loans. Immediate loans picked up in July 2009 after a change in *Smava.de*'s bidding mechanism.

We enrich this data by conducting a content analysis [25] to measure the amount of personal data in loan applications. Variation in personal data disclosure can be found in textual project descriptions, voluntary categories of the borrower profile page, and possibly associated pictures. Three trained coders independently read the textual descriptions and rated the disclosure of personal data without knowing our hypotheses. The underlying code book distinguishes between ten types of personal data, namely borrower's name, financial situation, education, profession, special skills and qualifications, housing situation, health situation, hobbies and memberships, contact details (address, phone, e-mail, etc.), and information about close relatives (family or partner). Each type has several sub-types that encode in which detail borrowers disclose personal data of the respective type.

Orthogonal to the disclosure of facts, privacy can also be measured by identifiability [26]. If individuals are identifiable, it is easier to complete a profile by linking facts from other sources. To measure identifiability, we asked the coders to rate the likelihood that a borrower can be identified on 7-point scales. Individual ratings were collected for several levels of prior knowledge, i. e., presumed identifiability by relatives, neighbors, colleagues or arbitrary persons with access to a search engine. For the purpose of this study, we add these four ratings to an *identifiability index*. This also helps to reduce measurement error. Expectedly, this index correlates with the raw count of disclosed data types, but it is still sufficiently distinct to be interpreted as additional source of information.

### 2.3 Method

Figure 1 shows the length of the user-provided description in all 4701 loan applications between March 2007 and July 2010 over time. The variance in the length of description (measured on a log scale) remains substantial throughout time. So, in principle, there is room to explain part of this variance with peer effects. Note that *Smava.de* changed the market mechanism in July 2009 by introducing so-called “immediate loans”. Instead of waiting for a posted loan application to be funded, the platform suggests an interest rate high enough so that the loan can immediately be financed by lenders who pre-committed offers in a kind of order book. Obviously, voluntary disclosure of personal data does not affect the credit decision for immediate loans. As immediate loans are not distinguishable from other loans in *Smava.de*’s public archive, we use a proxy and exclude from the analysis all loans where the maximum latency between two subsequent bids is less than two minutes. The high density of loans matching this criterion just after the introduction of immediate loans (see Fig. 1) confirms the validity of this procedure.



**Fig. 2.** Definition of *recent loan applications* for loan  $i$ : all other loans *listed* between the beginning of the window and the listing of loan  $i$

Next we have to define the peer group, more specifically, what qualifies *recent loan applications*, as referred to in the hypotheses. For each loan, we determine the specific set of recent loans as depicted in Fig. 2. We take a window of 15 days before the initial listing of the loan and consider all loan applications listed in this window as recent. To test Hypotheses 5–7, the set of recent loans is further reduced to the subset of loans which share a particular property. An exception arises for the property of successful funding (H7). Successful loans can only be more influential if the fact that the loan is successful had been observable at the time when the new application was drafted. Hence we define the subset of successful recent loans as all loans listed within the windows *and* fully funded before the listing of the new loan. For example, even if fully funded, the second (lower) recent loan in Fig. 2 would not be considered as recent successful loan.

The window size was chosen with regard to the expiry of loan applications after two weeks, which sets a lower bound. An upper bound is given by the

number of recent loans in the subsets when testing Hypotheses 5–7. Several unreported robustness checks also revealed that the choice of the window size does not affect the direction of the effects for window sizes between 5 and 60 days. For 15 days, the number of recent loans varies between 0 and 104, following a unimodal distribution with mean 63.2 and median 67. Loans listed later in the sample tend to have a higher number of recent loans due to the growth of the platform. Note that alternative definitions of the peer group are conceivable (e.g., a fixed number of most recent loans), but were not explored so far.

We use regression models to conduct hypothesis tests by statistical inference while controlling for several intervening factors. In general, the models for Hypotheses 1 and 4 are specified as follows:

$$y_i = \beta_0 + \beta_1 \bigcirc_{j \in P_i} y_j + \beta_2 \bigcirc_{j \in P_i \cap \{k | x_k = x_j\}} y_j + \cdots + \beta_3 \log a_i + \beta_4 x_i + \cdots + \beta_{(\cdot)} f(t_i) + \varepsilon_i,$$

where

- $y_i$  is the dependent variable of loan  $i$ ;
- $P_i$  is the set of recent loans of loan  $i$ ;
- $\bigcirc$  is an aggregation operator that calculates the arithmetic mean of the argument over a specified set of loans;
- $x_i$  is an auxiliary property of loan  $i$ , which can appear to build subsets of  $P_i$  to test Hypotheses 5–7. In this case, we also have to include the property as a control variable or—for multinomial properties—as fixed effect to avoid spurious results from an unbalanced sample;
- $a_i$  is the amount of loan  $i$ , for which we have to control as people might disclose more personal data if they ask for more money;
- $f(t_i)$  is a function generating time dummies to control for low frequency fluctuations over time (annual dummies unless otherwise stated);
- $\boldsymbol{\beta} = (\beta_0, \dots)$  is a coefficient vector that can be estimated with ordinary least squares to minimize the squared residuals  $\varepsilon_i$ , i.e.,  $\sum_i \varepsilon_i^2 \rightarrow \min$ .

We estimate several variants of this general model by including terms for different dependent variables  $y$  and auxiliary properties  $x$ . We report the estimated coefficients along with empirical standard errors and a significance level for the two-sided test of the null hypothesis  $\beta_k = 0$ . For the interpretation, estimates  $\hat{\beta}_1$  and  $\hat{\beta}_2$  are of most interest. If  $\hat{\beta}_1$  is positive and significant, the dependent variable can be explained by the aggregate realizations of the dependent variable in the respective sets of recent loans. This indicates the existence of peer effects. If both  $\hat{\beta}_1$  and  $\hat{\beta}_2$  are positive and significant, there is evidence for additional explanatory power of loans in the subset sharing the auxiliary property, hence peer effects are stronger if the loans share a property.

Hypotheses 2 and 3 concern binary indicators as dependent variables. In these cases, we use logistic regression analyses to regress the logit-transformed odds ratio of the dependent variable on the same set of predictor terms as above. These coefficients are estimated with the maximum likelihood method.

### 3 Results

#### 3.1 Length of Description

Table 1 shows the estimated coefficients for the regression models specified to test Hypothesis 1 in conjunction with Hypotheses 5–7. Each specification (A–G) is reported in a column. The dependent variable is given by the log of the number of characters in all text field descriptions provided by the loan applicant. For brevity, we do not report all coefficients for multinomial controls and fixed effects, but indicate whether the respective terms are included by the label “yes”.

Model A estimates a plain peer effect. The term for all recent loans (i. e.,  $\hat{\beta}_1$ ) is positive and highly significant. With a coefficient value of almost 0.9, a change by one order of magnitude in the lengths of description of recent loans translates to about 2.5 times longer descriptions for the average new loan application. Most likely, this specification overestimates the size of the peer effect because other relevant predictors of the verbosity are not controlled for. Model B resolves this by introducing a control for the loan amount (log transformed) and time dummies to capture long-term trends. The relevant coefficient remains positive and highly significant. Therefore our data supports Hypothesis 1. The coefficient for the loan amount is positive and significant, too. People tend to write more if they ask for more money.

Models C–F test Hypotheses 5–7. We do find support for Hypothesis 5. Recent loans of the same category have additional explanatory power in predicting the length of description even if base effects of individual categories are controlled for (model C). The picture is more mixed for Hypothesis 6, which can only be supported if the same credit grade is taken as indicator of social proximity (model D). Loan applicants apparently do not prefer peers of the same sex when mimicking their disclosure behavior (model E). This is most likely not an artifact of an unbalanced sample, as about one quarter of all loan are requested by women. Nor do we find support for Hypothesis 7: borrowers do not seem to copy from successful recent applications more often than from pending or unsuccessful applications (model F). Model G serves as robustness check to see if any coefficient changes its sign when all predictors are included at the same time. This does not happen, which indicates robustness. However, some estimates lose statistical significance presumably because of collinearity between the higher number of predictors.

Overall, the models explain between 8 and 14 % of the variance in the dependent variable. Note that the bulk of explained variance can be attributed to the direct peer effects (model A) and not, unlike in many other field studies, to the inclusion of fixed effects. The number of cases varies somewhat between models because missing values appear if subsets of recent loans turn out to be empty.

#### 3.2 Provision of a Picture

Table 2 shows the estimated coefficients for the logistic regression models specified to test Hypothesis 2 in conjunction with Hypotheses 5–7. We take the odds

**Table 1.** Factors influencing the verbosity of descriptions

Terms	Dependent variable: length of description (log)						
	A	B	C	D	E	F	G
all recent loans	0.88 *** (0.047)	0.63 *** (0.078)	0.53 *** (0.088)	0.53 *** (0.087)	0.60 *** (0.126)	0.58 ** (0.220)	0.35 (0.245)
<i>recent loans with ...</i>							
same category			0.08 * (0.038)				0.07 † (0.038)
same credit grade				0.12 ** (0.041)			0.15 *** (0.042)
borrower same sex					0.03 (0.100)		0.00 (0.110)
complete funding						0.06 (0.203)	0.06 (0.224)
<i>controls:</i>							
log amount		0.21 *** (0.021)	0.18 *** (0.023)	0.21 *** (0.022)	0.21 *** (0.021)	0.21 *** (0.022)	0.19 *** (0.024)
<i>fixed effects:</i>							
category			yes				yes
credit grade				yes			yes
gender					yes		yes
complete funding						yes	yes
annual	yes	yes	yes	yes	yes	yes	yes
(constant)	0.67 * (0.277)	0.49 (0.497)	0.75 (0.521)	0.24 (0.504)	0.42 (0.499)	0.37 (0.518)	0.47 (0.550)
(number of cases)	3784	3784	3553	3750	3777	3780	3531
(adjusted $R^2$ [%])	8.5	11.1	12.9	11.5	11.1	11.0	13.4

Std. errors in brackets; stat. significance: † $p < 0.1$ , \* $p < 0.05$ , \*\* $p < 0.01$ , \*\*\* $p < 0.001$

that a loan applicant has uploaded a custom picture as dependent variable. Overall, 22.6 % of all loan applications contain a custom project picture.

Model H tests Hypothesis 2 and finds highly significant peer effects. The interpretation of coefficients is less straightforward for logistic models. A value of  $\hat{\beta}_1 = 2.3$  denotes that the odds of providing a custom picture are  $\exp(\hat{\beta}_1) \approx 10$  times higher if *all* recent loans contained a custom picture than if *none* of the recent loans contained one. Model I finds weakly significant support for the strict test of Hypothesis 5 including category fixed effects. The effect remains robust and (unsurprisingly) stronger if the category fixed effects are omitted (model J). This demonstrates the importance of including the respective fixed effects, as different offsets between categories (e.g., positive sign for cars & motorcycles; negative sign for debt restructuring) otherwise feed into the “same category” terms and overstate the true effect size. Models K–M test Hypotheses 6 and 7 and do not find support for any of them.

The provision of a custom picture is probably the crudest indicators for several reasons. Its binary scale is susceptible to noise. The provisioning of a picture largely depends on the external factor whether a suitable picture is available. And the indicator does not differentiated between pictures of different information

**Table 2.** Factors influencing the decision to publish a project picture

Terms	Dependent variable: prob. of custom picture (logit link)						
	H	I	J	K	L	M	N
all recent loans	2.27 *** (0.421)	1.90 *** (0.530)	1.48 ** (0.519)	2.20 *** (0.505)	1.80 ** (0.669)	2.36 * (1.099)	2.36 † (1.357)
<i>recent loans with ...</i>							
same category		0.34 † (0.207)	0.82 *** (0.195)				0.34 (0.209)
same credit grade				-0.07 (0.237)			0.01 (0.253)
borrower same sex					0.30 (0.486)		0.18 (0.554)
complete funding						-0.31 (0.998)	-0.70 (1.202)
<i>controls:</i>							
log amount	0.14 ** (0.050)	0.17 ** (0.055)	0.14 ** (0.052)	0.15 ** (0.050)	0.14 ** (0.050)	0.15 ** (0.052)	0.19 ** (0.058)
<i>fixed effects:</i>							
category		yes					yes
credit grade				yes			yes
gender					yes		yes
complete funding						yes	yes
annual	yes	yes	yes	yes	yes	yes	yes
(constant)	-2.94 *** (0.447)	-3.43 *** (0.508)	-2.98 *** (0.473)	-3.25 *** (0.473)	-2.95 *** (0.450)	-3.01 *** (0.497)	-3.91 *** (0.592)
(number of cases)	3784	3553	3553	3750	3777	3780	3531

Std. errors in brackets; stat. significance: † $p < 0.1$ , \* $p < 0.05$ , \*\* $p < 0.01$ , \*\*\* $p < 0.001$

content and quality. Nevertheless we decided to report the results because they combine an objective indicator with the same logistic regression model that is used in the following to estimate peer effects for personal data disclosure. It therefore serves as reference to better interpret the results in Table 3.

### 3.3 Personal Data Disclosure by Type

Length of description and provision of a picture are imprecise indicators because they do not measure any semantic. A high value could either reflect verbosity or extensive disclosure of personal data. Our hand-coded indicators of data disclosure by type do not share this limitation. Due to resource constraints, data of this granularity is only available for 1558 fully funded loans between November 2008 and January 2010. Since partly funded and unsuccessful loans were not included in the coding task, we are unable to test Hypothesis 7 for subjective indicators. This is not a big shortcoming as this hypothesis has been refuted on the basis of objective indicators anyway.

Table 3 shows selected estimated coefficients for the logistic regression models specified to test Hypothesis 3 in conjunction with Hypotheses 5 and 6. Unlike in the previous tables, predictors appear in columns and different dependent

**Table 3.** Strength of peer effects on disclosure of personal data (by type)

Type of personal data	Overall frequency	Frequency in recent loans			
		all	category	credit grade	amount
Profession	62.3 %	2.76 *** (0.641)	0.18 (0.277)	0.73 * (0.307)	0.28 *** (0.080)
Financial situation	33.9 %	3.19 *** (0.577)	0.31 (0.273)	0.75 * (0.311)	0.20 * (0.083)
Family and partner	31.4 %	-0.13 (0.968)	-0.29 (0.316)	0.79 * (0.309)	0.11 (0.084)
Hobbies and memberships	29.3 %	2.62 *** (0.713)	0.13 (0.271)	0.22 (0.297)	0.11 (0.082)
Housing situation	19.1 %	1.00 (1.568)	0.26 (0.504)	-0.65 (0.518)	0.34 ** (0.107)
Education	10.8 %	1.68 (1.987)	0.21 (0.474)	0.34 (0.662)	-0.18 (0.122)
Name or first name	5.7 %	8.76 ** (2.721)	-0.57 (0.956)	0.55 (1.020)	0.25 (0.161)
Health situation	3.5 %	18.14 *** (4.590)	-1.66 (1.099)	-1.61 (2.416)	-0.11 (0.230)
Contact details	2.6 %	8.29 (6.773)	1.35 (1.159)	-1.22 (2.282)	0.20 (0.234)
Special skills & qualifications	1.9 %	17.69 ** (6.664)	-1.70 (2.475)	0.81 (2.480)	0.29 (0.300)

Std. errors in brackets; stat. significance:  $\dagger p < 0.1$ ,  $*$   $p < 0.05$ ,  $** p < 0.01$ ,  $*** p < 0.001$ . Summary of logit models for the probability of disclosing at least one item of the respective type.  $N = 1558$  fully funded loans between Nov 2008 and Jan 2010.

variables in rows. For each row, the dependent variable is defined by the odds that a loan application contains at least one data item of a specific type (including sub-types). The types are ordered by decreasing marginal probability, as reported in the second column.

Positive and highly significant peer effects can be found for 6 out of 10 types of personal data. Quite surprisingly, while Hypothesis 5 could be retained for objective indicators, it seems that recent loans with the same category have no additional impact on the decision to disclose data of a particular type. Therefore we refute Hypothesis 5 for this indicator. By contrast, recent loan applications with the same credit grade have significant influence in predicting the disclosure of personal data of the three types with the highest marginal probability of disclosure. With 3 out of 10 significant at the 5% level—1 out of 20 is the expected value under the null hypothesis—we have to acknowledge weak support for Hypothesis 6. Note that we have also estimated models with matching gender, but none of the relevant coefficients was significant.

A side-observation in Table 3 is that the disclosure of certain types of personal data, notably data about profession, financial and housing situation, correlated positively with the loan amount, whereas others types of data appear independent of the size of the loan.

**Table 4.** Factors influencing the identifiability of a borrower

Terms	Dependent variable: identifiability index			
	O	P	Q	R
all recent loans	0.61 *** (0.127)	0.56 *** (0.143)	0.48 *** (0.141)	0.37 (0.252)
<i>recent loans with ...</i>				
same category		0.00 (0.060)		0.01 (0.060)
same credit grade			0.13 * (0.060)	0.13 * (0.061)
borrower same sex				0.07 (0.200)
<i>controls:</i>				
log amount	0.85 *** (0.205)	0.51 * (0.228)	0.93 *** (0.207)	0.59 * (0.232)
<i>fixed effects:</i>				
category		yes		yes
credit grade			yes	yes
gender				yes
annual	yes	yes	yes	yes
(constant)	-3.64 (2.220)	-0.77 (2.404)	-4.67 * (2.297)	-1.49 (2.501)
(number of cases)	1659	1565	1651	1554
(adjusted $R^2$ [%])	4.2	3.9	4.3	4.2

### 3.4 Identifiability

For a sample of 1663 cases, we have sufficient observations of the identifiability index (see Sect. 2.2). Table 4 shows the estimated coefficients for the regression models specified to test Hypothesis 4 in conjunction with Hypotheses 5 and 6. Again, we find highly significant evidence for peer effects (model O). This supports Hypothesis 4. As in Sect. 3.3 above, Hypothesis 5 is not supported (model P), whereas Hypothesis 6 can be retained at least for the credit grade as matching property (model Q). Compared to the length of description as dependent variable (see Tab. 1), the ratio of explained variance is lower, but this is not uncommon for noisy measurements from field data that exhibit a lot of unexplained heterogeneity over an extended period of data collection.

## 4 Discussion

### 4.1 Summary and Interpretation

To the best of our knowledge, this paper is the first to report evidence for peer effects in voluntary disclosure of personal data using field data. The existence of this effect has been conjectured before [6, 26], however so far without evidence. Table 5 summarizes the results for all our hypotheses.

**Table 5.** Summary of hypothesis tests

		plain	in conjunction with ...		
			Hypothesis 5	Hypothesis 6	Hypothesis 7
Hypothesis 1	retained	retained	partly retained	refuted	
Hypothesis 2	retained	weak support	refuted	refuted	
Hypothesis 3	partly retained	refuted	weak support	not tested	
Hypothesis 4	retained	refuted	partly retained	not tested	

More specifically, we could find plain peer effects for all four indicators of disclosure—objective proxies, such as the length of description or the provision of custom pictures, and subjective ratings alike. The result pattern is less clear for our additional hypotheses on factors that potentially reinforce peer effects and it is probably too early to generalize from the specific operationalizations made in the context of our data source.

Peer effects imply that decisions to disclose personal data are driven by observed behavior of others. Understanding peer effects is relevant in general, because over time, peer effects can create self-reinforcing dynamics that may change the social attitude towards privacy: “if everybody discloses his or her full life on the Internet, it can’t be wrong to disclose my information as well.” Privacy-friendly interface design and increased user awareness might attenuate this dynamic, but we remain skeptical if those cues can ever be strong enough to reverse dynamics of descriptive social norms, not to mention the missing incentives of market participants to implement effective cues.

A specific relevance of peer effects in social lending emerges for platform designers in the financial industry. If data disclosure is merely a reaction to previous loans, then the fact whether data has been disclosed loses its value as a signal to lenders. A platform providing a better separation of roles, with distinct privileges to access loan applications (e.g., exclusively to registered lenders with positive account balance), could not only increase borrower privacy, but also make data disclosure decisions more individual and thus signal more valuable information to lenders. This might resolve the apparent puzzle that more disclosure does not always translate into better credit conditions, even after controlling for all available hard information [18, 24].

## 4.2 Related Work

Aside from the theoretical and empirical works referenced in the introduction to narrow down the research question, the following empirical studies are related most closely to this paper. Gross and Acquisti [26] use field data to measure personal data disclosure and identifiability in online social networks. Their approach is quantitative, but largely explorative with emphasis on descriptive statistics. The authors speculate about peer pressure and herding in their conclusions.

Acquisti, John, and Loewenstein [11] report positive influence of descriptive social norms on disclosure of *sensitive* personal data. In their laboratory experiments, participants reacted differently if they were told that previous participants revealed sensitive data. In contrast to the case of online social lending, the subjects had to answer closed-form questions and could not see the disclosed data of others. They had to believe in the abstract information instead. Barak and Gluck-Ofri [27] conducted a self-disclosure experiment in the framework of an online forum. They report peer effects from previous posts. In contrast to our study, the dependent variable is less tailored to privacy. Consistent with the tradition in this literature, the authors also analyze disclosure of feelings and thoughts.

Empirical research on online social lending also exists in the economic literature (see [14] for a review). Typical research questions concern market efficiency, signs of discrimination, or the influence of ties in the social networks between borrowers and lenders, respectively. Occasionally, lengths of description is used as an effort measure in these studies [28]. We are not aware of other attempts than our own previous works [18, 24] to quantify personal data disclosure and test theories of privacy behavior empirically with data from online social lending.

### 4.3 Limitations and Future Work

Compared to laboratory experiments, field studies suffer from limited control. In particular, disclosure of false information remains unnoticed in this study (a problem shared with some experiments, e.g., [11]). Our main results are robust to changes in the window size and the use of quarterly instead of annual time dummies, but refinements are needed to check the robustness against different definitions of the peer group. New research efforts are directed to replicate the study in other contexts. A more distant goal is to follow the approach in [29] and unify the collection of individual behavioral effects of data disclosure into a more general theory of planned behavior, which spans effects of attitude, social norms, and self-efficacy.

**Acknowledgements.** This paper has benefited from discussions with Jens Schade and Nicola Jentzsch. We thank our coders Ulrike Jani, Kristina Schäfer, and Michael Sauer for their excellent research assistance.

## References

1. Hui, K.L., Png, I.P.: The economics of privacy. In: Hendershott, T.J. (ed.) *Handbooks in Information System and Economics*, vol. 1, pp. 471–493. Elsevier (2006)
2. Posner, R.A.: The economics of privacy. *American Economic Review* 71(2), 405–409 (1981)
3. Calzolari, G., Pavan, A.: On the optimality of privacy in sequential contracting. *Journal of Economic Theory* 130(1), 168–204 (2006)
4. Jentzsch, N.: A welfare analysis of secondary use of personal data. In: *Workshop on the Economics of Information Security (WEIS)*, Harvard University (2010)
5. Margulis, S.T.: Privacy as a social issue and behavioral concept. *Journal of Social Issues* 59(2), 243–261 (2003)
6. Acquisti, A., Grossklags, J.: Privacy and rationality in individual decision making. *IEEE Security and Privacy* 3(1), 26–33 (2005)

7. Berendt, B., Günther, O., Spiekermann, S.: Privacy in e-commerce: Stated preferences vs. actual behavior. *Communications of the ACM* 48(4), 101–106 (2005)
8. Norberg, P.A., Horne, D.R., Horne, D.A.: The privacy paradox: Personal information disclosure intentions versus behaviors. *Journal of Consumer Affairs* 41(1), 100–126 (2007)
9. Acquisti, A.: Privacy in electronic commerce and the economics of immediate gratification. In: *Electronic Commerce (ACM EC)*, New York, NY, pp. 21–29 (2004)
10. Grossklags, J., Acquisti, A.: When 25 cents is too much: An experiment on willingness-to-sell and willingness-to-protect personal information. In: *Workshop on the Economics of Information Security (WEIS)*, Carnegie Mellon University (2007)
11. Acquisti, A., John, L., Loewenstein, G.: The impact of relative standards on concern about privacy. In: *Workshop on the Economics of Information Security (WEIS)*, University College London (2009)
12. Brandimarte, L., Acquisti, A., Loewenstein, G.: Misplaced confidences: Privacy and the control paradox. In: *Workshop on the Economics of Information Security (WEIS)*, Harvard University (2010)
13. John, L., Acquisti, A., Loewenstein, G.: Strangers on a plane: Context-dependent willingness to divulge personal information. *Consumer Research* 37(2) (2011)
14. Berger, S., Gleisner, F.: Emergence of financial intermediaries in electronic markets: The case of online P2P lending. *BuR – Business Research* 2(1), 39–65 (2009)
15. Chen, N., Ghosh, A., Lambert, N.: Social lending. In: *Electronic Commerce (ACM EC)*, Palo Alto, CA, pp. 335–344. ACM Press (2009)
16. Stiglitz, J.E.: Credit rationing in markets with imperfect information. *American Economic Review* 71(3), 393–410 (1981)
17. Jentzsch, N.: *Financial Privacy*. Springer, Heidelberg (2007)
18. Böhme, R., Pötzsch, S.: Privacy in online social lending. In: *AAAI Spring Symposium on Intelligent Privacy Management*, Palo Alto, CA, pp. 23–28 (2010)
19. Bikhchandani, S., Sharma, S.: Herd behavior in financial markets. *IMF Staff Papers* 47(3), 279–310 (2001)
20. Cialdini, R.B., Kallgren, C.A., Reno, R.R.: A focus theory of normative conduct. *Advances in Experimental Social Psychology* 24, 201–234 (1991)
21. Chaiken, S., Yaacov, T.: *Dual-process Theories in Social Psychology*. Guilford Press, New York (1999)
22. Böhme, R., Köpsell, S.: Trained to accept? A field experiment on consent dialogs. In: *Human Factors in Computing Systems (ACM CHI)*, pp. 2403–2406 (2010)
23. Derlega, V.J., Berg, J.H. (eds.): *Self-Disclosure. Theory, Research, and Therapy*. Plenum Press, New York (1987)
24. Pötzsch, S., Böhme, R.: The Role of Soft Information in Trust Building: Evidence from Online Social Lending. In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) *TRUST 2010. LNCS*, vol. 6101, pp. 381–395. Springer, Heidelberg (2010)
25. Holsti, O.R.: Content analysis for the social sciences and humanities. Addison-Wesley (1969)
26. Gross, R., Acquisti, A.: Information revelation and privacy in online social networks. In: *Privacy in the Electronic Society (ACM WPES)*, pp. 71–80 (2005)
27. Barak, A., Gluck-Ofri, O.: Degree and reciprocity of self-disclosure in online forums. *CyberPsychology & Behavior* 10(3), 407–417 (2007)
28. Herzenstein, M., Andrews, R., Dholakia, U., Lyandres, E.: The democratization of personal consumer loans? Determinants of Success in Online Peer-to-Peer Lending Communities (June 2008)
29. Yao, M.Z., Linz, D.G.: Predicting self-protections of online privacy. *CyberPsychology & Behavior* 11(5), 615–617 (2008)

# It's All about the Benjamins: An Empirical Study on Incentivizing Users to Ignore Security Advice

Nicolas Christin<sup>1</sup>, Serge Egelman<sup>2</sup>, Timothy Vidas<sup>3</sup>, and Jens Grossklags<sup>4</sup>

<sup>1</sup> INI/CyLab, Carnegie Mellon University

<sup>2</sup> National Institute of Standards and Technology

<sup>3</sup> ECE/CyLab, Carnegie Mellon University

<sup>4</sup> IST, Pennsylvania State University

nicolasc@andrew.cmu.edu

serge.egelman@nist.gov

tvidas@ece.cmu.edu

jensg@ist.psu.edu

**Abstract.** We examine the cost for an attacker to pay users to execute arbitrary code—potentially malware. We asked users at home to download and run an executable we wrote without being told what it did and without any way of knowing it was harmless. Each week, we increased the payment amount. Our goal was to examine whether users would ignore common security advice—not to run untrusted executables—if there was a direct incentive, and how much this incentive would need to be. We observed that for payments as low as \$0.01, 22% of the people who viewed the task ultimately ran our executable. Once increased to \$1.00, this proportion increased to 43%. We show that as the price increased, more and more users who understood the risks ultimately ran the code. We conclude that users are generally unopposed to running programs of unknown provenance, so long as their incentives exceed their inconvenience.

**Keywords:** Behavioral Economics, Online Crime, Human Experiments.

## 1 Introduction

Since the early 2000s, Internet criminals have become increasingly financially motivated [22]. Of particular concern is the large number of “bots,” that is, compromised end user machines that can be accessed and controlled remotely by these miscreants. Bots are a security concern, because they are federated in centrally-operated networks (“botnets”) that can carry out various kinds of attacks (e.g., phishing site hosting, spam campaigns, distributed denial of service, etc.) on a large scale with relative ease. Botnet operators usually do not launch these attacks themselves, but instead rent out computing cycles on their bots to other miscreants.

Estimates on the total number of bots on the Internet range, for the year 2009, from 6 to 24 million unique hosts [31, 4]. Individual botnets can be comprised of hundreds of thousands of hosts. For instance, researchers managed to hijack the Torpig botnet for a period of ten days, and observed on the order of 180,000 new infections over that period, and 1.2 million unique IP addresses contacting the control servers [30].

The scale of the problem, however alarming it may seem, suggests that most Internet users whose computers have been turned into bots either do not seem to notice they have been compromised, or are unable or unwilling to take corrective measures, potentially because the cost of taking action outweighs the perceived benefits. There is evidence (see, e.g., [29]) of hosts infected by multiple pieces of malware, which indicates that a large number of users are not willing to undertake a complete reinstallation of their system until it ceases to be usable. In other words, many users seem to be content ignoring possible security compromises as long as the compromised state does not noticeably impact the performance of the machine.

Consequently, bots are likely to be unreliable. For instance, they may frequently crash due to multiple infections of poorly programmed malware. Their economic value to the miscreants using them is in turn fairly low; and indeed, advertised bot prices can be as low as \$0.03-\$0.04 per bot,<sup>1</sup> according to a recent Symantec report [31].

Overall, bot markets are an interesting economic environment: goods are seemingly of low quality, and treachery amongst sellers and buyers is likely rampant [14], as transaction participants are all engaging in illegal commerce. Yet, the large number of bots, and the absence of notable decrease in this number, seem to indicate that bot markets remain relatively thriving. This puzzle is even more complicated by the fact most surveyed Internet users profess a strong desire to have secure systems [5].

In this paper, we demonstrate that, far from being consistent with their stated preferences, in practice, users actually do not attach any significant economic value to the security of their systems. While ignorance could explain this state of affairs, we show that the reality is much worse, as some users readily turn a blind eye to questionable activities occurring on their systems, as long as they can themselves make a modest profit out of it.

We describe an experiment that we conducted using Amazon’s Mechanical Turk, where we asked users to download and run our “Distributed Computing Client” for one hour, with little explanation as to what the software actually did,<sup>2</sup> in exchange for a nominal payment, ranging from \$0.01 to \$1.00. We conducted follow-up surveys with the users who ran the software to better understand the demographics of at-risk populations. We use the results of this experiment to answer the following questions:

- **How much do Internet users value access to their machines?** If users who are at risk of getting infected could be compensated for the infection, how much should the compensation be?
- **Is there a specific population at risk?** Do we observe significant demographic skew for certain types of populations in our study?
- **Are current mitigations effective at dissuading users from running arbitrary programs that they download from the Internet?** When software warns users about the dangers of running unknown programs, are they any less likely to continue running those programs?

---

<sup>1</sup> In this entire paper, all prices are given in US dollars. That is, the “\$” sign denotes US currency.

<sup>2</sup> As discussed in Section 3, in reality, the software merely collected running process information and other system statistics, ran down a countdown timer, and reported this information back to us; in our experimental condition, it also prompted users for administrative access to run.

Answering these questions is important because it allows us to quantify how risk-seeking people may be, which in turn may help optimize user education and other mitigation strategies. These answers also provide an idea on the upper bound of the value of a bot, and permit us to compare prices set in actual, realized transactions, to advertised prices [14]. Being able to dimension such prices helps us better understand which intervention strategies are likely to be successful; for instance, if a security mitigation costs users one hour of work, while they value access to their machine at a few cents, users are unlikely to adopt the security mitigation [13].

## 2 Related Work

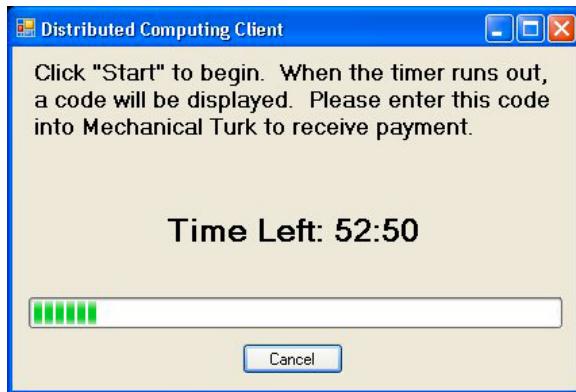
There has recently been a considerable amount of academic research devoted to trying to dimension economic parameters in the realm of online crime. Pioneering works in that realm mostly attempted to measure prices of illegitimately acquired commodities advertised in underground online markets [8, 32]. Industry research is also very active in that field. For instance, Symantec and Verizon issue yearly reports that dimension key economic indicators of the vitality of online criminal activity (see, e.g., [31, 6]).

Most of this research, though, relies on passive measurements; advertising channels are monitored, but researchers rarely become party to an actual transaction. As such, the prices monitored and reported may be quite different from the actual sales prices [14]. A few recent papers try to obtain more precise measurements by taking over (or infiltrating) botnets and directly observing, or participating in the transactions [18, 30].

To complicate matters, the reported value of botnets also varies based on perspective. From a prosecution perspective, costs of crimes may be inflated to seek larger sentences. From a botnet operator perspective, risks associated with operation may warrant hosting in countries that have ambiguous or non-existent cybercrime laws. It is the botnet renter's perspective that we explore. A US congressional report estimated renting an entire botnet in 2005 cost \$200-\$300 per hour [34]. Renting a bot in 2006 reportedly cost \$3-\$6 [21]. In 2007 studies show an asking cost as high as \$2-\$10 [8], but the actual price paid have been as low as \$0.25 [35]. Recently, prices have been discerned to cost \$50 for several thousand bots for a 24-hour period [24].

Other work aims at uncovering relationships between different online crime actors, and calculating economic metrics (e.g., actors, revenue, market volume, etc.) from network measurements. For instance, Christin et al. [7] gathered four years worth of data on an online scam prevalent in Japan, and show that the top eight groups are responsible for more than half of the malicious activity, and that considerable profit (on the order of \$100,000 per year) can be amassed for relatively little risk (comparatively small fines, and very light prison sentences). Likewise, Moore and Edelman [23] assessed the economic value of typosquatting domains by gathering a large number of them and evidence some disincentives for advertisement networks to intervene forcefully.

Closer to the research on which we report in this paper, Grossklags and Acquisti provide quantitative estimates of how much people value their privacy. They show that most people are likely to give up considerable private information in exchange for \$0.25 [11]. Similarly, Good et al. conducted a laboratory study wherein they observed whether participants would install potentially risky software with little functional benefit [10].



**Fig. 1. Screenshot of the “Distributed Computing Client.”** After subjects clicked a start button, a timer ran for one hour. Once the timer expired, a completion code was displayed. Subjects entered this completion code into Mechanical Turk in order to receive payment.

Participants overwhelmingly decided to affirmatively install peer-to-peer and desktop-enhancing software bundled with spyware. Most individuals only showed regret about their actions when presented with much more explicit notice and consent documents in a debriefing session. To most participants, the value extracted from the software (access to free music or screen savers) trumped the potentially dangerous security compromises and privacy invasions they had facilitated.

We complement these works by trying to provide a precise measurement of how little it could cost an attacker to have unfettered access to a system with the implicit consent of the owner. We also contribute to a better understanding of behavioral inconsistencies between users’ stated security preferences [5] and their observed actions.

### 3 Experimental Methodology

The key part of our study lies in our experiment, which we ran using Amazon’s Mechanical Turk. The objective was to assess at what price, and under which conditions, would people install an unknown application. In this section, we describe the study environment and two different conditions we created. We analyze our results in Section 4.

Mechanical Turk is a marketplace for pairing “workers” with “requesters.” Requesters post tasks that are simple for humans to complete, but difficult to automate (e.g., image tagging, audio transcription, etc.). In return, the requesters pay workers micropayments—on the order of a few cents—for successfully completing a task. However, recently researchers have begun using it as a crowd-sourcing tool for performing studies on human subjects [19]. This is because the demographics of Mechanical Turk do not significantly differ from the demographics of worldwide Internet users [27]. But what about the quality of the resulting data? To collect data for a 2008 study, Jakobsson posted a survey to Mechanical Turk while also hiring a market research firm. He concluded that the data from Mechanical Turk was of the same quality, if not higher [16].

In September of 2010, we created a Mechanical Turk task offering workers the opportunity to “get paid to do nothing.” Only after accepting our task did participants see a detailed description: they would be participating in a research study on the “CMU Distributed Computing Project,” a fictitious project that we created. As part of this, we instructed participants to download a program and run it for an hour (Figure 1). We did not say what the application did. After an hour elapsed, the program displayed a code, which participants could submit to Mechanical Turk in order to claim their payment.

Because this study involved human subjects, we required Institutional Review Board (IRB) approval. We could have received a waiver of consent so that we would not be required to inform participants that they were participating in a research study. However, we were curious if—due to the pervasiveness of research tasks on Mechanical Turk—telling participants that this was indeed a research task would be an effective recruitment strategy. Thus, all participants were required to click through a consent form. Beyond the consent form, there was no evidence that they were participating in a research study; all data collection and downloads came from a third-party privately-registered domain, and the task was posted from a personal Mechanical Turk account not linked to an institutional address. No mention of the “CMU Distributed Computing Project” appeared on any CMU websites. Thus, it was completely possible that an adversary had posted a task to trick users into downloading malware under the guise of participating in a research study, using a generic consent form and fictitious project names in furtherance of the ruse.

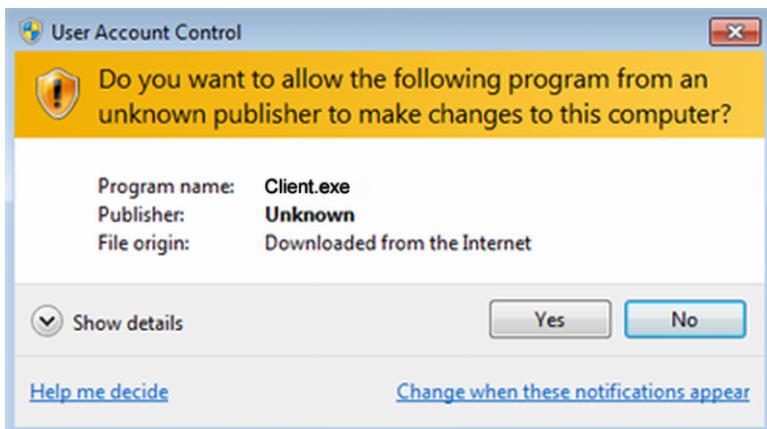
We reposted the task to Mechanical Turk every week for five weeks. We increased the price each subsequent week to examine how our results changed based on the offering price. Thus, the first week we paid participants \$0.01, the second week \$0.05, the third week \$0.10, the fourth week \$0.50, and the fifth week \$1.00. In order to preserve data independence and the between-subjects nature of our experiment, we informed participants that they could not participate more than once. We enforced this by rejecting results from participants who had already been compensated in a previous week.

We further restricted our task to users of Microsoft Windows XP or later (i.e., XP, Vista, or 7).<sup>3</sup> In Windows Vista and later, a specific security mitigation is included to dissuade users from executing programs that require administrator-level access: User Account Control (UAC). When a program is executed that requires such access, a prompt is displayed to inform the user that such access is requested, and asks the user if such access should be allowed (Figure 2). To examine the effectiveness of this warning, we created a second between-subjects condition. For each download, there was a 50% chance that the participant would download a version of our software that requested administrator-level access via the application manifest, which in turn would cause the UAC warning to be displayed prior to execution.

The main purpose of this experiment was to collect data on the ratio of people who downloaded the application versus viewed the task, and the ratio of people who ran the application versus downloaded the application, as well as how these ratios changed

---

<sup>3</sup> Certain commercial software is identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the software identified is necessarily the best available for the purpose.



**Fig. 2. UAC prompt.** Users of Windows Vista or 7 saw this prompt in the experimental condition.

based on the payment amount and presence of the UAC warning. Finally, we collected various (anonymous) data from the systems of the participants who ran the application. We designed our application to report anonymous statistics using port 123, the port normally reserved for the Network Time Protocol, since we reasoned that outgoing connections on this port were least likely to be firewalled (beyond the HTTP ports, which were already in use on our server). We specifically collected the following information:

- **Experimental Condition**— The application reported its version so we could examine if participants were disproportionately executing the application in either the experimental (UAC) or control condition.
- **Version**— We collected Windows versions to examine whether participants were exposed to the UAC prompt (i.e., XP users who executed the application in the experimental condition were not exposed to the prompt because the UAC feature did not exist in their version). Additionally, this also gave us insights into how up-to-date their software was (i.e., whether they had the latest service packs installed).
- **Process List**— We collected a list of the image names for all running processes to examine whether participants were taking any security precautions. Specifically, we were looking for known anti-virus (AV) software, as well as indications that they were running the software from within a virtual machine (VM).
- **Virtual Machine Detection**— To further detect whether our application was being executed from within a virtual machine (VM), we performed two heuristics. First, we used the Red Pill VM-detection routine to report the memory address of the Interrupt Descriptor Table (IDT) and number of CPUs [28]. Second, we reported the name of the motherboard manufacturer. Some VM software will report a generic manufacturer rather than forwarding this information from the host operating system. Obviously this is an incomplete list of VM-detection techniques, but we believe these heuristics allowed us to show a lower bound on VM usage.

Other than collecting this information, our program did absolutely nothing other than running a timer and periodically reporting to our server that it was still running.

**Table 1.** Experimental results showing the Mechanical Turk users who viewed the task; of those, the fraction who downloaded the program; and of those, the fraction who ran the program. The latter represents only those who submitted data to our server, and therefore is a lower bound.

	\$0.01	\$0.05	\$0.10	\$0.50	\$1.00
<i>viewed task</i>	291	272	363	823	1,105
<i>downloaded</i>	141 49%	135 50%	190 52%	510 62%	738 67%
<i>ran</i>	64 45%	60 44%	73 38%	294 58%	474 64%

After participants submitted valid payment codes at the end of the experiment, we invited them to complete an exit survey for a \$0.50 bonus payment. This bonus payment was identical across all experimental conditions. This survey was designed to answer questions about the participants' risk perceptions during and outside of the study, what security precautions they normally take, and various demographic information.

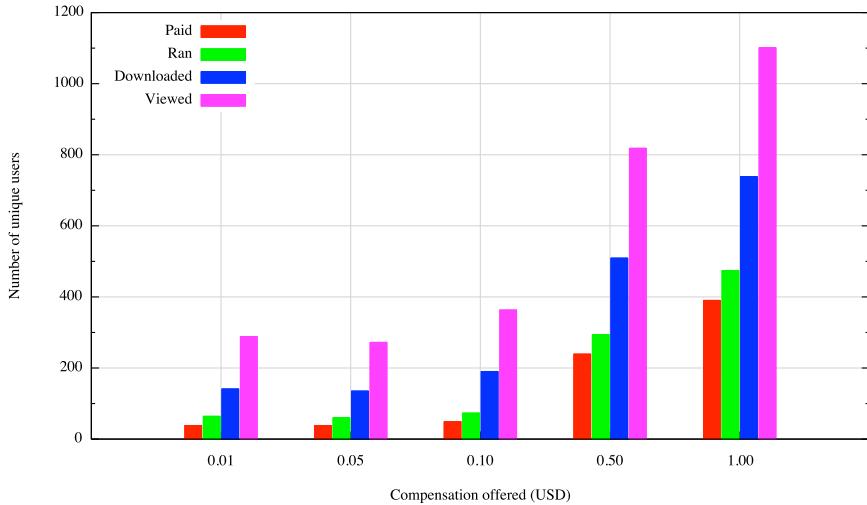
## 4 Results

During the course of our five-week study, our task was viewed 2,854 times. This corresponds to 1,714 downloads, and 965 confirmed executions. We found that the proportion of participants who executed the program significantly increased with price, though even for a payment of \$0.01, 22% of the people who viewed the task downloaded and executed the program (Table 1). This raises questions about the effectiveness of well-known security advice when competing against the smallest of incentives. In this section we analyze our results with regard to differing behaviors among the pricing conditions, data collected from participants' systems, and the exit survey data.

### 4.1 Price Points

Overall, we observed statistically significant differences between the payment amounts with regard to the fraction of participants who downloaded the program after viewing the task description ( $\chi^2_4 = 59.781, p < 0.0005$ ). Upon performing post-hoc analysis using Fisher's exact test, we observed that this was due to differences between the \$0.50 and \$1.00 price points and each of the three lowest price points ( $p < 0.002$  and  $p < 0.0005$ , respectively). We also observed that significantly more people downloaded the program when the price was raised from \$0.50 to \$1.00 ( $p < 0.030$ ).

Once participants downloaded the program, our only indication that they had executed it was when our server received data from the program. If a firewall prevented users from reaching port 123 on our server, we assumed that they had not run the program. Thus, our reported proportion of executes-to-downloads represents a lower bound. Nevertheless, we observed statistically significant differences between the conditions ( $\chi^2_4 = 58.448, p < 0.0005$ ). As was the case with the proportion of downloads, post-hoc analysis showed that this was due to the increase in execution at the \$0.50 and \$1.00 price points ( $p < 0.01$  and  $p < 0.0001$ , respectively). Likewise, significantly more people ran the program when the price was raised from \$0.50 to \$1.00 ( $p < 0.021$ ).



**Fig. 3. Task enrollment and completion rates.** “Paid” corresponds to users who ran the program for a full hour and correctly submitted a payment code; “Ran” corresponds to users who ran the program, no matter how briefly; “Downloaded” and “Viewed” correspond to users who downloaded our program and viewed the task, respectively. We noticed statistically significant differences in participants’ behaviors at the \$0.50 and \$1.00 price points.

Thus, participants’ behaviors did not change relative to the price we were paying them until it was increased to \$0.50, at which point not only did we see a dramatic increase in the total number of people viewing the task, but also a significant increase in the proportion who opted to perform the task. The increase in the former is purely an artifact of our experimental platform, Mechanical Turk, indicating that many users are unwilling to view tasks that pay less than \$0.50. The latter increase is more important since we observed differences in participants’ behaviors as a function of price. These changes in participants’ behaviors continued as the price was further increased to \$1.00.

## 4.2 Participant Behavior

When we compared the participants who were exposed to the UAC dialog (Figure 2) with the participants in the control condition, we found no significant differences.<sup>4</sup> Thus, regardless of price, the warning had no observable effect on participants’ willingness to download and execute an unknown program—even one that required them to explicitly grant it administrator-level permissions. Because of the lack of effect, we disregarded this variable in our analysis.

Because the Red Pill VM-detection routine [28] only works reliably on single-CPU computers, we also collected information on the number of CPUs. Using Red Pill, we

<sup>4</sup> It is possible that some participants in the UAC condition had disabled this feature, which we did not attempt to detect. However, we do not believe these participants confound our findings since they provide a realistic snapshot of UAC usage.

detected the presence of a VM on a single participant’s machine. Examining each participants’ process lists, we were able to confirm that this participant was running VMware. Additionally, we detected VMware Tools running on an additional fifteen machines (sixteen in total), and Parallels Tools running on a single machine. Thus, we can confirm that at least seventeen participants (1.8% of 965) took the precaution of using a VM to execute our code.<sup>5</sup> Eleven of these participants were in the \$1.00 condition, five were in the \$0.50 condition, and one was in the \$0.01 condition. The information we collected on participants’ motherboards was not useful in determining VM usage.

When the program was executed, we received a list of the other processes that each participant was running. In this manner we amassed a list of 3,110 unique process names, which we categorized as either corresponding to malware or security software using several online databases. Because we only recorded the image names and not the sizes, file locations, or checksums, these classifications are very rough estimates. We took great care to only classify something as malware after being completely certain; this created a lower bound on the number of infected systems. Conversely, we took a liberal approach to categorizing something as security software in order to create an upper bound; it is possible—indeed likely—that some of the image names masquerading as security software were actually malware. All in all, we found no significant differences between the pricing conditions with regard to malware infections or the use of security software; at least 16.4% (158 of the 965 who ran the program) had a malware infection, whereas as many as 79.4% had security software running (766 of 965). Surprisingly, we noticed a significant positive trend between malware infections and security software usage ( $\phi = 0.066, p < 0.039$ ). That is, participants with security software were *more* likely to also have malware infections (17.6% of 766), whereas those without security software were *less* likely to have malware infections (11.6% of 199). While counter-intuitive, this may indicate that users tend to exhibit risky behavior when they have security software installed, because they blindly trust the software to fully protect them.

Once we increased the payment to \$0.50, participants made different decisions with regard to their system security. We collected data on their Windows versions in order to examine if they had applied the most recent service packs. We considered this a proxy for risk aversion—similar to the use of security software; in the computer security context, higher risk aversion should correspond to an increase in up-to-date systems. Indeed, we observed this effect: 72.1% and 67.5% of participants who were paid \$0.50 and \$1.00, respectively, had a current version of Windows, compared to an average of 54.3% of participants who were paid less ( $p < 0.0005$  for Fisher’s exact test).

Finally, we found that the better-compensated participants also performed the task more diligently. We examined the number of participants in each condition who submitted an invalid payment code (Table 2) and found that “cheating” significantly decreased as the payment increased ( $\chi^2_4 = 52.594, p < 0.0005$ ). Likewise, we found a statistically significant correlation between time spent running the executable and payment amount ( $r = 0.210, p < 0.0005$ ).

---

<sup>5</sup> We do not know if this was a precaution or not. It is equally likely that participants were simply using VMs because they were Mac or Linux users who wanted to complete a Windows task.

**Table 2.** For each payment amount, each row depicts: the average number of minutes participants spent running the program (out of 60), the percentage of participants who ran the program for the full 60 minutes, and the percentage of participants who attempted to cheat by submitting an invalid payment code (out of the total number of submitted payment codes)

	\$0.01	\$0.05	\$0.10	\$0.50	\$1.00
<i>average runtime in minutes</i>	39.30	43.67	43.63	52.02	52.65
<i>percentage running the program for 60 minutes</i>	59.4%	63.3%	67.1%	81.3%	82.3%
<i>percentage of invalid codes submitted</i>	46.5%	28.3%	24.6%	12.8%	14.8%

Thus, we observed that not only did the \$0.50 and \$1.00 price points allow us to recruit more participants, but the overall quality—reliability and modernity—of the machines recruited and the diligence of their operators considerably increased as well.

The measurements we obtained through instrumentation of our web site and of our software give us a pretty detailed picture of user behavior when facing decisions about whether to run untrusted code or not. We were curious to further examine the “at-risk” population of users who elected to run untrusted code, and thus completed our experiment, in terms of demographic and behavioral characteristics. To that effect, after completing the experimental portion of this study, we invited participants who had submitted a valid payment code to complete an exit survey for a bonus payment of \$0.50.

### 4.3 Self-reported Data

In the exit survey we collected information about participants’ demographics, their risk perceptions, and their stated security practices.

*Demographics.* A total of 513 participants completed the survey. Forty percent of respondents came from India, 30% from the US or Canada, and the majority of the rest were from Europe.<sup>6</sup> Using the United Nations’ list of developed countries [33], we partitioned participants based on their nationalities. We observed that the proportion of participants from the developed world increased with the payment amount: from 9.4% to 23.4%. We found that this increase was statistically significant when comparing the \$0.01 condition to the \$0.50 and \$1.00 conditions ( $p < 0.02$  and  $p < 0.01$ , respectively), but was not significant when comparing the other conditions. We did not observe significant differences with regard to age (mean  $\mu = 28.1$  years old, with standard deviation  $\sigma = 8.95$  years) or gender (349 males and 164 females).

We gauged participants’ security expertise with four questions:

- Do you know any programming languages?
- Is computer security one of your main job duties?
- Have you attended a computer security conference or class in the past year?
- Do you have a degree in computer science or another technical field (e.g. electrical engineering, computer engineering, etc.)?

<sup>6</sup> Thirty-three participants listed their nationalities as “Caucasian.” After unsuccessfully trying to locate Caucasia on a map, we omitted their responses to this question.

We observed no differences between the conditions with regard to the individual conditions nor when we created a “security expert score” based on the combination of these factors. That is, the proportion of people who answered affirmatively to each of the questions remained constant across the price intervals. So why did participants execute the program?

*Security concerns.* Windows’ automatic update functionality is disabled when it determines that the software may have been pirated. We observed a significant correlation between participants from the developing world and unpatched versions of Windows ( $\phi = 0.241, p < 0.0005$ ), which hints at a correlation between software piracy and the developing world. However, we do not believe that this effect is solely responsible for the changes in behavior as a function of price that we observed during our study.

We asked participants to rate the danger of running programs downloaded from Mechanical Turk using a 5-point Likert scale. What we found was that as the price went up, participants’ perceptions of the danger also increased ( $F_{508,4} = 3.165, p < 0.014$ ). Upon performing post-hoc analysis, we observed that risk perceptions were similar between the \$0.01, \$0.05, and \$0.10 price points ( $\mu = 2.43, \sigma = 1.226$ ), as well as between the \$0.50 and \$1.00 price points ( $\mu = 2.92, \sigma = 1.293$ ); once the price hit \$0.50 and above, participants had significantly higher risk perceptions ( $t_{511} = 3.378, p < 0.001$ ). This indicates that as the payment was increased, people who “should have known better” were enticed by the payment; individuals with higher risk perceptions and awareness of good security practices did not take the risk when the payment was set at the lower points, but ignored their concerns and ran the program once the payment was increased to \$0.50. These perceptions were not linked to demographic data.

These results suggest that participants may have felt an increased sense of security based on the context in which our program was presented. As mentioned earlier, though, there was no actual proof this was a legitimate university study, as we purposefully stored the executable on a third-party server and did not provide any contact information.

We periodically browsed Mechanical Turk user forums such as Turker Nation [2] and Turkopticon [3], and noticed with interest that our task was ranked as legitimate because we were paying users on time and anti-virus software was not complaining about our program. Clearly, neither of these statements is correlated in any way to potential security breaches caused by our software. Anti-virus software generally relies on malware being added to blacklists and may not complain about applications that were given full administrative access by a consenting user. Perhaps a warning to the effect that the application is sending network traffic would pop up, but most users would dismiss it as it is consistent with the purpose of a “Distributed Computing Client.” As such, it appears that people are reinforcing their false sense of security with justifications that are not relevant to the problem at hand.

Last, one Turkopticon user also had an interesting comment. He incorrectly believed we were “running a port-scanner,” and was happy with providing us with potentially private information in exchange for the payment he received. This echoes Grossklags and Acquisti’s finding [11] that people do not value privacy significantly.

## 5 Discussion and Conclusions

In this experiment, we observed what economists have known for centuries: it really is all about the Benjamins.<sup>7</sup> We show in the security context how users can be motivated by direct incentive payments to ignore commonly known security advice. Even though around 70% of all our survey participants understood that it was dangerous to run unknown programs downloaded from the Internet, all of them chose to do so once we paid them. We also demonstrate that the strength of user participation is consistent with basic economic principles, so that more money equals more interest in the task, downloads and executions of potentially harmful code, and commitment to the instructions given [9, 15]. This study serves as a clarion call to security researchers and practitioners: security guidance should not only create incentives for compliance with advice and policies, but also explicitly include mechanisms to support users who are prone to trade away security cheaply.

When investigating the correspondence between security concerns and security behaviors more closely, we found that subjects in the high payment condition were more concerned about the risks of downloading programs while doing online work. Similarly, these participants were on average better suited to withstand security threats (i.e., they owned more recently patched machines). In other words, more security-conscious and security-aware subjects were lured into our project with the higher available remuneration. On the one hand, these subjects act consistently in applying security patches to react to their heightened concerns. On the other hand, they waste all their hard security efforts by consensually inviting undocumented code on their machines and bypassing Windows security stopping blocks (i.e., UAC).

There are a number of interpretations for our observations that fit the model of the *bounded rational* computer user. In particular, participation may be due to a strong pull of “immediate gratification” and revealing of time-inconsistent preferences [5]. Further, the task triggers “relative thinking.” That is, we have to consider the promised ease of our task in conjunction with a wage that is appealing relative to many more cumbersome Mechanical Turk offerings, e.g., annotating pictures or transcribing recordings [17]. As a result, individuals further neglect to consider the harm that they may inflict on themselves. Relative thinking also helps to account for the surprising diligence of our subjects in the \$0.50 and \$1.00 conditions. At the same time, even in the \$0.01 condition, 22% of the users who viewed the task downloaded and ran the program. This indicates that even for a negligible remuneration, Internet users are willing to bear significant risk; indeed a better title for this paper may be “It’s All About The Abrahams.”<sup>8</sup>

We further believe that a heightened sense of “false security” explains the involvement of many subjects. First, many of the more concerned individuals owned better patched machines and therefore may have erroneously believed that seemingly legitimate tasks were unlikely to cause harm. Second, this observation was mirrored by the entire user population. We found that the presence of security software was correlated with the number of infections with malicious code on user machines. Third, users’

<sup>7</sup> Benjamin Franklin is pictured on US \$100 bills, hence the slang “Benjamin” for banknotes.

<sup>8</sup> Abraham Lincoln is pictured on US \$0.01 coins.

discussions of our task on message boards may have enticed others to participate despite the lack of any security assurances and tests. Moreover, such messages could have easily been posted by co-conspirators.

To explain why *rational* individuals who are concerned about security and who undertake precautions (i.e., install patches and anti-virus software) nevertheless fail to act securely, we can draw from the safety regulation literature. Most prominently, Peltzman outlined in his theory of risk compensation that individuals evaluate the combined demand for safety and usage intensity (e.g., faster driving on a highway) [25]. In the online security context this means that users with more security-ready systems—irrespective of whether they safeguard themselves or administrators act for them—are comparatively more likely to engage in risky behaviors. For example, they may be more likely to seek free adult content or download undocumented code/media that increases their perceived utility from experience. While this process may be driven by selfish and rational considerations for at least some users, it always increases the risk of collateral damage due to negative externalities.

In the online security context, this behavior is particularly significant because malware developers not only carefully conceal their code from users, but also limit the adverse impact on hosts' systems. Distributed attacks may be implicitly tolerated by users as long as they are not directly affected, and lead to what Bill Cheswick likens to a “toxic waste dump” billowing blue smoke across the Internet [26]. Miscreants can benefit from these negative externalities by harvesting resources at a price far below the social cost they are inflicting on the user population.

In conclusion, providing direct incentives for the installation of undocumented code constitutes another malware distribution channel. In this manner, the notion of a “Fair Trade” botnet is not out of the realm of possibility.<sup>9</sup> The occurrence of such an approach may be relatively short-lived such as in previous examples with some free centralized systems (e.g., Download.com [20]), if distributed work platforms exercise considerably more control over the advertised tasks. However, a payment scheme could also give rise to unforeseen semi-legitimate business models that cannot be easily outlawed.

We argue that once payment is high enough and inconvenience is low enough, users have little incentive to comply with security requests [13]. Worse, behavioral biases further magnify the impact of negative externalities in network security [12]. By allowing attackers to use their machines for further attacks, users themselves become a threat to the network. We hope this research will stimulate further discussions on how to realign user incentives, that have seemingly drifted very far away from what would be needed to produce a desirable outcome.

**Acknowledgments.** This paper greatly benefited from discussions with Stuart Schechter, David Molnar, and Alessandro Acquisti. Lorrie Cranor assisted with IRB documentation. This work is supported in part by CyLab at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office, and by the National Science Foundation under ITR award CCF-0424422 (Team for Research in Ubiquitous Secure Technology) and IGERT award DGE-0903659 (Usable Privacy and Security).

---

<sup>9</sup> “Fair Trade” is a certification for foods coming from producers who receive a living wage [1].

## References

1. Fair Trade USA, <http://www.transfairusa.org/>
2. Turker nation, <http://www.turkernation.com>
3. Turkopticon, <http://turkopticon.differenceengines.com>
4. Acohido, B.: Are there 6.8 million – or 24 million – botted PCs on the Internet? <http://lastwatchdog.com/6-8-million-24-million-botted-pcs-internet/> (Last accessed September 16, 2010)
5. Acquisti, A., Grossklags, J.: Privacy and rationality in individual decision making. *IEEE Security & Privacy* 3(1), 26–33 (2005)
6. Baker, W., Hutton, A., Hylander, C., Novak, C., Porter, C., Sartin, B., Tippett, P., Valentine, J.: Data breach investigations report. In: *Verizon Business Security Solutions* (April 2009)
7. Christin, N., Yanagihara, S., Kamataki, K.: Dissecting one click frauds. In: *Proceedings of the Conference on Computer and Communications Security (CCS)*, Chicago, IL, pp. 15–26 (October 2010)
8. Franklin, J., Paxson, V., Perrig, A., Savage, S.: An inquiry into the nature and causes of the wealth of internet miscreants. In: *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VA, pp. 375–388 (October 2007)
9. Gaechter, S., Fehr, E.: Fairness in the labour market - A survey of experimental results. In: Bolle, F., Lehmann-Waffenschmidt, M. (eds.) *Surveys in Experimental Economics, Bargaining, Cooperation and Election Stock Markets*. Physica Verlag (2001)
10. Good, N., Dhamija, R., Grossklags, J., Aronovitz, S., Thaw, D., Mulligan, D., Konstan, J.: Stopping spyware at the gate: A user study of privacy, notice and spyware. In: *Proceedings of the Symposium on Usable Privacy and Security (SOUPS 2005)*, Pittsburgh, PA, pp. 43–52 (July 2005)
11. Grossklags, J., Acquisti, A.: When 25 cents is too much: An experiment on willingness-to-sell and willingness-to-protect personal information. In: *Proceedings (online) of the Sixth Workshop on Economics of Information Security (WEIS)*, Pittsburgh, PA (2007)
12. Grossklags, J., Christin, N., Chuang, J.: Secure or insure? A game-theoretic analysis of information security games. In: *Proceedings of the 2008 World Wide Web Conference (WWW 2008)*, Beijing, China, pp. 209–218 (April 2008)
13. Herley, C.: So long, and no thanks for the externalities: The rational rejection of security advice by users. In: *Proceedings of the New Security Paradigms Workshop (NSPW)*, Oxford, UK, pp. 133–144 (September 2009)
14. Herley, C., Florêncio, D.: Nobody sells gold for the price of silver: Dishonesty, uncertainty and the underground economy. In: *Proceedings (online) of the Eighth Workshop on Economics of Information Security (WEIS)* (June 2009)
15. Horton, J., Rand, D., Zeckhauser, R.: The online laboratory: Conducting experiments in a real labor market. Harvard Kennedy School and NBER working paper (May 2010)
16. Jakobsson, M.: Experimenting on Mechanical Turk: 5 How Tos (July 2009), <http://blogs.parc.com/blog/2009/07/experimenting-on-mechanical-turk-5-how-tos/>
17. Kahneman, D., Tversky, A.: Choices, values and frames. Cambridge University Press, Cambridge (2000)
18. Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G., Paxson, V., Savage, S.: Spamalytics: An empirical analysis of spam marketing conversion. In: *Proceedings of the Conference on Computer and Communications Security (CCS)*, Alexandria, VA, pp. 3–14 (October 2008)
19. Kittur, A., Chi, E., Suh, B.: Crowdsourcing User Studies with Mechanical Turk. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2008)*, Florence, Italy, pp. 453–456 (2008)

20. Kucera, K., Plaisent, M., Bernard, P., Maguiraga, L.: An empirical investigation of the prevalence of spyware in internet shareware and freeware distributions. *Journal of Enterprise Information Management* 18(6), 697–708 (2005)
21. Matwyshyn, A.: Penetrating the zombie collective: Spam as an international security issue. *SCRIPT-ed* 4 (2006)
22. Moore, T., Clayton, R., Anderson, R.: The economics of online crime. *Journal of Economic Perspectives* 23(3), 3–20 (2009)
23. Moore, T., Edelman, B.: Measuring the Perpetrators and Funders of Typosquatting. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 175–191. Springer, Heidelberg (2010)
24. Namestnikov, Y.: The economics of botnets. In: Analysis on Viruslist. com, Kapersky Lab (2009)
25. Peltzman, S.: The effects of automobile safety regulation. *Journal of Political Economy* 83(4), 677–726 (1975)
26. Reeder, R., Arshad, F.: SOUPS 2005. *IEEE Security & Privacy* 3(5), 47–50 (2005)
27. Ross, J., Zaldivar, A., Irani, L., Tomlinson, B.: Who are the Turkers? Worker Demographics in Amazon Mechanical Turk. Technical Report SocialCode-2009-01, University of California, Irvine (2009)
28. Rutkowska, J.: Red pill.. or how to detect VMM using (almost) one CPU instruction (November 2004), <http://invisiblethings.org/papers/redpill.html>
29. Saroiu, S., Gribble, S., Levy, H.: Measurement and analysis of spyware in a university environment. In: Proceedings of the 1st USENIX Symposium on Networked Systems Design & Implementation (NSDI 2004), San Francisco, CA, pp. 141–153 (2004)
30. Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydłowski, M., Kemmerer, R., Kruegel, C., Vigna, G.: Your botnet is my botnet: Analysis of a botnet takeover. In: Proceedings of the Conference on Computer and Communications Security (CCS), Chicago, IL, pp. 635–647 (October 2009)
31. Symantec Corp. Symantec global internet security threat report trends for 2009 (April 2010)
32. Thomas, R., Martin, J.: The underground economy: Priceless. *Login* 31(6), 7–16 (2006)
33. United Nations Statistics Division. Composition of macro geographical (continental) regions, geographical sub-regions, and selected economic and other groupings (April 2010), <http://unstats.un.org/unsd/methods/m49/m49regin.htm>
34. Wilson, C.: Botnets, cybercrime, and cyberterrorism: Vulnerabilities and policy issues for congress. In: Library of Congress Washington DC Congressional Research Service (January 2008)
35. Zeltser, L.: So long script kiddies. *Information Security Magazine* (2007)

# Evaluating the Privacy Risk of Location-Based Services

Julien Freudiger, Reza Shokri, and Jean-Pierre Hubaux

LCA1, EPFL, Switzerland  
firstname.lastname@epfl.ch

**Abstract.** In modern mobile networks, users increasingly share their location with third-parties in return for location-based services. Previous works show that operators of location-based services may identify users based on the shared location information even if users make use of pseudonyms. In this paper, we push the understanding of the privacy risk further. We evaluate the ability of location-based services to identify users and their points of interests based on different sets of location information. We consider real life scenarios of users sharing location information with location-based services and quantify the privacy risk by experimenting with real-world mobility traces.

## 1 Introduction

In traditional cellular networks, users share their location with their network operator in order to obtain voice and data services pervasively. With the emergence of data services, users increasingly share their location with other parties such as location-based services (LBSs). Specifically, users first obtain their location by relying on the localization capability of their mobile device (e.g., GPS or wireless triangulation), share it with LBSs and then obtain customized services based on their location. Yet, unlike cellular operators, LBSs are mainly provided for free and generate revenue with location-based advertisement. Hence, there is a key difference between the business models of LBS providers and cellular operators: LBS providers aim at profiling their users in order to serve tailored advertisement.

Subscribers of cellular networks know that their personal data is contractually protected. On the contrary, users of LBSs often lack an understanding of the privacy implications caused by the use of LBSs [17]. Some users protect their privacy by hiding behind pseudonyms that are mostly invariant over time (e.g., usernames). Previous works identified privacy threats induced by the use of LBSs and proposed mechanisms to protect user privacy. Essentially, these mechanisms rely on trusted third-party servers that anonymize requests to LBSs. However, such privacy-preserving mechanisms are not widely available and users continue sharing their location information unprotected with third-parties. Similarly, previous work usually considers the worst-case scenario in which users continuously upload their location to third-parties (e.g., traffic monitoring systems). Yet, with most LBSs, users do not share their location continuously but instead, connect *episodically* to LBSs depending on their needs and thus reveal a few location samples of their entire trajectory. For example, a localized search on Google Maps [19] only reveals a location sample upon *manually* connecting to the service.

In this work, we consider a model that matches the common use of LBSs: we do not assume the presence of privacy-preserving mechanisms and consider that users access LBSs on a regular basis (but not continuously). In this setting, we aim at understanding the privacy risk caused by LBSs. To do so, we experiment with real mobility traces and investigate the *dynamics* of user privacy in such systems by measuring the *erosion* of user privacy. In particular, we evaluate the success of LBSs in predicting the true identity of pseudonymous users and their points of interest based on different samples of mobility traces. Our results explore the relation between the *type* and *quantity* of data collected by LBSs and their ability to de-anonymize and profile users. We quantify the potential of these threats by carrying out an experimentation based on real mobility traces from two cities, one in Sweden and one in Switzerland. We show that LBS providers are able to uniquely identify users and accurately profile them based on a small number of location samples observed from the users. Users with a strong routine face higher privacy risk, especially if their routine does not coincide with that of others. To the best of our knowledge, this work is among the first to investigate the erosion of privacy caused by the sharing of location samples with LBSs using real mobility traces and to quantify the privacy risk.

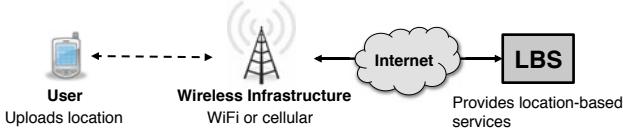
## 2 State of the Art

Location-based services [1, 19, 12, 30] offer users to discover their environment. While some services such as road-traffic monitoring systems require users to continuously share their location, in most services, users share their location *episodically*. We consider users that manually reveal few samples of their mobility.

The IETF Geopriv working group [3] aims at delivering specifications to implement location-aware protocols in a privacy-conscious fashion. It suggests that independent location servers deliver data to LBSs according to user-controlled privacy policies. We complement the IETF proposal by quantifying the privacy threat induced by sharing specific locations with LBSs.

Privacy-preserving mechanisms (PPMs) impede LBSs from tracking and identifying their users [5, 20, 21, 22, 24, 32, 43]. Most mechanisms alter users' identifiers or location samples, and either run on third-party anonymizing servers or directly on mobile devices. PPMs' effectiveness is usually evaluated by measuring the level of privacy [10, 38, 41]. PPMs are rarely used in practice. One reason may be that users do not perceive the privacy threat because they are not intensively sharing their location. We aim at clarifying the privacy threat when users disregard PPMs.

Without PPMs, pseudonymous location information identifies mobile users [4, 6, 26, 33]. Beresford and Stajano [4] identified all users in continuous location traces. Using GPS traces from vehicles, two studies by Hoh *et al.* [23] and Krumm [26] found home addresses of most drivers. De Mulder *et al.* [33] could identify mobile users in a GSM cellular network from pre-existing location profiles. These works rely on continuous location traces precisely capturing users' whereabouts. Yet, in most scenarios, users share location samples episodically. Partridge and Golle [18] could identify most US working population using *two* location samples, i.e., approximate home and work locations. It remains unclear however whether users of LBSs face this threat. We consider real-life



**Fig. 1.** System model. Users episodically upload their location wirelessly to LBSs.

scenarios and investigate associated privacy risks. Ma *et al.* [31] study the erosion of privacy caused by published anonymous mobility traces and show that an adversary can relate location samples to published anonymous traces. We push further the analysis by considering an adversary without access to anonymized traces.

### 3 System Model

We present the assumptions regarding LBSs and the associated privacy threats.

#### 3.1 Network Model

We study a network (Fig. 1) of mobile users equipped with wireless devices communicating with LBSs via a wireless infrastructure. Wireless devices feature localization technology such as GPS or wireless triangulation for users' localization. The geographic *location* of a user is denoted by  $l$ . The wireless infrastructure relies on technology such as WiFi, GSM or 3G. LBSs are operated by independent third-parties that provide services based on users' location.

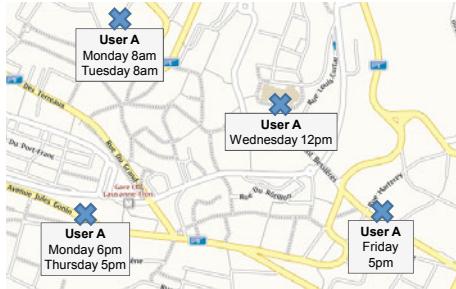
Users send their location together with a service request to LBSs. For each request, users may identify themselves to the LBS using proper credentials. We assume that users are identified with *pseudonyms* (i.e., fictitious identifiers), such as their username, HTTP cookie or IP address. Some services may require users registration with username and password, whereas others may use HTTP cookies to recognize users.

LBSs provide users with services using the location information from requests. LBSs store collected users' locations in a database. Each location sample is called an *event* [39] denoted by  $< i, t, l >$ , where  $i$  is a user's pseudonym,  $t$  is the time of occurrence, and  $l$  is a user's location. A collection of user events forms a *mobility trace*.

#### 3.2 Threat Model

LBS operators passively collect information about the locations of pseudonymous users over time. For example, an LBS can observe location samples of user A over the course of weeks (Fig. 2). A priori, the LBS also knows the map over which users move and has access to geographic information systems that provide details such as points of interest in maps [8]. In addition, the LBS may use the increasing amount of location information available from other sources such as public transportation systems [37].

We consider that the LBS aims at obtaining the true identity of its users and their points of interest. To do so, the LBS studies the collected information. Even if communications are pseudonymous, the spatio-temporal correlation of location traces may



**Fig. 2.** Threat model. The LBS learns multiple locations of user A over several days. It can then infer the activities of user A and possibly obtain his true identity.

serve as a *quasi-identifier* [5, 6, 9, 18]. This is a significant threat to user privacy as such information can help the LBS to profile users. In this work, we investigate the ability of LBSs to identify users and their points of interest based on the collected information.

## 4 Privacy Erosion

In this section, we describe the process of *privacy erosion* caused by the use of LBSs. To do so, we first discuss how users tend to share their location with LBSs. Then, we explain how LBS operators can obtain the true identity of their users and their points of interest from the collected information.

### 4.1 Collection of Traces by LBSs

Depending on the provided service, LBSs collect *location samples* about their users. For example, services (such as Foursquare, Google Maps, Gowalla, or Twitter) offer users to connect with friends, to discover their environment or to optimize their mobility. Depending on their needs, users access such LBSs at different times and from different locations, thus revealing multiple location samples. In general, users of these services do not continuously share their locations. Instead, users have to manually decide to share their location with their mobile devices. We classify popular LBSs into three broad categories and describe the shared information in each case.

**Localized Search.** Many LBSs enable users to *search* for services around a specific location (e.g., localized Google Search [19]). Localized searches offer mobile subscribers spontaneous access to nearby services. A user location acts as a spatial query to the LBS. For example, users can obtain the location of nearby businesses, products or other local information depending on the type of information provided by the LBS.

Such information helps users navigate unfamiliar regions and discover unknown places. Users episodically connect to LBSs, revealing samples of their mobility. LBSs obtain statistical information about visited locations and learn popular locations and user habits. Yet, LBSs do not learn the actual activity of mobile users (e.g., the name of the restaurant) as they do not know users' decision about the provided information.

**Street Directions.** Another popular use of LBSs consists in finding a route between two locations. Typically, a user shares its location with an LBS and requests the shortest route to another location (e.g., Google Maps).

Users of such services usually reveal their current location and a potential destination. Hence, users may leak their home/work locations to LBSs in addition to samples of their mobility. This enables LBSs to obtain statistical information about the preferred origins and destinations of mobile users.

**Location Check-ins.** A novel type of location-based service offers users to check-in to specific *places* in return for information related to the visited location [12]. For example, it can be used to check into shops, restaurants or museums. It allows users to meet other users that share similar interests and to discover new aspects of their city through recommendations [30].

With such services, users not only precisely reveal their location (GPS coordinates), but also their intention. Indeed, the LBS can learn the current activity of its users. Users can check-in to public places, but also private homes.

In summary, depending on the provided service, users share different *samples* of their mobility. In order to take this into account, in the following, we consider that LBSs obtain various *type* of location samples out of users' whereabouts.

## 4.2 Attacks by LBSs

The spatial and temporal information contained in mobility traces may serve as location-based quasi-identifiers [6,9]: an LBS may obtain the true identity and points of interests of its pseudonymous users from the collected mobility traces.

**Location-Based Quasi-Identifiers.** Quasi-identifiers were introduced by Delenius [9] in the context of databases. They characterize a set of attributes that in combination can be linked to identify to whom the database refers (see [34] for more). In [6], Bettini *et al.* extend the concept to mobile networking. They consider the possibility to identify users based on spatio-temporal data and propose the concept of location-based quasi-identifiers. They describe how a sequence of spatio-temporal constraints may specify a mobility pattern that serve as a unique identifier. For example, as already mentioned, Golle and Partridge [18] identify location-based quasi-identifiers by showing that home and work locations uniquely identify most of the US population. Hence, if an LBS learns users' home and work locations, it can obtain their identity with high probability.

In this work, we assume that the LBS succumbs to the temptation of finding the identity of its users. To do so, we consider that the LBS uses the home and work locations of users as location-based quasi-identifiers.

**Inferring Home and Work Locations.** Previous works investigated the problem of characterizing and extracting important places from pseudonymous location data. These works propose various algorithms to infer important locations based on the spatial and temporal evidence of the location data. We group existing work in two categories. In the first category [2, 23, 26], the authors use *clustering* algorithms to infer the *homes* of mobile users. For example in [26], Krumm proposes four different clustering techniques to identify the homes of mobile users in vehicular traces: traces are pseudonymous and

contain time-stamped latitudes and longitudes. Similarly, in [23], Hoh *et al.* propose a  $k$ -mean clustering algorithm to identify the homes of mobile users in anonymous vehicular traces: traces do not have pseudonyms, but contain the speed of vehicles, in addition to their location. In the second category [28, 29], Liao *et al.* propose *machine learning* algorithms to infer the different type of *activities* from mobile users data (e.g., home, work, shops, restaurants). Based on pedestrian GPS traces, the authors are able to identify (among other things) the *home and work locations* of mobile users.

We rely on previous work to derive an algorithm that exclusively infers users' *home* and *work* locations based on spatial and temporal constraints of location traces. The algorithm has two steps: first, it spatially clusters events to identify frequently visited regions; second, it temporally clusters' events to identify home/work locations.

The spatial clustering of the events uses a variant of the  $k$ -means algorithm as defined in [2]: it starts from one random location and a radius. All events within the radius of the location are marked as potential members of the cluster. The mean of these points is computed and is taken as the new centre point. The process is repeated until the mean stops changing. Then, all the points within the radius are placed in the cluster and removed from consideration. The procedure repeats until no events remain. The number of points falling into a cluster corresponds to its weight and is stored along with the cluster location. Clusters with a large weight represent frequently visited locations.

Based on the output of the spatial filtering, the algorithm uses temporal evidence to further refine the possible home/work locations. In practice, users have different temporal patterns depending on their activities (e.g., students). The algorithm considers simple heuristics that apply to the majority of users. For example, most users spend the night at home and commute in the beginning/end of the day. The algorithm considers all events in each cluster and labels them as home or work. Some events may remain unlabeled. The algorithm considers two temporal criteria: First, the algorithm checks the duration of stay at each cluster. To do so, it computes the time difference between the arrival and departure at a certain cluster. A user that stays more than 1 hour in a certain cluster over night is likely to have spent the night at home. Hence, the algorithm labels events occurring at such cluster as home events. Second, the algorithm labels events occurring after 9am and before 5pm as work events. Finally, for each cluster, the algorithm checks the number of events labelled home/work and deduces the most probable home/work locations.

**Inferring User Points of Interest.** Usually, LBSs use the content of queries to infer the points of interest of their users. Yet, LBSs may further profile users by analyzing the location of multiple queries and inferring users' points of interest.

We use the spatial clustering algorithm defined above to obtain the possible points of interest of users that we call *uPOIs*: a uPOI is a location regularly visited by a user. For each identified uPOI, we store the number of visits of the user and derive the probability  $P_v^i$  that a user  $i$  visits a specific uPOI, i.e., the number of visits to a uPOI normalized by the total number of visits to uPOIs.

**Metrics.** The real home/work addresses are unavailable in our data sets. Hence, we apply our algorithm to the original mobility traces and derive a baseline of home/work locations. We then evaluate the probability of success of the LBS by comparing the baseline to the outcome of our algorithm on the *samples* of location data collected by

the LBS. In other words, we compare the home/work location pairs predicted from the sampled traces with the baseline. In practice, it is complicated to obtain the real home and work locations of users (i.e., the ground truth) in mobility traces without threatening their privacy. Because no real ground truth is available, this approach does not guarantee that we have identified the real home/work locations. Yet, it allows us to compare the effectiveness of the attack in various conditions.

The probability  $P_s$  of a successful identification by the LBS is then:

$$P_s = \frac{\text{Number of home/work pairs correctly guessed}}{\text{Total number of home/work pairs}} \quad (1)$$

This probability measures the ability of LBSs to find the home/work locations from sampled traces and thus uniquely identify users. This metric relies on the assumption that home/work location pairs uniquely identify users [18] (in Europe as well): it provides an upper-bound on the identification threat as home/work location pairs may in practice be insufficient to identify users especially in the presence of uncertainty about the home/work locations.

We also evaluate the normalized *anonymity set* of the home and work pairs of mobile users. To do so, we compute the number of home/work locations that are in a certain radius from the home/work location of a certain user  $i$ . For every user  $i$ , we define its home location as  $h_i$  and its work location as  $w_i$ . For each user  $i$ , we have:

$$A_{\text{home}}^i = \frac{1}{|h|} \sum_{j \neq i} 1_{|h_j - h_i| < R_A} \quad (2)$$

$$A_{\text{work}}^i = \frac{1}{|w|} \sum_{j \neq i} 1_{|w_j - w_i| < R_A} \quad (3)$$

where  $R_A$  specifies the radius considered for the anonymity set.

We measure the ability of LBSs to infer uPOIs by considering for each user  $i$ , the number of uPOIs correctly inferred. For every user  $i$ , we have:

$$P_{\text{uPOI}}^i = \frac{\text{Number of uPOIs correctly guessed}}{\text{Number of uPOIs}} \quad (4)$$

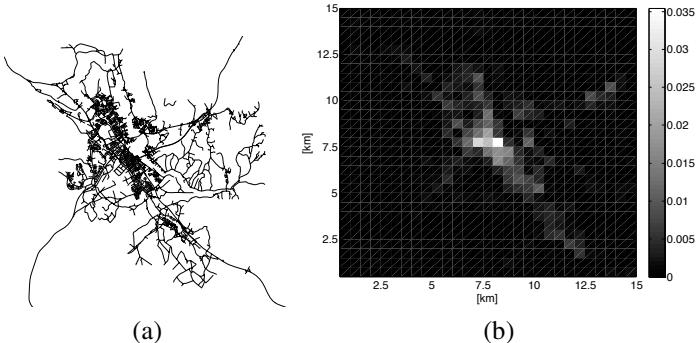
We also use the notion of Kullback-Leibler divergence [27] to measure the ability of the adversary to guess the probability of each user visiting specific uPOIs. For every user  $i$ , we have:

$$D_{KL}(P_v^i || Q_v^i) = \sum_j P_v^i(j) \log \frac{P_v^i(j)}{Q_v^i(j)} \quad (5)$$

where  $P_v^i$  is the actual probability that user  $i$  visits specific uPOIs and  $Q_v^i$  is the probability guessed by the adversary.

## 5 Evaluation

We present our methodology to evaluate the erosion of privacy caused by LBSs.



**Fig. 3.** Borlange data set. (a) Map of Borlange, Sweden. The city has 46000 inhabitants and spreads over  $15 \times 15\text{km}^2$ . (b) Spatial histogram showing the density of users per cell  $c(z)$ .

## 5.1 Setup

We start from data sets of real mobility traces. The data sets contain the location of users at a high granularity. Because users usually reveal only a few location samples to LBS operators, we artificially reduce the information available to the LBSs by selecting a few events from the traces. Then, we consider various de-anonymization attacks on the location traces. In practice, we load mobility traces in Matlab and apply the algorithm described in Section 4.2. We repeat every analysis 100 times and consider the average.

## 5.2 Mobility Traces

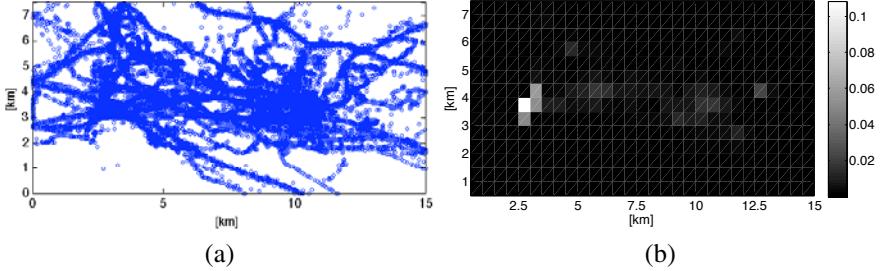
There exist several publicly available data sets of human mobility. For example, there are mobility traces of taxis [36], of student mobility in campus [11], or of sport activities [35]. Yet, most of these data sets have a limited applicability to our problem because the mobility of users is tied to specific scenarios (e.g., taxis, campus).

In this work, we consider two data sets representing normal activities of users in cities. These mobility traces contain several *trips* for each user. A trip defines a trajectory of a user going from one source location to a destination (e.g., a user commuting from home to work). Users move on a map following road constraints.

**Borlange Data Set.** The city of Borlange is a middle-sized ( $15 \times 15\text{km}^2$ ) Swedish city of approximately 46000 inhabitants. Borlange has 3077 road intersections interconnected by 7459 roads (Fig. 3 (a)). The data set was collected over two years (1999-2001) as part of an experiment on traffic congestion that took place there.<sup>1</sup> About 200 private cars (with one driver per car) within a 25 km radius around the city center were equipped with a GPS device. At regular intervals (approximately every 5 seconds), the position, time and speed of each vehicle was recorded and stored. Mostly because of GPS accuracy issues, many observed trips did not match the Borlange map. The data was thus manually verified and corrected using road fitting algorithms for a subset of 24 vehicles resulting

<sup>1</sup> The data set is available at

<http://icapeople.epfl.ch/freudiger/borlange.zip>



**Fig. 4.** Lausanne data set. (a) Map of Lausanne area, Switzerland. The city has 120000 inhabitants and spreads over  $15 \times 7\text{km}^2$ . (b) Spatial histogram showing the density of users per cell  $c(z)$ .

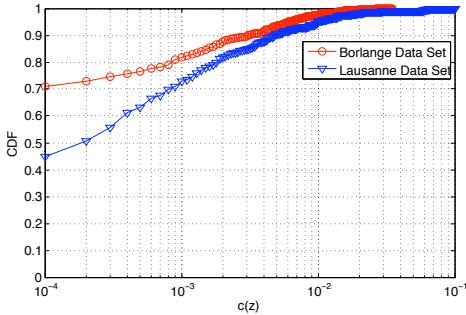
in a total of 420814 “clean” trips (see [13] for more details). This data set was obtained by civil engineers and used to analyze the route choices of mobile users.

**Lausanne Data Set.** The Lausanne area in Switzerland is a region of  $15 \times 7\text{km}^2$  of approximately 120000 inhabitants (Fig. 4 (a)). In September 2009, Nokia [25] began running a data collection campaign in Lausanne area. Around 150 users are equipped with GPS-enabled Nokia phones that record their daily activities and upload them on a central database. Among other things, the phones measure the GPS locations of users at regular intervals (approximately every 10 seconds). In July 2010, we took a snapshot of the database containing traces of 143 users tracked over 12 months.<sup>2</sup> Note that the database contains traces of pedestrians, but also of users in cars, buses and trains. It has thus a larger diversity in terms of mobility patterns than the Borlange data set. We consider location samples in the Lausanne area and 40 users with at least 1000 samples.

In order to evaluate the statistical relevance of the mobility traces, we compute statistics of mobility in the data sets. We divide the whole region of Borlange/Lausanne into square cells of equal size ( $500 \times 500\text{m}$ ). and evaluate the distribution of users’ visits in each cell. We define a variable  $C_z$  that counts the number of events among all users that happen in each cell  $z$ . For each cell, we compute the empirical probability that an event falls into the cell  $z$ ,  $c(z) = \frac{C_z}{\sum_x C_x}$ . In Figure 3 (b), we show the density map (i.e., the set of cells with their corresponding  $c(z)$ ) for the Borlange data set. We observe that the activity of users is concentrated in a few regions. We observe a similar distribution in the Lausanne data set (Fig. 4 (b)). Yet, in the latter, there is a small bias towards one location (the EPFL campus), indicating that many users from the experiment share the same work place. The Lausanne data set reflects scenarios in which many users share the same work place, for example, downtown of a large city.

In Figure 5, we show the empirical Cumulative Distribution Function (CDF) of  $c(z)$  for both data sets in semi-log scale. We observe that the CDF increases linearly, indicating a heavy-tailed distribution of user density. This confirms that some cells have a density much above the average. Our observations about the heavy-tailed distribution match existing results in the literature on mobility traces [7, 35, 42] and confirm the statistical relevance of the data sets. Intuitively, the heavy-tailed distribution indicates that users share few locations.

<sup>2</sup> The data set is not publicly available.



**Fig. 5.** Empirical CDF of  $c(z)$  in semi-log scale. We observe a linear behavior for both data sets indicating a heavy-tailed distribution of user density in the network.

### 5.3 Modeling the Collection of Traces by LBSs

As described in Section 4.1, the *type* and *quantity* of location information collected by LBSs depends on the service. We select in various ways a subset of mobility traces containing location samples at high granularity. Each event is a *query* to LBSs.

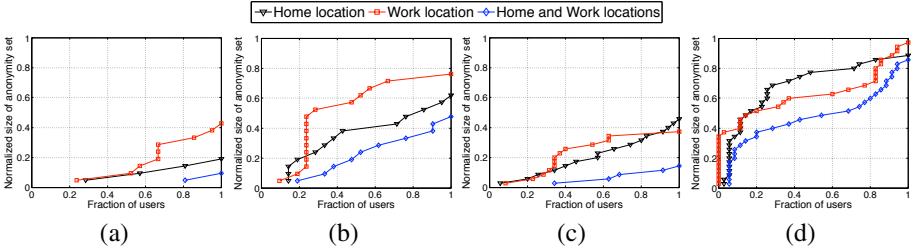
**Uniform Selection (UF).** We select events uniformly at random from the set of all possible events of each user. This models users likely to use an LBS anytime and anywhere.

**Home/Work Selection (HW).** We distinguish between two types of events: *home/work* and *miscellaneous*. Home/work events refer to queries made from home or work, whereas miscellaneous events refer to other visited locations. We select location samples uniformly in each set of *home/work* events with probability  $\rho$  and miscellaneous events with probability  $1 - \rho$ . A large  $\rho$  models users accessing LBSs mostly from home/work (e.g., street directions), whereas a small  $\rho$  models users accessing LBSs mostly on the go (e.g., localized search or location check-ins).

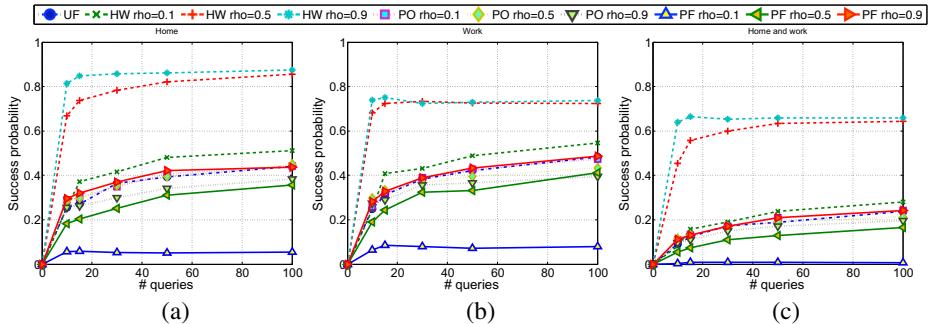
**Points of Interest Selection (PO).** We distinguish between two types of events: *cPOIs* and *miscellaneous*. cPOI events refer to queries made from regions with high density of points of interest (e.g., POIs of the city), whereas miscellaneous events refer to other visited locations. We select location samples uniformly in each set of cPOI events with probability  $\rho$ . A large  $\rho$  models users accessing LBSs mostly from popular locations (i.e., localized search), whereas a small  $\rho$  models users accessing LBSs mostly in unpopular areas such as residential areas.

**Preferred Selection (PF).** We distinguish between two types of events: *preferred* and *miscellaneous*. Preferred events refer to queries made from locations frequently visited by each user (i.e., uPOIs), whereas miscellaneous events refer to other visited locations. We select location samples corresponding to preferred events with probability  $\rho$ . A large  $\rho$  models users accessing LBSs mostly during their routine, whereas a small  $\rho$  models users accessing LBSs mostly in unfamiliar areas.

We tune the selection type using probability  $\rho$ . Note that home/work selection strategy with  $\rho = 0.5$  is different from the uniform selection strategy: with  $\rho = 0.5$  in home/work selection, home/work events and miscellaneous events have the same



**Fig. 6.** Normalized size of anonymity set  $A^i_{home}$ ,  $A^i_{work}$  and  $A^i_{homeWork}$ . Borlange with (a)  $R_A = 1\text{km}$  and (b)  $R_A = 3\text{km}$ . Lausanne with (c)  $R_A = 1\text{km}$  and (d)  $R_A = 3\text{km}$ .



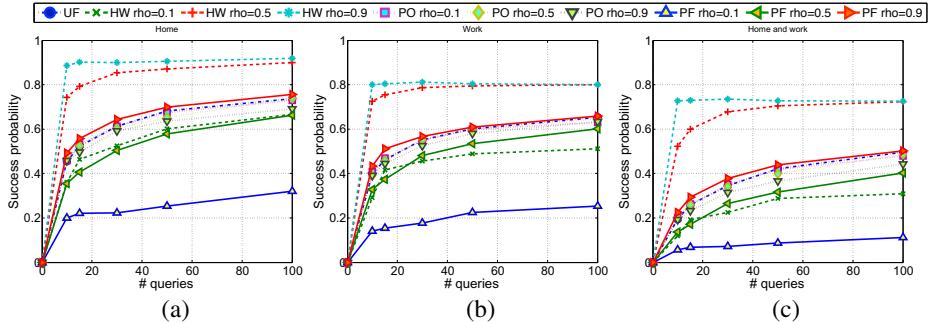
**Fig. 7.** Privacy erosion in Borlange with varying selection probability  $\rho$  and number of queries  $\lambda$ . (a) Home identification. (b) Work identification. (c) Home and work identification ( $P_s$ ).

probability to be chosen, whereas with the uniform selection, all events have the same probability to be chosen. We consider various number of queries  $\lambda$  in order to model the quantity of data collected by LBSs. For example, a number of queries  $\lambda = 60$  means that 60 samples of all location samples of each user are shared with the LBS.

## 5.4 Results

Unless otherwise stated, we consider that users obtain their location with a 10 meters precision (i.e., GPS), that the clustering radius in the spatial clustering algorithm is 100 meters and that the adversary has a tolerable error margin of 50 meters to correctly guess a home/work/uPOI locations.

**Size of Anonymity Set.** The graphs in Fig. 6 detail the size of the anonymity set for home locations, work locations, or both normalized with the number of users in the data set. On the x-axis, the graphs show the fraction of users that has an anonymity set of less than a given normalized size on the y-axis. We consider two radius  $R_A = 1\text{km}$  and  $R_A = 3\text{km}$ . As predicted in [18], the anonymity set size is low especially when a small radius is used and revealing home/work locations is more identifying than revealing one of them. The difference is attenuated for a larger radius. Hence, we obtain results similar to [18] for two European cities. We observe that more users tend to share a common work place than home.



**Fig. 8.** Privacy erosion in Lausanne with varying selection probability  $\rho$  and number of queries  $\lambda$ . (a) Home identification. (b) Work identification. (c) Home and work identification ( $P_s$ ).

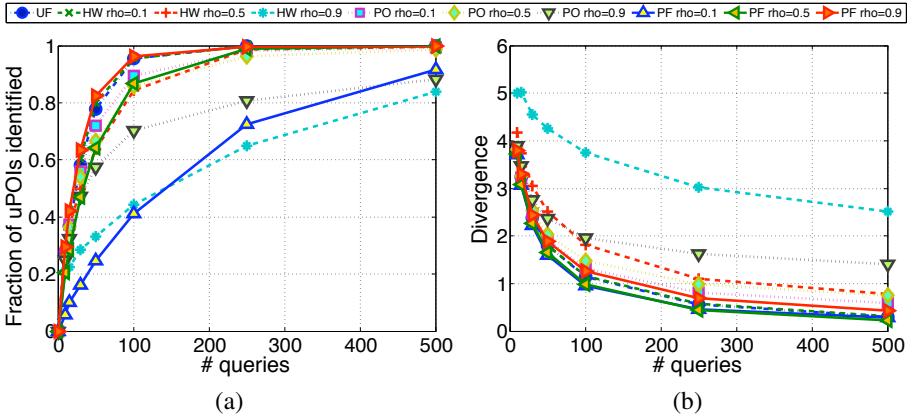
**Privacy Erosion.** We evaluate the privacy erosion of users from the Borlange and Lausanne data sets in multiple scenarios. We measure the probability that an LBS successfully identifies the home location, the work location, or both. In the case of a successful home and work identification, the LBS successfully identifies its users. We consider different data collection scenarios as described earlier (UF, HW, PO and PF) with three selection probabilities:  $\rho = 0.1$ ,  $\rho = 0.5$  and  $\rho = 0.9$ . We also vary  $\lambda$ , the amount of information shared with LBSs.

In Fig. 7 and Fig. 8, we show the erosion of privacy in Borlange and Lausanne for various  $\rho$ ,  $\lambda$  and selection strategies. The success probability  $P_s$  first increases fast and then slowly increases until  $P_s = 1$ . We observe that with HW selection, the probability of identification of a home, work, or home/work pair increases the fastest with respect to the number of sent queries indicating that LBSs uniquely identify users with few locations: in Borlange, if  $\rho = 0.9$ , 20 queries are sufficient to identify 65% of users. We observe that as  $\rho$  increases, so does the identification success. In Lausanne, the identification success is slightly higher but still leads to the same conclusions. We observe that PF selection with  $\rho = 0.1$  makes de-anonymization particularly difficult. In this case, users share their location only in unfamiliar areas and it is thus difficult for LBSs to infer users' identity. For other selection strategies, the identification success saturates around 20 to 40% and increases slowly with the number of queries.

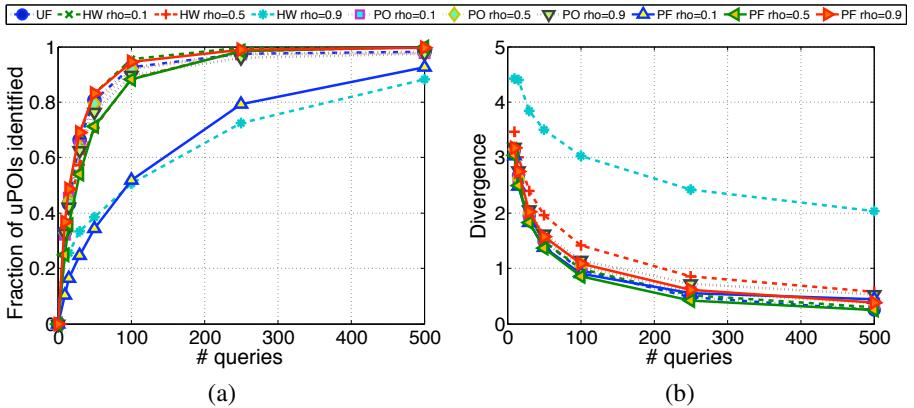
**Inferred User Points of Interest.** Table 1 shows the average fraction of visits to uPOIs. Each uPOI identifies a region of 200 meters radius frequently visited by each user. In both data sets, the distribution is long tail showing that few uPOIs are frequently visited.

**Table 1.** Average probability  $E[P_v^i]$  of visiting a uPOI

Data Set	uPOIs									
	1	2	3	4	5	6	7	8	9	10
Borlange	0.357	0.209	0.112	0.078	0.052	0.031	0.021	0.017	0.015	0.012
Lausanne	0.401	0.14	0.092	0.063	0.045	0.035	0.028	0.023	0.019	0.016



**Fig. 9.** Inferring the top ten uPOIs in Borlange data set. (a) Average fraction of uPOIs identified  $E[P_{uPOI}^i]$ . (b) Average divergence  $E[D_{KL}(P_v^i || Q_v^i)]$ .



**Fig. 10.** Inferring the top ten uPOIs in Lausanne data set. (a) Average fraction of uPOI identified  $E[P_{uPOI}^i]$ . (b) Average divergence  $E[D_{KL}(P_v^i || Q_v^i)]$ .

In Figure 10, we show the ability of LBSs to infer the top ten uPOIs of each user: we compute the average fraction of uPOIs identified  $E[P_{uPOI}^i]$  within a 100 meters error margin and evaluate the average divergence  $E[D_{KL}(P_v^i || Q_v^i)]$ . We observe that the adversary can infer a large number of uPOIs with a small number of samples: with 30 samples, it can learn up to 65% of uPOIs in the case of  $\text{PF } \rho = 0.9$ . The best selection strategies are  $\text{PF } \rho = 0.9$ ,  $\text{HW } \rho = 0.1$  and  $\text{UF}$ . Intuitively, revealing preferred visited locations reveals clusters, similarly, uniform across visited locations will have high probability to sample from frequently visited location. On the contrary, with  $\text{PO } \rho = 0.9$ ,  $\text{HW } \rho = 0.9$  or  $\text{PF } \rho = 0.1$ , the attack works less efficiently. Hence, even with a few location samples, the adversary is also able to infer most uPOIs.

In terms of divergence, a divergence of zero indicates a perfect match. We observe that the divergence decreases fast indicating that the adversary obtains a probability distribution similar to the true one and identifies the most probable uPOIs. We observe a similar behavior with the Lausanne data set. Note that the ability to infer uPOIs is at odds with the ability to infer users' identity: with HW  $\rho = 0.9$ , it is harder to identify uPOIs and easier to identify users.

## 6 Conclusion

We have considered the problem of privacy erosion when using location-based services. We identify the quantity and type of location information that statistically helps LBSs find users' real identity and points of interest. In contrast with previous work (mostly showing that de-anonymization based on location information is possible), we push the understanding of the threat further by showing how de-anonymization depends on the collected data. We experiment with two real data sets of mobility traces, model the collection of traces by LBSs and implement various attacks. Our results show that in many scenarios a small amount of information shared with LBSs may identify users. These results stem from the fact that the spatio-temporal correlation of location traces tends to be unique to individuals and persistent. We also show that in some scenarios, users have high privacy without using privacy-preserving mechanisms.

The results of this work can help prevent the false sense of anonymity that users of LBSs might have by increasing the awareness of location privacy threats. In particular, it may encourage users to stop revealing sensitive information to third-parties, such as their home/work locations, and adopt PPMs. In the future, other attacks could be considered to explore the ability to de-anonymize location information in other settings [40]. If these results question the ability of PPMs to obfuscate highly correlated information such as users' whereabouts, they can also help design more efficient PPMs [14, 15, 16] and may encourage the use of distributed solutions in which users store maps and the related information directly on their mobile devices.

**Acknowledgements.** We would like to thank Selma Chouaki for helping with simulations.

## References

1. Aki, A.: The discovery of a lifetime., <http://www.aka-aki.com>
2. Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing* 7(5), 275–286 (2003)
3. Barnes, R., Cooper, A., Sparks, R., Jennings, C.: IETF geographic location/privacy, <http://www.ietf.org/dyn/wg/charter/geopriv-charter.html>
4. Beresford, A.R., Stajano, F.: Location privacy in pervasive computing. *Pervasive Computing, IEEE* 2(1), 46–55 (2003)
5. Beresford, A.R., Stajano, F.: Mix zones: User privacy in location-aware services. In: PerSec (March 2004)
6. Bettini, C., Wang, X.S., Jajodia, S.: Protecting Privacy Against Location-Based Personal Identification. In: Jonker, W., Petković, M. (eds.) SDM 2005. LNCS, vol. 3674, pp. 185–199. Springer, Heidelberg (2005)

7. Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., Scott, J.: Impact of human mobility on opportunistic forwarding algorithms. *IEEE TMC* 6, 606–620 (2007)
8. Cloudmade. Makes maps differently, <http://cloudmade.com>
9. Dalenius, T.: Finding a needle in a haystack - or identifying anonymous census records. *Journal of Official Statistics* 2(3), 329–336 (1986)
10. 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)
11. Eagle, N., Pentland, A., Lazer, D.: Inferring social network structure using mobile phone data. *National Academy of Sciences (PNAS)*, 15274–15278 (2009)
12. Foursquare. Check-in, find your friends, unlock your city, <http://foursquare.com>
13. Frejinger, E.: Route choice analysis: data, models, algorithms and applications. PhD thesis, EPFL (2008)
14. Freudiger, J., Manshaei, M.H., Boudec, J.-Y.L., Hubaux, J.-P.: On the age of pseudonyms in mobile ad hoc networks. In: *Infocom* (2010)
15. Freudiger, J., Manshaei, M.H., Hubaux, J.-P., Parkes, D.C.: On non-cooperative location privacy: A game-theoretic analysis. In: *CCS* (2009)
16. Freudiger, J., Shokri, R., Hubaux, J.-P.: On the Optimal Placement of Mix Zones. In: Goldberg, I., Atallah, M.J. (eds.) *PETS 2009. LNCS*, vol. 5672, pp. 216–234. Springer, Heidelberg (2009)
17. Friedland, G., Sommer, R.: Cybercasing the joint: On the privacy implications of geo-tagging. In: *HotSec* (2010)
18. Golle, P., Partridge, K.: On the anonymity of home/work location pairs. In: *Pervasive* (2009)
19. Google Mobile Blog. Finding places “near me now” is easier & faster than ever (2010), <http://googlemobile.blogspot.com/2010/01/finding-places-near-me-now-is-easier.html>
20. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: *MobiSys* (2003)
21. Hoh, B., Gruteser, M.: Protecting location privacy through path confusion. In: *SECURECOMM* (2005)
22. Hoh, B., Gruteser, M., Herring, R., Ban, J., Work, D., Herrera, J.-C., Bayen, A.M., Annavaram, M., Jacobson, Q.: Virtual trip lines for distributed privacy-preserving traffic monitoring. In: *MobiSys* (2008)
23. Hoh, B., Gruteser, M., Xiong, H., Alrabady, A.: Enhancing security and privacy in traffic-monitoring systems. *Pervasive Computing*, 38–46 (2006)
24. Hoh, B., Gruteser, M., Xiong, H., Alrabady, A.: Preserving privacy in GPS traces via uncertainty-aware path cloaking. In: *CCS* (2007)
25. Kiukkonen, N., Blom, J., Dousse, O., Gatica-Perez, D., Laurila, J.: Towards rich mobile phone datasets: Lausanne data collection campaign. In: *ICPS* (2010)
26. Krumm, J.: Inference attacks on location tracks. In: *Pervasive* (2007)
27. Kullback, S., Leibler, R.A.: On information and sufficiency. *The Annals of Mathematical Statistics* 22(1), 79–86 (1951)
28. Liao, L., Fox, D., Kautz, H.: Location-based activity recognition using relational Markov networks. In: *IJCAI* (2005)
29. Liao, L., Patterson, D.J., Fox, D., Kautz, H.: Learning and inferring transportation routines. *Artificial Intelligence* (171), 311–331 (2007)
30. Loopt. Discover the world around you, <http://loopt.com>
31. Ma, C.Y.T., Yau, D.K.Y., Yip, N.K., Rao, N.S.V.: Privacy vulnerability of published anonymous mobility traces. In: *MobiCom* (2010)
32. Mokbel, M.F., Chow, C.-Y., Aref, W.G.: The new casper: Query processing for location services without compromising privacy. In: *VLDB* (2006)

33. Mulder, Y.D., Danezis, G., Batina, L., Preneel, B.: Identification via location-profiling in GSM networks. In: WPES (2008)
34. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: Security and Privacy (2009)
35. Piorkowski, M.: Sampling urban mobility through on-line repositories of GPS tracks. In: HotPlanet (2009)
36. Piorkowski, M., Sarafijanovic-Djukic, N., Grossglauser, M.: A parsimonious model of mobile partitioned networks with clustering. In: ComsNets, pp. 1–10 (2009)
37. Roth, C., Kang, S.M., Batty, M., Barthelemy, M.: Commuting in a polycentric city. Technical report, CNRS (2010)
38. 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)
39. Shokri, R., Freudiger, J., Hubaux, J.-P.: A unified framework for location privacy. In: HotPETs (2010)
40. Shokri, R., Theodorakopoulos, G., Boudec, J.-Y.L., Hubaux, J.-P.: Quantifying Location Privacy. In: IEEE S&P (2011)
41. Sweeney, L.: k-anonymity: A model for protecting privacy. Uncertainty, Fuzziness and Knowledge-based systems 10, 557–570 (2002)
42. Vojnovic, M., Boudec, J.-Y.L.: Perfect simulation and stationarity of a class of mobility models. In: Infocom (2005)
43. Zhong, S., Li, L.E., Liu, Y.G., Yang, Y.R.: Privacy-preserving location-based services for mobile users in wireless networks. Technical report, SUNY at Buffalo (2005)

# Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance<sup>\*</sup>

Jeremy Clark and Urs Hengartner

University of Waterloo

{j5clark,uhengart}@cs.uwaterloo.ca

**Abstract.** We present Selections, a new cryptographic voting protocol that is end-to-end verifiable and suitable for Internet voting. After a one-time in-person registration, voters can cast ballots in an arbitrary number of elections. We say a system provides over-the-shoulder coercion-resistance if a voter can undetectably avoid complying with an adversary that is present during the vote casting process. Our system is the first in the literature to offer this property without the voter having to anticipate coercion and precompute values. Instead, a voter can employ a panic password. We prove that Selections is coercion-resistant against a non-adaptive adversary.

## 1 Introductory Remarks

From a security perspective, the use of electronic voting machines in elections around the world continues to be concerning. In principle, many security issues can be allayed with cryptography. While cryptographic voting has not seen wide deployment, refined systems like Prêt à Voter [11,28] and Scantegrity II [9] are representative of what is theoretically possible, and have even seen some use in governmental elections [7]. Today, a share of the skepticism over electronic elections is being apportioned to Internet voting.<sup>1</sup> Many nation-states are considering, piloting or using Internet voting in elections. In addition to the challenges of verifiability and ballot secrecy present in any voting system, Internet voting adds two additional constraints:

- Untrusted platforms: voters should be able to reliably cast secret ballots, even when their devices may leak information or do not function correctly.
- Unsupervised voting: coercers or vote buyers should not be able to exert undue influence over voters despite the open environment of Internet voting.

As with electronic voting, cryptography can assist in addressing these issues. The study of cryptographic Internet voting is not as mature. Most of the literature concentrates on only one of the two problems (see related work in Section 1.2). In this paper, we are concerned with the unsupervised voting problem. Informally, a system that solves it is said to be *coercion-resistant*.

---

<sup>\*</sup> Full version available: <http://eprint.iacr.org/2011/166>

<sup>1</sup> One noted cryptographer, Ronald Rivest, infamously opined that “best practices for Internet voting are like best practices for drunk driving” [25].

## 1.1 Contributions

Coercion-resistant, end-to-end verifiable Internet voting systems have been proposed [1,4,14,24,31,33]. However, these systems all require the voter to remember cryptographic information after registration. Since the information is too long to memorize, authentication can be considered to be based on “something you have.” Voters must prepare for the possibility of coercion by creating fake values, proofs, or transcripts. Our system works with passwords, “something you know,” and it allows a voter to supply a panic password during ballot casting that can be created mentally in real-time by the voter. In summary, our system provides:

- Password-based authentication and cognitive coercion-resistance,
- In-person registration that can be performed bare-handed,
- Tallying that is linear in the number of voters, and
- Efficient revocation of voters from the roster during and between elections.

We compare Selections to three systems: JCJ [24], Civitas [14], and AFT [4] (see Section 1.2). Of these properties, only Selections meets each while AFT achieves the third and both JCJ and Civitas achieve the fourth.

## 1.2 Related Work

The field of cryptographic voting is mature, and proposals for new systems should be soundly motivated. Our system addresses the problem of coercion and vote selling when voters are not required to vote in a private booth. Only a small number of the most recent papers in cryptographic voting address this threat.

Coercion-resistance was first formalized by Juels *et al.* [24], who also provide a coercion-resistant system, often referred to as JCJ. JCJ was independently implemented as Civitas [14]. The main drawback of both is that tallying is quadratic in the number of voters. Aquisti [1] refined JCJ to use Paillier encryption and support write-in candidates, while both Smith [31] and Weber *et al.* [33] made the first attempts at reducing the complexity of tallying to linear. Unfortunately, all three are considered broken [4,14,5]. More recently (concurrent with Selections), Spycher *et al.* have proposed a different approach to making JCJ linear [32].

Araujo *et al.* provide a linear-time system we refer to as AFT [4]. Both JCJ/Civitas and AFT provide registered voters with anonymous credentials. A voter submits a credential along with her vote and a procedure for computing a fake credential is provided (but cannot be done without a computer). In JCJ/Civitas, the credentials of registered voters are posted and these are anonymously and blindly compared to the credential accompanying each submitted vote. In AFT, the credentials of registered voters are essentially signed and the presence of a valid signature on a credential submitted during casting is anonymously and blindly checked. Due to the difficulty of revoking a signed value, voters cannot be revoked in AFT without a change of cryptographic keys.

Some Internet systems are designed for low-coercion elections. These include Helios [2], which was used in a binding university election [3]. Other Internet voting systems concentrate on the untrusted platform issue. A common approach

is “code voting,” where acknowledgement codes are returned to voters upon receipt of a vote. The codes are a function of the vote and not known in advance to the network carrier. This principle can be seen in SureVote [8], CodeVoting [23], Pretty Good Democracy [29], and Heiberg *et al.* [18].

## 2 Preliminaries

### 2.1 Selections: High-Level Overview

Selections is a protocol designed to allow voters to cast ballots over the Internet during a window of time prior to traditional in-person voting. Voters can opt out of Selections at any time prior to election day and cast a ballot in-person.

To be eligible for Selections, voters first complete a one-time, in-person registration protocol in a private booth without needing her own computational device. After this registration, the voter can vote in future elections over a tapable channel (see Section 2.3). The registration involves the voter choosing a password to be used for vote casting. However this password is non-traditional—it is a password from a panic password system (see Section 2.5). A semantically-secure homomorphic encryption of this password is posted on a public roster. The roster has an entry for each registered voter containing this ciphertext. The voter must be convinced that her entry is a correct encryption without being able to prove what it encrypts to anyone.

During vote submission, the voter asserts what her password is: it may be her actual password or a panic password. The voter creates a binding commitment to this asserted password. The voter then rerandomizes her entry off the roster. The voter proves in zero-knowledge the latter ciphertext is a re-encryption of some random subset of passwords off the public roster, without revealing which one. The commitment to her asserted password, re-encrypted roster entry, proof (and some additional proofs that things are well-formed), and an encryption of her vote are submitted over an anonymous channel to a public bulletin board.

When the voting period expires, a distributed group of trustees will eliminate submissions with invalid proofs, eliminate duplicate votes based on the password commitment, and then use a verifiable mix network to shuffle the order of the remaining submissions. After shuffling, voters can no longer determine where their submission is in the new permuted list. For each submission, the trustees will determine if the asserted password matches the roster entry without revealing either. If it does not, the entry is eliminated. The output of Selections is a list of encrypted votes from registered voters without duplicates. The entire protocol can be verified for soundness.

### 2.2 Coercion-Resistance

Informally, Juels *et al.* define coercion-resistance as providing receipt-freeness, while preventing three attacks: randomization, abstention, and simulation [24].

A voting system is said to be receipt-free if the voter cannot produce a transcript that constitutes a sound argument for how they voted. Adversaries should not be able to force a registered voter to cast a random vote or to abstain from voting. Finally, the system should protect against voters surrendering their credentials and allowing a coercer or vote buyer to cast their vote for them. The dominant approach to preventing such a simulation is providing voters with the ability to create fake credentials. If an adversary cannot distinguish a real credential from a fake one, he will only be willing to pay what a fake credential is worth, which is nothing.

### 2.3 Untappable Channels

The main challenge for coercion-resistant Internet voting is dealing with the elimination of the private voting booth, modelled as an untappable channel. One approach is to use multiple secure channels and assume that while any individual channel can be tapped, no adversary can tap all channels simultaneously. The second is to use an untappable channel just once, and bootstrap the output of this interaction into an arbitrary number of future interactions over secure (or anonymous) channels. We use the latter approach.

### 2.4 Registration Authority

In most coercion-resistant Internet voting systems, voters interact with a distributed registration authority [1,4,24]. To achieve coercion-resistance, it is assumed that at least one registrar is not corrupted by the adversary. Voters may be corrupted to retain a transcript, however the transcript has deniability by using a designated verifier proof [21].

While distributing trust is usually an effective approach for achieving correctness and secrecy in a protocol, it is more complex with coercion-resistance. The voter must be aware of which entity she trusts, so she can fake a proof that will not be compared to the original. If the voter discloses her private key to an adversary, it only requires a single malicious registrar to collude with the adversary and undetectably issue the voter an incorrect credential share (while retaining the correct value for potential adversarial use).

These concerns leave it unclear if the benefits of a distributed registration authority are worthwhile. While Selections is amenable to a distributed registration authority (voters would submit encryptions of shares of their password, which are homomorphically combined to create an encryption of the password), we describe the protocol using a single registrar that is assumed to not collude with a coercer (but may still misbehave in any other regard).

### 2.5 Panic Passwords

A panic password system [12] initializes three categories of passwords: a password, a set of panic passwords, and the residual set of inadmissible passwords.

From the user’s view, submission of a password or a panic password is indistinguishable, while an inadmissible password will prompt the user to try again. If the user registers a password and one panic password, an adversary can demand two distinct admissible passwords and submit the coerced vote with each—therefore, the number of panic passwords should be arbitrarily large to prevent these “iteration” attacks. If a user registers a password and all other values are panic passwords, an accidental mistyping will result in the vote being discarded—therefore, the distance between admissible and inadmissible passwords should be maximized. Finally, with an arbitrarily large number of panic passwords distributed sparsely among inadmissible passwords, set-membership tests for panic passwords should be cognitively easy to perform.

Clark and Hengartner propose the 5-Dictionary panic password system to meet these requirements [12]. Admissible passwords consist of five words from an agreed upon dictionary: the user chooses one combination as her password and any other combination is a panic password. A typo is likely to mutate the intended word into a string not found in the dictionary. With the Unix dictionary of English words, this system offers up to 70 bits of entropy. The authors also propose the 5-Click alternative based on graphical passwords, and new panic password schemes could be developed based on, for example, preferences [22]. Voters would be free to choose which to use.

### 3 The Selections Protocol

Selections involves a set of voters, a set of election trustees, an election authority, and a registrant. The system has six main protocols: registration set-up, voter preparation, registration, election set-up, casting, and pre-tallying. Let  $\langle \text{DKG}, \text{Enc}, \text{DDec} \rangle$  be a threshold encryption scheme. Distributed key generation  $\text{DKG}(n, m)$  generates public key,  $e$ , and a private key share,  $d_i$ , for each of  $n$  trustees. Encryption,  $\text{Enc}_e(m, r)$ , is semantically secure and homomorphic with respect to one operation. Distributed decryption,  $\text{DDec}_{d_i}(c)$ , on ciphertext  $c$  can be performed with  $m + 1$  trustees submitting shares  $d_i$ .<sup>2</sup> We use threshold Elgamal [26].

#### 3.1 Registration Setup

The registration set-up protocol involves a set of  $n$  trustees:  $T_1, \dots, T_n$  and the election authority. Primes  $p$  and  $q$  are chosen such that the DL-problem and DDH-problem are hard in the multiplicative subgroup  $\mathbb{G}_q$  of  $\mathbb{Z}_p^*$ . Each  $T_j$  participates in  $\text{DKG}(n, m)$ . Commitments are sent to the election authority, who posts them to an append-only broadcast channel called the Bulletin Board. At the end of the protocol, each  $T_j$  has private key share  $d_j$  and public key  $e$  is posted. The protocol is standard and will not be described here [26].

---

<sup>2</sup> Proactive security can maintain the secrecy of the shares over time, both the number of shares and the threshold can be adjusted without a dealer, and more a complex access structure than  $m$ -out-of- $n$  can be created.

### 3.2 Voter Preparation

The voter preparation procedure is performed by each voter  $\mathcal{V}_i$  on a trusted computational client. Let  $\langle P, I \rangle$  be the domain of a panic password system.  $P$  represents the set of admissible passwords and  $I = \neg P$  is the set of inadmissible passwords.  $\mathcal{V}_i$  chooses a password  $\hat{\rho}$ . The client runs  $\text{PassSubmit}(\hat{\rho})$ , which tests if  $\hat{\rho} \in P$ . If  $\hat{\rho} \in I$ ,  $\text{PassSubmit}(\hat{\rho})$  returns an error. The set of panic passwords are the remaining passwords in  $P$ :  $\{\forall \hat{\rho}^* \in P | \hat{\rho}^* \neq \hat{\rho}\}$ .  $\text{PassSubmit}(\hat{\rho}^*)$  will behave identically upon submission of a panic password (otherwise an adversary could distinguish the case where he is given a panic password).

Once  $\text{PassSubmit}(\hat{\rho})$  accepts  $\hat{\rho}$ , the client encodes  $\hat{\rho}$  as a bitstring and appends a non-secret salt to prevent accidental collisions with other users. This string is supplied as input to a password-based key derivation function (PBKDF) for strengthening and encoding into  $\mathbb{Z}_q$ . For brevity, we denote this entire password processing procedure as  $\phi$ :  $\rho \leftarrow \phi(\hat{\rho}) = \text{PBKDF}(\text{PassSubmit}(\hat{\rho}) \parallel \text{salt})$ .

Perhaps through a user-guided tutorial familiarizing the voter with the system, the voter will generate  $\alpha$  admissible passwords:  $\hat{\rho}_1, \dots, \hat{\rho}_\alpha$ . The value of  $\alpha$  will determine the soundness of the registration protocol. An example value for  $\alpha$  is 10. The password the voter wishes to register is in a random location in the list. Each is encrypted by the voter under the trustees' public key  $e$ . The voter prints out the list of ciphertexts on to a piece of paper, *e.g.*, with the ciphertexts encoded into barcodes. The registration protocol in Algorithm 1 includes the voter preparation protocol.

### 3.3 Registration

The registration protocol (Algorithm 1) is completed by each voter  $\mathcal{V}_i$ . It is a two-party cut-and-choose protocol between a voter  $\mathcal{V}_i$  and the registrar  $\mathcal{R}$ . It is an adaptation of the Benaloh's voter initiated auditing [6], with a predetermined number of challenges. The voter enters the protocol with a list of  $\alpha$  encrypted passwords  $\{c_1, \dots, c_\alpha\}$  and the protocol completes with a re-encryption of one of the  $\rho$ 's being posted to an append-only broadcast channel, called the Roster. The protocol itself is conducted over an untappable channel which is instantiated as an in-person protocol.

The voter presents identification and is authorized to register. The voter is given a blank transcript card and enters a private booth that has a computer in it capable of printing and scanning barcodes. A transcript card has  $\alpha$  rows and two columns. The second column for each row has a scratch-off surface. The voter is provided the option of downloading and printing a document from the Internet—with the intention that the voter could print her voter preparation sheet in the event that an adversary ensured she entered the registration process without her sheet. The computer has a barcode scanner, which the voter uses to submit her  $\alpha$  ciphertexts.

The computer will rerandomize each ciphertext and print the value in the first column of the transcript card. Beside this value on the scratch-off surface, it will print the original ciphertext and the randomization used. The voter chooses one

**Algorithm 1.** Registration Protocol

---

**Participants** : Voter  $\mathcal{V}_i$  and registrant  $\mathcal{R}$

**Public Input:** Encryption parameters  $p, q, g$ , public key  $e$ , and soundness parameter  $\alpha > 1$

**Private Input ( $\mathcal{V}_i$ ):** Ciphertexts  $\{c_1, \dots, c_\alpha\}$  as described below

**Prior to the protocol, each voter should:**

- 1    **for**  $k$  from 1 to  $\alpha$  **do**
- 2     Choose a password  $\hat{\rho}_k$ .
- 3     Process password:  $\rho_k \leftarrow \phi(\hat{\rho}_k)$ .
- 4     Encrypt  $g^{\rho_k}$  with random  $r_k$ :  $c_k \leftarrow \text{Enc}_e(g^{\rho_k}, r_k)$ .
- 5     Complete a NIZKP of knowledge of plaintext  $g^{\rho_k}$ :  

$$\pi_k \leftarrow \text{NIZKP}_{pok}\{(\rho_k, r_k) : c_k = \text{Enc}_e(g^{\rho_k}, r_k)\}.$$
- 5     Record  $\langle c_k, \pi_k \rangle$ .
- end

**Registrar should:**

- 6    Receive  $\{\langle c_1, \pi_1 \rangle, \dots, \langle c_\alpha, \pi_\alpha \rangle\}$ .
- 7    **for**  $k$  from 1 to  $\alpha$  **do**
- 8     Check  $\pi_k$ .
- 9     Rerandomize  $c_k$  with random  $r'_k$ :  $c'_k \leftarrow \text{ReRand}(c_k, r'_k)$ .
- 10    Print  $\langle c'_k, (c_k, r'_k) \rangle$ .
- end

**Each voter should:**

- 11    Receive for each  $k$ :  $\langle c'_k, (c_k, r'_k) \rangle$ .
- 12    Optionally, rewind to line 1.
- 13    Choose  $s \leftarrow [1, \alpha]$ .
- 14    Erase  $(c_s, r'_s)$ .
- 15    Send  $s$  to  $\mathcal{R}$ .
- end

**Registrar should:**

- 16    Receive  $s$ .
- 17    Publish  $\langle \text{VoterID}, c'_s \rangle$  on the Roster.
- end

**Each voter should:**

- 18    After leaving, check that  $c'_k \leftarrow \text{ReRand}(c_k, r'_k)$  for all  $k \neq s$ .
- 19    Check that received  $c'_s$  matches  $\langle \text{VoterID}, c'_s \rangle$  on the Roster.
- end

*Remarks:* This protocol is completed *bare-handed* [27] with pre-computations and erasures. The proof of knowledge of an Elgamal plaintext is standard. The option to rewind is included to prevent coercion contracts [13].

---

password to register: for that password, the voter will erase the original ciphertext and randomization by scratching off the appropriate cell.<sup>3</sup> It is assumed

<sup>3</sup> Under each scratch-off could be a pre-committed code in the form of a barcode, which the voter could scan to prove to the system that she scratched off the correct cell. We leave the details for such an augmented transcript card for future work.

the voter cannot memorize or copy the randomization (*e.g.*, it is encoded into a barcode). The voter shreds her preparation sheet and retains the transcript card. The remaining  $\alpha - 1$  re-encryptions can be shown to anyone and checked for correctness at home.

### 3.4 Election Set-Up

The Roster is a universal registration. To prepare for an election, entries from the Roster are copied to smaller lists, called **ElectionRosters**. An **ElectionRoster** is specific to a particular election, precinct or district. The trustees will also modify the encrypted message in each entry from  $g^\rho$  to  $g_0^\rho$ , where  $g_0$  is a unique publicly-known generator for that election. This prevents information leakage across elections.

Recall that Roster entries are encrypted with  $\rho$  in the exponent:  $\{c_1, c_2\} = \{g^r, g^\rho y^r\}$ . For each **ElectionRoster**, each trustee chooses  $b_i \leftarrow_r \mathbb{G}_q$ . Then each trustee will in turn blind each ciphertext on the **ElectionRoster** as follows: output  $g^{b_i}, c_1^{b_i}$  and  $c_2^{b_i}$ , and prove knowledge of  $b_i$  such that  $g, c_1, c_2, g^{b_i}, c_1^{b_i}, c_2^{b_i}$  form a threewise DH-tuple with a NIZKP (*cf.* [10]). The next trustee will repeat the process using the previous trustee's output as input. All outputs are posted to an appendix on the **ElectionRoster**. Let  $b_0 = \prod b_i$  and  $g_0 = g^{b_0}$ . The blinding sequence re-randomizes each ciphertext from  $r$  to  $r' = r \cdot b_0$  and changes the encrypted message from  $g^\rho$  to  $g_0^\rho$ . The public and private key shares are the same. The public value  $g_0$  will be used during the casting protocol.

### 3.5 Casting

The casting protocol involves a voter  $\mathcal{V}_i$  and the election authority. The protocol is described in Algorithm 2. The communication occurs over an anonymous channel. The anonymity is to be built into the voter's client using an anonymous remailer or onion routing technology.

$\mathcal{V}_i$  submits a commitment to her asserted (*i.e.*, real or panic) password,  $g_0^{\rho^*}$ , and a rerandomization of her entry on the **ElectionRoster**,  $c'$ . If  $\rho^*$  matches the  $\rho$  encrypted in  $c'$ , the pre-tallying protocol will ensure the ballot is included in the final result. Otherwise if it does not match, it will be discarded in a way that is unlinkable to the original submission.

$\mathcal{V}_i$  must prove that  $c'$  is from the **ElectionRoster**. Simply including her entry without rerandomizing it reveals that she submitted a vote. To prevent abstention attacks, she instead rerandomizes it, draws an additional  $\beta - 1$  entries randomly from the **ElectionRoster**, and proves in zero-knowledge that  $c'$  is a rerandomization of one of these  $\beta$  entries (her entry plus the additional ones).  $\beta$  acts as an anonymity set. Most voters will use a small value of  $\beta$ , however privacy-conscious voters can also (at extra computational cost) cast a **stealth** vote where  $\beta$  includes all the entries on the **ElectionRoster**.

**Algorithm 2.** Casting Protocol

---

**Participants** : Voter  $\mathcal{V}_i$  and election authority

**Public Input:** Encryption parameters  $g, p, q$ , election parameter  $g_0$ , public key  $e$ , ElectionRoster, and anonymity parameter  $\beta$ 
**Private Input** ( $V_i$ ): Password (either real or panic)  $\hat{\rho}^*$ 
**Each voter should:**

- 1 Find  $c$  for her VoterID from ElectionRoster.
- 2 Rerandomize  $c$  with random  $r$ :  $c' \leftarrow \text{ReRand}(c, r)$ .
- 3 Randomly select  $\beta$ -1 other  $c_k$  from the ElectionRoster.
- 4 Form set  $\mathcal{C} = \{c, c_1, \dots, c_{\beta-1}\}$  in order of appearance on ElectionRoster.
- 5 Generate a NIZKP that  $r$  rerandomizes 1-out-of- $\beta$  of  $\mathcal{C}$ .  
 $\pi_1 \leftarrow \text{NIZKP}_{pok}\{(r) : c' = (\text{ReRand}(c, r) \vee \text{ReRand}(c_1, r) \vee \dots)\}$ .
- 6 Encode asserted password into  $\mathbb{Z}_q$ :  $\rho^* \leftarrow \phi(\hat{\rho}^*)$ .
- 7 Commit to  $\rho^*$ :  $g_0^{\rho^*}$ .
- 8 Complete an NIZKP of knowledge of  $\rho^*$ :  
 $\pi_2 \leftarrow \text{NIZKP}_{pok}\{(\rho^*) : g_0, g_0^{\rho^*}\}$ .
- 9 Complete a ballot and retain ballot information  $\mathbf{B}$ .
- 10 Send  $\langle g_0^{\rho^*}, c', \mathbf{B}, \pi_1, \pi_2 \rangle$  to  $A$ .

**end**
**Authority should:**

- 11 Publish  $\langle g_0^{\rho^*}, c', \mathbf{B}, \pi_1, \pi_2 \rangle$  on AllVotes.

**end**

*Remarks:* Rerandomization proofs are formed with a knowledge of a DDH-tuple proof due to Chaum and Pedersen [10]. 1-out-of-m proofs are due to a heuristic by Cramer, Damgard and Schoenmakers [15]. Proof of knowledge of a discrete log is due to Schnorr [30]. Parameter  $\beta$  represents the voter's anonymity set.

---

Selections is designed to be versatile with different options for capturing and tallying the votes themselves. Thus we leave the information the voter submits with regard to their vote abstractly as  $\mathbf{B}$  while only requiring that  $\mathbf{B}$  is submit-table to a mix-network. For example,  $\mathbf{B}$  could be an encryption of the preferred candidate(s) or a tuple of cryptographic counters for each option, accompanied by proofs of validity as appropriate. Note that our coercion-resistance guarantee extends only to the delivery of valid, eligible, and unique  $\mathbf{B}$  values, and care should be taken to ensure that tallying these values does not break coercion-resistance.

Each ZKP uses the Fiat-Shamir heuristic to make it non-interactive, and each uses the values  $\langle g_0^{\rho^*}, c', \mathbf{B} \rangle$  in creating the challenge. This prevents an adversary from replaying any of the proofs individually. The submission is posted to an append-only broadcast channel called AllVotes.

If the voter is under coercion, she makes up a panic password and follows the rest of the protocol as specified. She can later cast a stealth vote with her real password. If a voter wants to overwrite a previous vote submitted under

---

**Algorithm 3.** Pre-Tallying Protocol

---

**Participants** : Authorized set of trustees  $T_1, \dots, T_m$  and election authority

**Public Input:** AllVotes

**Private Input ( $T_i$ ):** Share of private key,  $d_i$

**Authority should:**

- 1 For each entry, check  $\pi_1$  and  $\pi_2$ .
- 2 Remove all tuples with invalid proofs to form list ProvedVotes
- 3 Find all entries in ProvedVotes with duplicate values for  $g_0^\rho$ .
- 4 Remove all but the most recent to form list UniqueVotes.

end

**Each participating trustee should:**

- 5 Participate in verifiable mix network for shuffling UniqueVotes.  
Note: the initial  $g_0^{\rho^*}$  is treated as  $c_\rho = \text{Enc}_e(g_0^{\rho^*}, 0)$ .
- 6 Output is AnonUniqueVotes.

end

**Each participating trustee should:**

- 7 **for each entry in AnonUniqueVotes do**
- 8     Read entry  $\langle c_\rho, c', \mathbf{B} \rangle$ .
- 9     Participate in a plaintext-equality test of  $c_\rho$  and  $c'$ :  
 $\{T, F\} \leftarrow \text{PET}_{d_i}(c_\rho, c')$ .

end

**Authority should:**

- 10 Remove all tuples with PET outcome of **False** to form list ValidVotes.

end

**Each participating trustee should:**

- 11 **for each entry in ValidVotes do**
- 12     Participate in threshold decryption of  $\mathbf{B}$ .

end

*Remarks:* Various protocols exist for verifiable mix networks. An efficient technique with statistical soundness is randomized partial checking [19]. The plaintext equality test (PET) is due to Juels and Jakobsson [20]. The output of this protocol is the ballot information for unique and registered voters in an order that is unlinkable to the order of submission.

---

password  $\rho^*$ , the inclusion of the same  $g_0^{\rho^*}$  will indicate in cleartext that it is an overwrite. Therefore, she should use the same  $\beta$  entries from the ElectionRoster as her anonymity set. Also note that the inclusion of the same  $g_0^{\rho^*}$  across multiple elections would also be linkable if the value  $g_0$  was not changed in each election.

### 3.6 Pre-tallying

The pre-tallying protocol (Algorithm 3) involves an authorized subset of trustees. The protocol takes AllVotes and produces a shorter list of only the most recently cast votes for voters that supply the correct, registered password. Checking the validity of each vote is linear in  $\beta$ . For these voters, the list includes just the ballot information,  $\mathbf{B}$ , in an order that is unlinkable to the order of submission. How

**Table 1.** Comparison of the efficiency of the main protocols in Civitas, AFT, and Selections, measured with modular exponentiations.

		Civitas	AFT	Selections
Registration	Registrar	7	9	$2\alpha$
	Voter	11	10	$4\alpha-1$
Casting	Voter	10	24	$(2\beta + 9)$
Pre-Tally	Check Proofs	$4V_0$	$20V_0$	$(4\beta + 6)V_0$
	Remove Duplicates	$(1/2)(V_1^2 - V_1)(8T + 1)$	—	—
	Check Removal	$(1/2)(V_1^2 - V_1)(8T + 1)$	—	—
	Mix	$8V_2T + 4RT$	$20V_2T$	$12V_2T$
	Check Mix	$4V_2T + 2RT$	$10V_2T$	$6V_2T$
	Remove Unregistered	$(8A + 1)V_2R$	$(16T + 8)V_2$	$(8T + 1)V_2$
	Check Removal	$(8A + 1)V_2R$	$(16T + 10)V_2$	$(8T + 1)V_2$

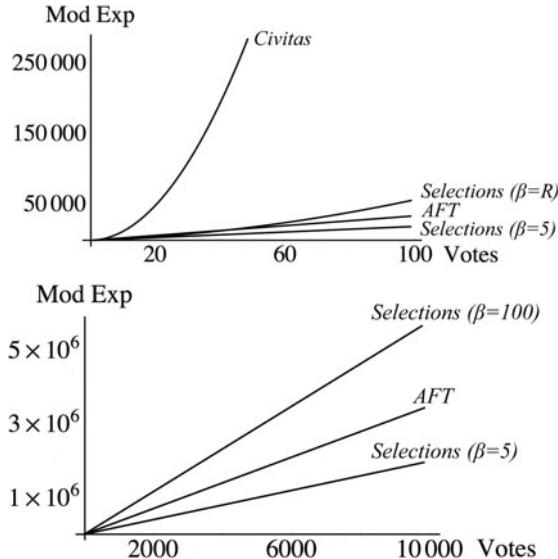
this list is further processed to produce a tally is dependent on the voting system our system interfaces with (which is why this is called a pre-tally). In a simple case,  $\mathbf{B}$  is an encryption of the voter’s selections (with a proof of correctness) and the final step is jointly-decrypting each  $\mathbf{B}$  from the list.

### 3.7 Voter Revocation

Between elections, Selections offers a way of choosing which registered voters are eligible or not to vote in a particular election. In Selections, it is also possible to revoke a voter at any point before the pre-tallying protocol. This could arise because the voter forgot their password (and is issued a new one) or registered to vote online but decides to vote in person. For every submitted vote that includes the revoked voter among its  $\beta$  registered voters in its anonymity set (which will include any potentially valid vote by the revoked voter herself), the submitted password is checked against the revoked voter’s entry on the `ElectionRoster` using a plaintext-equality test. Revocation of this type is the same in Civitas and is not possible in AFT. Coercion-resistance does not necessarily extend to all types of revocation.

## 4 Performance

We compare the performance of Selections to JCJ as implemented in Civitas [14] and to AFT [4]. We make a number of standardizing assumptions to facilitate a better comparison. We assume a single registrar,  $T$  trustees,  $R$  registered voters, and  $V_0$  submitted votes. We do not use the “blocking” technique of Civitas, which could improve the performance of all three systems. Of the  $V_0$  submitted votes,  $V_1 \leq V_0$  have correct proofs,  $V_2 \leq V_1$  are not duplicates, and  $V_3 \leq V_2$  correspond to registered voters. Recall that for Selections,  $\alpha$  are the number of submitted ciphertexts in registration and  $\beta$  is the size of the voter’s anonymity set during casting.



**Fig. 1.** Pre-tallying efficiency in modular exponentiations with  $T = 5$  and variable  $R = V_0 = V_1 = V_2$

We use Elgamal encryption in each system, with proofs of knowledge of plaintexts where appropriate. We assume each trustee participates in decryption (*i.e.*, distributed instead of threshold). We assume that ballot material is encrypted with only a proof of knowledge (no additional proofs of well-formedness). The pre-tallying protocol ends with a list of  $V_3$  encrypted ballots. Finally, we assume mixing is done with a re-encryption mixnet and randomized partial checking [19], where each authority produces two mixes and half of these re-encryptions are checked. The complete details of our comparison are in the full paper.<sup>4</sup>

Table 1 shows the efficiency in terms of modular exponentiations and Figure 4 shows a comparison of the pre-tallying protocols. With full forced-abstention, Selections is quadratic like Civitas but with a smaller constant. When  $\beta$  is a constant, Selections is linear in the number of submitted votes like AFT. The exact value of  $\beta$  dictates which is exactly faster. Recall our goal was not to improve the efficiency of AFT but rather to create a password-based system with similar performance to AFT. To this end, we are successful.

## 5 Security Analysis (Abstract)

### 5.1 Soundness of Registration

In the full paper,<sup>4</sup> we show that the Registration protocol is a cut-and-choose argument for  $\{(c, r) : c' = \text{ReRand}_e(c, r)\}$ . It takes soundness parameter

<sup>4</sup> <http://eprint.iacr.org/2011/166>

$\alpha$  (e.g.,  $\alpha = 10$ ). It is complete and has statistical soundness of  $1 - \alpha^{-1}$  for a single run. After  $k$  runs, soundness increases to  $1 - \alpha^{-k}$ . Designing a bare-handed argument with stronger soundness (e.g.,  $1 - 2^{-\alpha}$  for a single run) is open. With erasures, the protocol has deniability for  $c$  and computational secrecy for  $r$ .

The protocol does not protect against covert channels. This has been addressed in the literature with verifiable random functions [17] or pre-committed randomness [16]. The protocol protects against coercion contracts [13] with rewinds. Rewinds can be eliminated if the voter commits to their choice of password at the beginning of the protocol.

## 5.2 Coercion-Resistance

In the full paper,<sup>4</sup> we show several results concerning the coercion-resistance (cr) of Selections. Juels *et al.* define an experiment  $\text{Exp}_{ES,\mathcal{A}}^{\text{cr}}$  for non-adaptive adversary  $\mathcal{A}$  in election system  $ES$ , as well as an ideal  $\text{Exp}_{ES,\mathcal{A}}^{\text{cr-ideal}}$ . The critical component in  $\text{Exp}_{ES,\mathcal{A}}^{\text{cr}}$  is a coin flip  $b \leftarrow_r \{0, 1\}$  defining a corrupted voter's behaviour. If  $b = 0$ , the voter provides (in Selections) a panic password to the adversary and casts a vote with her real password. If  $b = 1$ , the voter complies with the adversary and provides her real password. In both cases, the adversary can use the supplied password to submit a vote. We define the advantage of  $\mathcal{A}$ , where an output of 1 is the adversary correctly stating  $b$ , as,

$$\text{adv}_{ES,\mathcal{A}}^{\text{cr}} = |\Pr[\text{Exp}_{ES,\mathcal{A}}^{\text{cr}}(\cdot) = 1] - \Pr[\text{Exp}_{ES,\mathcal{A}}^{\text{cr-ideal}}(\cdot) = 1]|.$$

*Case 1:  $\beta = R$ .* We show that when  $\beta$  is the full roster  $R$ ,  $\text{adv}_{ES,\mathcal{A}}^{\text{cr}}$  for Selections is negligible. Setting  $\beta = R$  does impact performance. Vote casting is linear in the size of the ElectionRoster and Pre-Tallying is quadratic. However the only quadratic component is checking the 1-out-of- $\beta$  rerandomization proof, where the proof length is linear in the size of the roster. These proofs can be pre-checked, while voters submit votes.

*Case 2:  $\beta = \text{const}$ .* We show that when  $\beta$  is constant (e.g., 5 or 100),  $\text{adv}_{ES,\mathcal{A}}^{\text{cr}} < \delta$ , where  $\delta$  is small but non-negligible. Recall there are  $V_2$  votes with valid proofs and  $R$  entries on the ElectionRoster. Let  $\mathbf{F}(k; p, n)$  be the cumulative distribution function of a Binomial distribution with  $n$  trials, success probability  $p$ , and  $k$  successes. We show that  $\delta$  for this case is,

$$\delta = \frac{1}{2} \left( F\left(\frac{\beta V_2}{R}; V_2, \frac{\beta}{R}\right) + 1 - F\left(\frac{\beta V_2}{R} - 1; V_2 - 1, \frac{\beta}{R}\right) \right).$$

*Case 3:  $\beta \geq \text{const}$ .* Finally we consider the case where  $\beta$  is required to be at least a constant value (e.g., 5 or 100) but voters can submit stealth votes where  $\beta = R$ . We show that if a corrupted voter's coercion-resistant strategy is to submit their real vote as a stealth vote,  $\text{adv}_{ES,\mathcal{A}}^{\text{cr}}$  is negligible. We do make one small change to  $\text{Exp}_{ES,\mathcal{A}}^{\text{cr}}$ : instead of the corrupted voter's real vote being appended to the cast ballots, it is inserted at a random place (i.e., she votes her real ballot at some arbitrary time after being coerced).

## 6 Concluding Remarks

Selections has many benefits: users can evade coercion without computations, registration does not require a computer, tallying the votes is linear in the number of voters, and voters can have their registration efficiently revoked. Future work includes providing protection against untrusted platforms, perhaps by merging Selections with existing work on code voting.

**Acknowledgements.** We acknowledge Richard Carback for suggesting that panic passwords could be employed in an Internet voting system to prevent undue influence. We thank Aleks Essex, Michael Clarkson, various participants of SecVote 2010, and the reviewers for useful feedback. This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)—the first author through a Canada Graduate Scholarship and the second through a Discovery Grant.

## References

1. Acquisti, A.: Receipt-free homomorphic elections and write-in ballots. Tech. rep., IACR Eprint Report 2004/105 (2004)
2. Adida, B.: Helios: web-based open-audit voting. In: USENIX Security Symposium, pp. 335–348 (2008)
3. Adida, B., Marneffe, O.d., Pereira, O., Quisquater, J.J.: Electing a university president using open-audit voting: analysis of real-world use of Helios. In: EVT/WOTE (2009)
4. Araujo, R., Foulle, S., Traore, J.: A practical and secure coercion-resistant scheme for remote elections. In: Frontiers of Electronic Voting (2007)
5. Araújo, R., Ben Rajeb, N., Robbana, R., Traoré, J., Youssfi, S.: Towards Practical and Secure Coercion-Resistant Electronic Elections. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 278–297. Springer, Heidelberg (2010)
6. Benaloh, J.: Simple verifiable elections. In: EVT (2006)
7. Carback, R.T., Chaum, D., Clark, J., Conway, J., Essex, A., Hernson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II election at Takoma Park. In: USENIX Security Symposium (2010)
8. Chaum, D.: Surevote: Technical overview. In: WOTE (2001)
9. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T.: Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In: EVT (2008)
10. 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)
11. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
12. Clark, J., Hengartner, U.: Panic passwords: authenticating under duress. In: HotSec (2008)
13. Clark, J., Hengartner, U., Larson, K.: Not-so-hidden information: optimal contracts for undue influence in E2E voting systems. In: VOTE-ID (2009)

14. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: IEEE Symposium on Security and Privacy, pp. 354–368 (2008)
15. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof 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)
16. Feldman, A.J., Benaloh, J.: On subliminal channels in encrypt-on-cast voting systems. In: EVT/WOTE (2009)
17. Gardner, R.W., Garera, S., Rubin, A.D.: Coercion Resistant End-to-End Voting. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 344–361. Springer, Heidelberg (2009)
18. Heiberg, S., Lipmaa, H., van Laenen, F.: On E-vote Integrity in the Case of Malicious Voter Computers. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 373–388. Springer, Heidelberg (2010)
19. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium (2002)
20. Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation via Cipher-texts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
21. 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)
22. Jakobsson, M., Stolterman, E., Wetzel, S., Yang, L.: Love and authentication. In: CHI (2008)
23. Joaquim, R., Ribeiro, C.: Codevoting: protection against automatic vote manipulation in an uncontrolled environment. In: VOTE-ID (2007)
24. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES (2005)
25. Kane, C.: Voting and verifiability: interview with Ron Rivest. RSA Vantage Magazine 7(1) (2010)
26. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
27. Rivest, R.L., Smith, W.D.: Three voting protocols: Threeballot, VAV, and Twin. In: EVT (2007)
28. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à Voter: a voter-verifiable voting system. IEEE TIFS 4(4) (2009)
29. Ryan, P.Y.A., Teague, V.: Pretty good democracy. In: Workshop on Security Protocols (2009)
30. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptography 4 (1991)
31. Smith, W.D.: New cryptographic election protocol with best-known theoretical properties. In: Frontiers in Electronic Elections (2005)
32. Spycher, O., Koenig, R., Haenni, R., Schlapfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 182–189. Springer, Heidelberg (2011)
33. Weber, S.G., Araujo, R.S.d., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: ARES (2007)

# Malice versus AN.ON: Possible Risks of Missing Replay and Integrity Protection

Benedikt Westermann<sup>1</sup> and Dogan Kesdogan<sup>1,2</sup>

<sup>1</sup> Q2S\*, NTNU, 7491 Trondheim, Norway

<sup>2</sup> Chair for IT Security, FB5, University of Siegen, 57068 Siegen, Germany

**Abstract.** In this paper we investigate the impact of missing replay protection as well as missing integrity protection concerning a local attacker in AN.ON. AN.ON is a low latency anonymity network mostly used to anonymize web traffic. We demonstrate that both protection mechanisms are important by presenting two attacks that become feasible as soon as the mechanisms are missing. We mount both attacks on the AN.ON network which neither implements replay protection nor integrity protection yet.

## 1 Introduction

Anonymity networks like Tor [1] and AN.ON [2] aim to provide anonymity for their users, i.e., to hide the relation between a sender and a receiver of a message. Both networks are low-latency anonymity networks and can be used, for example, for anonymous web browsing. The low-latency requirement demands to find the right trade-off between protection and performance. Consequently, it is necessary to use only protection mechanisms that are strictly necessary concerning the attacker model. The attacker model used in Tor and AN.ON describes a local active adversary who controls a small fraction of the network.

Interestingly, Tor and AN.ON do not implement the same protection mechanisms, i.e., AN.ON neither has integrity protection nor replay protection. This raises the question about the possible risks that are introduced in AN.ON<sup>1</sup> by omitting the two protection mechanisms facing a local attacker.

We answer this question by presenting two different attacks. The first attack, which is referred to as *redirection attack*, exploits the lack of integrity protection and checks against a list of thousands of web sites, which web sites have been visited by a user. The result is normally a small list of web sites. The second attack, which is referred to as *replay attack*, exploits the lack of replay protection and is capable of confirming, given a small set of possible web sites that a user has visited a web site.

---

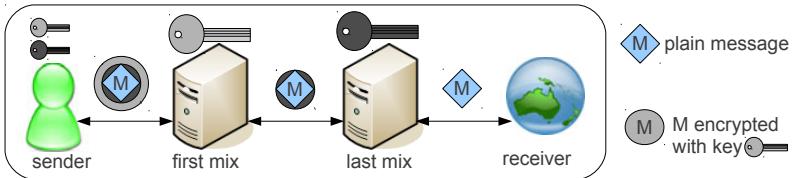
\* “Center for Quantifiable Quality of Service in Communication Systems, Center of Excellence” appointed by The Research Council of Norway, funded by the Research Council, NTNU and UNINETT. <http://www.q2s.ntnu.no>

<sup>1</sup> Tor implements both protection mechanisms and is therefore not vulnerable to the presented attacks.

The paper is structured in the following way. In Section 2 we describe AN.ON’s concept and the basic idea of its protocol. The attacker model and the assumptions for our attacks are introduced in Section 3. In Section 4, we present the redirection attack. The replay attack is described in Section 5. Section 6 presents related works, and in Section 7 we draw our conclusion.

## 2 Description of AN.ON

AN.ON uses the *cascade principle* to route the users’ traffic through the AN.ON network. The cascade principle describes the selection process of the route: a user can only choose from a set of predefined routes, so-called *cascades*. Once a user is connected to a cascade, all his messages are sent through the cascade via the same sequence of *mixes*. A cascade of length two is depicted in Figure 1. In addition to the routing over different nodes, messages are encrypted several times. With each mix a message passes, one layer of encryption is removed until eventually the last mix removes the final layer of encryption and reads the address of the receiver to which the message is eventually forwarded.



**Fig. 1.** A message traveling along a cascade

In order to use AN.ON, a user/sender needs to connect to one of AN.ON’s cascades. The connection is established by exchanging three messages. The first message contains a signed *descriptor* that is sent by the mix to the user. The descriptor includes, among others, the public encryption keys of the mixes in the cascade. The second message sent by the user to the mix, contains two encrypted symmetric keys. The two keys are used to encrypt the connection between the user and the first mix. Finally the mix sends a confirmation to the user. The confirmation is a signed hash value of the keys.

When this is done, the user can request *channels*. A channel represents the (anonymous) end-to-end connection between the user and the receiver. A user can open a channel with a *channel-open* packet. It includes two session keys  $K^s, K^r$  for each mix in the cascade. The keys are encrypted with the corresponding public encryption key of the mix. In addition, the channel open packet includes the address of the end point, i.e., the receiver, of the channel which is encrypted for the last mix.

Once a channel is opened, namely the symmetric keys are exchanged and a TCP connection between the last mix and the receiver is established, a user can exchange data with the other end point of a channel. Thereby the data is sent as

payload in a *mix packet*. A mix packet is a fixed sized data structure that is used to exchange data in the cascade. The payload of a mix packet has the size of 992 bytes and it is usually encrypted in layers. AN.ON uses the *advanced encryption standard (AES)* in *output feedback (OFB)* mode as encryption scheme. By using the OFB mode, a keystream is generated that is independent of the plaintext. In order to encrypt/decrypt data, the generated keystream is simply xored with the plaintext/ciphertext.

The encrypted payload of a regular mix packet, e.g., received by a user, can be described in the following way. Let  $P = p_1, \dots, p_{992}$  be the fully decrypted payload of a mix packet and let  $C = c_1, \dots, c_{992}$  be the fully encrypted payload. Let  $k_j^{r_i}$  be the  $j$ -th byte of the keystream of mix  $i$  for the packet which is generated with key  $K^{r_i}$ , then we can describe the encryption of a mix packet received by a user in a cascade of length  $n$  with the equation 1.

$$c_j = p_j \oplus_{i=0}^n k_j^{r_i} \quad (1 \leq j \leq 992) \quad (1)$$

As pointed out in [3], AN.ON's protocol is vulnerable to replays. Currently, a replay can only be prevented if the mix stores all the keys that were used together with the mix's public encryption key. However, this is not done. The integrity of a mix packet is not protected by integrity protection mechanisms. Therefore an attacker can arbitrarily modify the encrypted packets. A more detailed description of AN.ON can be found in [3,4,5].

### 3 Attacker Model and Assumptions

For our attack we assume a local attacker who only controls the connection between the user and the first mix. Alternatively, the attacker can control the first mix in a cascade only. The attacker can add, delete, replay, and modify messages passing on the connection, but he cannot break cryptographic primitives. For both attacks, we assume that the anonymized traffic is HTTP traffic.

In addition to the above assumptions, we presume that the attacker has following capabilities:

**Redirection attack:** the attacker controls a web server.

**Replay attack:** the attacker has a small list of web sites that are likely to be a destination of a user's request, e.g., the one that is produced by the redirection attack.

### 4 Attack 1: Redirection Attack

The objective of this attack is to find some of the servers that the user has once visited. The attack is divided into two stages. In the first stage, the attacker tries to redirect a user to a web server that is under his control. Thereby, the attacker can greatly extend his influence. This also increases the number of possible attacks. In the second stage, the attacker tries to extract information on previously visited servers. Here we use a *cascading style sheets (CSS)* based history recovery attack like the one presented in [6].

#### 4.1 Redirecting the User

For our attack we utilize HTTP, which is likely to be the tunneled protocol in AN.ON. The HTTP follows the client-server paradigm. It consists of requests issued by a client and responses sent by a server. The current version is HTTP 1.1 and it is described by the RFC 2616 [7]. A typical request as well as a typical response is given in Figure 2. Figure 2(a) depicts a request and the response is shown in Figure 2(b).

<pre> 1 GET / HTTP/1.1 2 Host: www.torproject.org 3 User-Agent: Mozilla/5.0 4 Accept: text/html 5 Accept-Language: en-us 6 Accept-Encoding: deflate 7 Accept-Charset: utf-8 8 Connection: close </pre>	<pre> 1 HTTP/1.1 200 OK 2 Date: Tue, 06 Jul 2010 3           12:46:06 GMT 4 Server: Apache 5 Accept-Ranges: bytes 6 Content-Length: 6828 7 Connection: close 8 Content-Type: text/html </pre>
(a) HTTP Request	(b) HTTP Response

**Fig. 2.** An example of the HTTP

The first line of an HTTP request is called the *HTTP request line* and it consists of three different parts, namely the *method*, the *request URI* and the *protocol version*. The following lines consist of key-value pairs. Thereby the key is separated by a colon from the value. The *status line* is the first line of an HTTP-response. It consists of three parts: the *protocol version*, the *status code*, and a *status message*. The following lines consist of response headers. A response header has a key and a value separated by a colon. The order of the headers is arbitrary.

For now we assume that the order of the HTTP response headers is fixed and the headers have the same order as in the example in Figure 2(b). Thus the *date* header is the first header in a response.

Normally, a web server replies with the status code `HTTP/1.1 200 OK` or `HTTP/1.0 200 OK`. We can use this knowledge to selectively modify the response.

As mentioned previously, AN.ON’s designers chose AES in OFB to perform per hop symmetric encryption. Due to the missing integrity protection on any of the relevant layers in AN.ON, even an external attacker can arbitrary modify the packets passing an observed link.

The fact that the packets are encrypted various times does not complicate the attack, since each layer of encryption basically xors another keystream to the message. The different keystreams can be aggregated to a single keystream<sup>2</sup>. Therefore we can consider for our attack AN.ON’s various layers of encryption

---

<sup>2</sup> Due to the fact that AES in OFB is used to encrypt the connection between the mix and the user too, we can extend the argumentation in the same way.

simply as one layer of encryption. The ciphertext is created by XORing the plaintext bitwise with the keystream and therefore an attacker can selectively modify single bits/bytes in an encrypted packet without modifying the remaining bytes of the eventually decrypted plaintext. For example, an attacker knows the  $i$ th byte of the plaintext, i.e.,  $p_i$ , and he wants to change the ciphertext  $c_i$  such that  $c'_i$  decrypts to  $p'_i$ . He can achieve this by replacing  $c_i = k_i \oplus p_i$  with  $c'_i = c_i \oplus (p_i \oplus p'_i) = (k_i \oplus p_i) \oplus (p_i \oplus p'_i) = k_i \oplus p'_i$ . The attacker can use this to redirect a user to a different server.

In this stage of the attack, the objective of an attacker is to redirect a user to his own web server. He can do so by exploiting the features of HTTP. In HTTP a status code 302 indicates a temporarily moved resource. If the browser receives such a code, it automatically redirects the user to a new resource that is given in a `location` header. Therefore, an attacker has to do two things. Firstly, he has to change the status code from 200 to 302. Secondly, he has to inject a `location` header in the HTTP response that points to his server.

For now, we assume that the first two lines of the response are as in the example in Figure 2(b). In order to change the status code from 200 to 302, an attacker needs to change two bytes in the ciphertext. He needs to xor the ciphertext byte<sup>3</sup>  $c_{10}$  with  $2 \oplus 3$ , i.e.,  $c'_{10} = c_{10} \oplus 2 \oplus 3$ . In addition, he needs to change the ciphertext byte  $c_{12}$  by XORing  $0 \oplus 2$  to this byte. The injection of the location header, which points to the attacker's server, can be done by replacing the `date` header. The `date` header is according to our assumption the next header in the response. This header has a fixed length and most parts of it are known, e.g., the year, the day, the month and most likely also the hour. Thus the attacker can exchange the `date` header by a `location` header. These two small changes are sufficient to redirect a user to the server of an attacker.

Up until now we assumed that the response is as the one given in Figure 2(b). In order to evaluate if this assumption is reasonable, we collected the HTTP responses of various websites. To this end, we queried the 2000 most popular websites according to Alexa<sup>4</sup>. The HTTP response headers with its sequence number as well as the status line were stored for each website. The hostname, which is given in the Alexa database, was transformed to a URL. For example, the hostname `example.org` was transformed to `http://www.example.org`.

In 767 cases, our request resulted in a response<sup>5</sup> like the one shown in Figure 3 which corresponds to a percentage of 38.35 %. Here  $x$  marks the positions of the variant parts. This type of response was the most frequently seen.

```
HTTP/1.x 200 OK\r\n
Date: Wed, 07 Jul 2010 xx:xx:xx GMT\r\n
```

**Fig. 3.** The beginning of 38 % of the HTTP responses

<sup>3</sup> Here we ignore the header of the mix packet, but this is just a constant.

<sup>4</sup> <http://www.alexa.com/topsites>

<sup>5</sup> The data was collected on the 07/07/2010.

This brief analysis indicates that an attacker can modify up to 31 bytes starting from the status code 200. According to the HTTP specification, the status message is an arbitrary text including an empty text. Consequently, an attacker can use up to 25 bytes for the `location` header. Hereby 17 bytes are used for “`Location: http://`” and one byte is used for a trailing “`/`” to separate the domain from the path. Therefore 7 bytes remain for the domain name including the top level domain. For example, the domain `abcd.xy` is a suitable domain for the attack. If the header is guessed correctly, the modification should lead to a response presented in Figure 4. The remaining part of the `date` header can be handled by the web server as long as a “`/`” separates the domain from the path.

```
HTTP/1.1 \r\n
Location: http://abcd.xy/xx:xx:xx GMT\r\n
```

**Fig. 4.** The HTTP response header after the modification

## 4.2 History Recovery

In this part we describe how an attacker can recover the browser history of a user. To do this, the attacker can exploit a well-known feature in CSS, namely the ability to display visited links differently than non-visited links. An attack based on this feature is described in [6]. In the paper the authors show the feasibility of the attack and they estimate that at least 76% of the Internet users are vulnerable to this kind of attack.

One possibility to mount the attack is to embed a list of links to potential websites in the served HTML document. Additionally, the attacker instructs the browser to load a unique picture as background picture for each link with help of cascading style sheets. An example of such a document is given in Figure 5.

Normally, a browser parses the document and applies the style that is most specific for each element. For the example in Figure 5, the browser would apply the style with ID `#site1`, if the site `site1.example.org` has been visited earlier.

```
1 <style type="text/css">
2   #site1 a:visited {
3     background-image: url(' / pic.php?id=1');
4   }
5   #site2 a:visited {
6     background-image: url(' / pic.php?id=2');
7   }
8 </style>
9 <a id='site1' href='http://site1.example.org/'></a>
10 <a id='site2' href='http://site2.example.org/'></a>
```

**Fig. 5.** A example of an HTML file capable of leaking the history of a user

In case a user has not visited a site earlier, it applies the default style. Subsequently, the browser would issue a request to retrieve the background picture for each of the links that have been visited at least once. The URL for the referenced picture is unique due to the GET-parameter of the URL. Thus the attacker can log the sites that a user has visited earlier, since only the pictures of the visited sites are requested. The presented attack is feasible with the version 3.6.10 of Firefox. However, it should be noted that the browser vendors proposed and have partially implemented countermeasures for this kind of attack [8].

### 4.3 From Theory to Practice

In order to mount the attack on the real AN.ON network, we intercepted and modified the first packet containing an HTTP response that was received by a patched *Java AN.ON Proxy* (JAP). JAP is AN.ON's client software. The packet was selected by counting the total number of received packets. Thus it was not necessary to read the message. We modified the data prior to all other operations. Therefore, we simulated an external attacker. Even though we modified the JAP to intercept and modify the packets, it does not limit the generalizability. It is also possible to modify the packets directly on the transport layer, but it was found more convenient to patch the client to modify and intercept the packets than modifying the TCP packets on the fly.

We used the patched JAP to redirect the request by modifying the encrypted packet as described in Section 4.1 and tested the attack with various sites, e.g., <http://www.youtube.com>. We opened the JAP and requested the YouTube site. As expected we were redirected to our own server. This shows that the attack is feasible and capable of redirecting a user to a different web server that is chosen by an attacker. With the user redirected to our own server, we mounted the previously described CSS history attack. We recovered parts of the history.

### 4.4 Evaluation

We demonstrate with our proof-of-concept that the attack can be mounted in the AN.ON network. In this part, we estimate the success rate of the attack.

In [6], the authors concluded that the attack succeeded at least for 76 % of the clients. There are various reasons why the attack can fail. First of all, the attack fails if a user has not visited the websites that are tested by an attacker. Another reason is that some users disable the history of the browser, or have deleted it recently.

The authors in [6] have not investigated users of anonymity networks and therefore their rate cannot be used directly for the computation of the success rate. There are at least two reasons for this. Firstly, AN.ON's users are recommended to use a modified Firefox version, the so-called JonDoFox. The modified version disables, among others, the history of the browser such that the history attack fails. Therefore the number of users of the JonDoFox heavily influences the actual success rate of the attack. Secondly, users of anonymity networks might be more privacy aware and are therefore more likely to deactivate the

history, or at least delete the history more frequently. While the effect of the latter is hard to estimate, we can take the first factor into consideration. According to the status page of the AN.ON network<sup>6</sup>, only 14.1% of AN.ON's users are using the JonDoFox and 4.94% using “other” browsers that are not further specified. The remaining 80.96% represent well-known browsers like Firefox. For our estimation, we assume that the history attack is equally likely to work with every browser except the category “others” and “JonDoFox” with a probability of 76%. This is the success rate given in [6]. For the other two categories, we assume that the history attack always fails. Thus the probability of success is 0. Let  $B$  be the event that the attack succeeds and let  $P(A_1)$  to  $P(A_3)$  be the following probabilities:

$$\begin{aligned} P(A_1) &= P(\text{“The user uses a well-known browser”}) = 0.8096 \\ P(A_2) &= P(\text{“The user uses JonDoFox”}) = 0.141 \\ P(A_3) &= P(\text{“The user uses another browser”}) = 0.0494 \end{aligned}$$

According to our assumption, we can express the following probabilities:

$$\begin{aligned} P(B|A_1) &= 0.76 \\ P(B|A_2) &= P(B|A_3) = 0 \end{aligned}$$

By applying the law of total probability, we can compute  $P(B)$ :

$$P(B) = \sum_{i=1}^3 P(B|A_i) \cdot P(A_i) = 0.62$$

In Section 4.1, our analysis showed that in 38.35 % of the cases the redirect of user would be possible due to a correctly assumed header. Let  $C$  be the event of a successful redirect, then  $P(C) = 0.38$  describes the corresponding probability.

In order to successfully mount the two stages of the attack, the redirect has to work as well as the history attack. Thus  $P(B \cap C)$  is the success rate of the whole attack. Since it seems unlikely that the success of a redirect is (significantly) influenced by the success of a history attack and vice versa, we assume that both events are stochastically independent, namely  $P(B \cap C) = P(B) \cdot P(C)$ . Therefore the success rate of the proposed attack for a randomly chosen user in the AN.ON network for the given scenario, i.e., the attacker modifies the first HTTP response of an HTML document, is roughly 0.24.

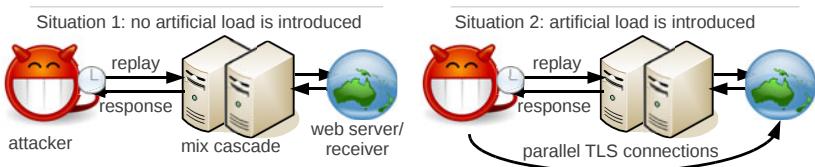
## 5 Attack 2: Replay Attack

In this section, the objective of the attacker is to confirm the relation between a user and his communication partner. We focus on confirming an HTTP connection to a web server. In order to mount the attack, an attacker records messages

---

<sup>6</sup> <http://infoservice.inf.tu-dresden.de:6543/status> (visited 30.09.2010).

that are sent by the user to the cascade, i.e., the first mix in the cascade. Afterwards, he replays continuously the recorded messages into the cascade. The replayed messages are processed normally by the mixes in the cascade. Eventually they are sent to the web server that replies to the requests. The responses of the web server travel back along the cascade. The attacker measures the time that it takes to get the responses to the replayed messages. At the same time as the attacker replays the packets, he introduces an alternating artificial load pattern at the web server, e.g., by performing various search queries, or establishing various TLS connections to the web server. Our hypothesis is that the introduced pattern should influence the response times to replayed requests such that the attacker by observing the response times can detect the pattern. Therewith he can confirm a relationship between the user and the web server. The idea of the attack is sketched in Figure 6.



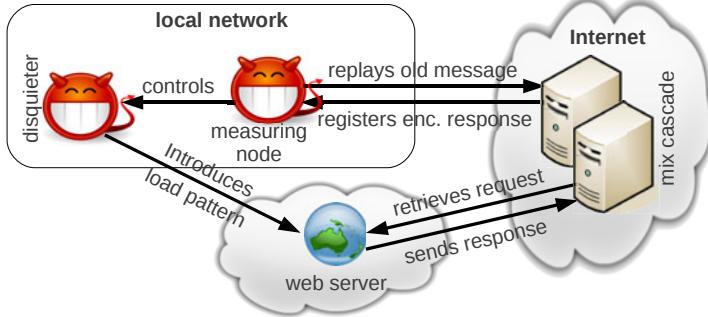
**Fig. 6.** An attacker has to distinguish the two situations at the delay of the responses

In our setup, the attacker introduces the artificial load by establishing various TLS connections in parallel to an anticipated web server. We assume that the web server has at least one port available that accepts a TLS connection, e.g., HTTPs, or SMTPs. However, there are many different possibilities to introduce reasonable load on a web server that do not even require the establishment of a TLS connection. The only requirement is that the request is costly for web server. Examples of such requests include, but are not limited to: search requests, dynamic page generation, or cryptographic operations.

### 5.1 Methodology

The proof-of-concept involves three different parties: a *measuring node*, a *disquieter*, and a *web server*. The task of the measuring node is to replay the previously recorded messages into the cascade and to measure the response times to the replayed requests. Additionally, the measuring node controls the disquieter. The task of the disquieter is to introduce the load pattern at a suspected web server by establishing in parallel various TLS connections. The web server is the actual receiver of a replayed request and is attacked by the disquieter.

Figure 7 depicts our experimental setup. As shown in the figure, the disquieter and the measuring node are located in the same network. Both share the same network resources. However, the network is unlikely to be a bottleneck in our experiments and it is unlikely to influence the results significantly. The

**Fig. 7.** Sketch of the Experimental Setup

equipment of the used computers is given in Table 1. We configured two different web servers in a typical *LAMP* configuration. *LAMP* stands for Linux, Apache, MySQL and PHP. On both web servers, we installed a popular blog application, i.e., Wordpress 3.0.1. The first server was installed in Amazon's Elastic Compute Cloud (EC2). The other web server is located in a data center in Germany. All machines were solely used for our experiments. Despite the lack of users, both web servers should represent a typical web server setup. Since our attack can be interpreted by others as denial of service attack (attacking the availability of a third party's system is against the law, e.g., in Germany), we only performed the attack on our systems.

**Table 1.** Equipment and location of the used computers

	<b>Measuring Node</b>	<b>Disquieter</b>	<b>Web Server 1</b>	<b>Web Server 2</b>
<b>CPU</b>	Intel P8400 2.26 GHz	Intel Atom N280 1.66 GHz	Intel i7 920 2.67 GHz	Intel Xeon 5410 2.33 GHz 2 logical CPUs
<b>RAM</b>	4 GB	2 GB	8 GB	1.7 GB
<b>Location</b>	Norway NTNU's university Network		Germany in a data center	Ireland Amazon's EU EC2

We distinguish between two types of measurements. A measurement of type 0 represents the situation in which the disquieter idles. The other situation is referred to as measurement of type 1. Here the disquieter builds up TLS connections to the anticipated web server. We alternated the two types every 30 samples. After 30 samples the measuring node instructed the disquieter to toggle the load. In total, we collected 300 samples for each type of measurement. The samples were collected in serial with a delay of 500 ms between each measurement of a sample.

We mounted the attack on several cascades with varying properties, namely *Speedpartner-ULD*, *Dresden* and *Koelsch-Rousseau-SecureInternet1*, for each of

the two web servers. The *Dresden* cascade is the most popular and most frequently used cascade. It does not limit the number of users. The *Speedpartner-ULD* cascade allows at maximum 1200 users to be connected. Contrary to the first two cascades, the cascade *Koelsch-Rousseau-SecureInternet1* is a premium cascade. Here the user has to pay for the relayed traffic. Due to the involved payment protocol used on a premium cascade, a replay is not possible without controlling the first mix. Hence we modified the experiment slightly. Instead of replaying the messages into the cascade, we issued regular requests to our server with a modified version of the JAP.

The packets for the replay were generated with the JAP. To this end, we started the JAP, connected to the cascade, issued a single HTTP request via the JAP to the web server and closed the JAP. All packets sent during the period were recorded and later on replayed into the cascade.

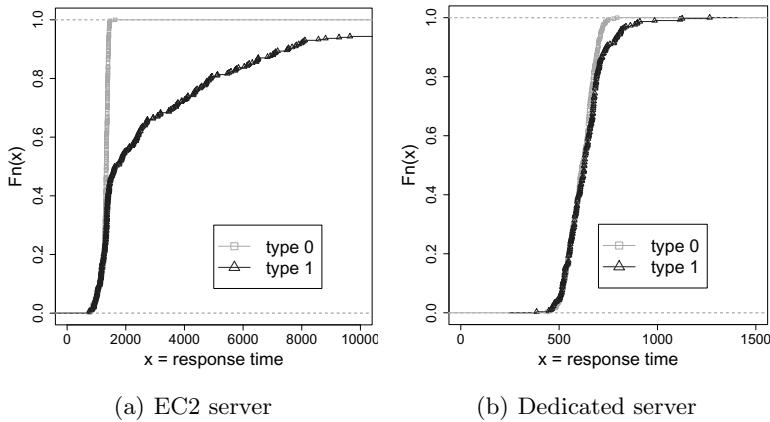
To replay packets, we established a TCP connection to the first mix, waited for the descriptor sent by a mix, and afterwards we replayed the first packet containing the encrypted keys. As soon as the mix had sent the third message, which is the confirmation, the time measurement starts and the remaining previously sent packets were replayed into the cascade. The time measurement stopped as soon as the same number of packets has been received as previously recorded.

In order to test if the two types of measurements are distinguishable, we used the *Kolmogorov-Smirnov* (KS) test due to its few requirements concerning the sample distribution: the only requirement is that the samples are *independently identically distributed* (iid). The KS test can be used to decide if two sets of samples stem from the same underlying probability distribution. The test is done by calculating the maximum distance between the empirical cumulative distribution functions. For our statistical analysis, we used the program R.

## 5.2 Measurement Results

We mounted the attack first on the not so heavily used *Speedpartner* cascade. The cascade limits the number of users on the cascade to 1200. Additionally, the bandwidth available to each user is also limited. Therefore less noise is expected. This can ease the recognition of the introduced pattern. The cascade consists of two mixes. Both are located in Germany. The first mix is located in Dusseldorf and the other one in Karlsruhe.

Figure 8 shows the *cumulative distribution functions* (CDFs) of the experiment at the *Speedpartner* cascade. While the difference of the CDFs in Figure 8(a) is clearly visible with respect to the EC2 instance, it is not as clear for the dedicated server. The CDFs for the dedicated server are shown in Figure 8(b). Here the introduced pattern has a slightly increased probability for having higher response times. However, this result is expected as the dedicated server is by far more powerful than the EC2 instance. The numeric values of the test are given in Table 2. In both cases, the KS test results in a  $p$ -value below 0.05. The value 0.05 is our rejection level for the null hypothesis, namely  $F(type0) = F(type1)$ . Therefore, we reject in both cases the null hypothesis. For an attacker this would mean that there is a difference between the measurement of type 0 and type 1.



**Fig. 8.** The CDFs of the measurements at the Speedpartner cascade

Since the replays are sent to the same destination and the only variable changed by the attacker is the load on the anticipated server, it is likely that the replayed packets are sent to the tested web server.

The results for the other cascades are summarized in Table 2. Most notable is the case in which the Dresden cascade was tested together with the dedicated server as anticipated server. Although the packets were relayed to the dedicated server, it was not possible to distinguish the samples with the used methodology. The difference between the measurement of type 0 and type 1 was probably overshadowed by the noise introduced by roughly 2500 users that were using the cascade during our measurements. In all the other cases, we were able to distinguish the two types of measurements.

**Table 2.** Summary of the p-values for all performed measurements

		Tested server ( <i>p</i> -values)				
cascade	attacked server	EC2	dedicated	google	wikipedia	facebook
Koelsch	ec2	$< 3 \cdot 10^{-16}$	-	0.52	0.72	0.72
	dedicated	-	$< 3 \cdot 10^{-16}$	0.72	0.40	0.08
Dresden	ec2	$< 3 \cdot 10^{-16}$	-	0.08	0.21	0.72
	dedicated	-	0.21	0.79	0.58	0.21
Speedpart.	ec2	$< 3 \cdot 10^{-16}$	-	0.45	0.85	0.79
	dedicated	-	0.04	0.72	0.25	0.90

Due to possible legal consequences, we did not attack productive systems. However, to test for false positive, we used some productive systems, namely *google.com*, *wikipedia.org* and *facebook.com*. We used the servers as destinations for the replayed packets while attacking one of our own servers, i.e., the EC2 instance and the dedicated server respectively. The results are given in Table 2.

The absence of false positives indicates that the performance of a normally used web server is constant enough to mount the attack.

In total, we observed only a single false negative and not a single false positive. The false negative is probably caused by the load of the Dresden cascade. We assume that the load was too high to detect the introduced delay. Worth to mention is that the attack is limited. There are several cases in which the attack is likely to fail, e.g., when load balancing techniques are used. However, the attack demonstrates the risk introduced by omitting replay protection. Moreover, it is likely that the possibility of replays can help to improve other attacks, e.g. web site fingerprinting attacks [9]. This also stresses the need of replay protection even against a local attacker.

## 6 Related Works

There have been various timing attacks proposed during the last years. A recent example is the attack of Evans et al. [10] which is an improved attack of [11]. Here the authors identify the user selected path through the Tor network.

In [12], Hopper et al. present a timing attack that allows colluding web servers to decide whether two connections from the Tor network originates from the same user. In a second attack, the authors exploit the timing information to narrow down the location of a user of the Tor network.

Another timing attack was presented in [13]. Here the authors introduce a traffic burst on a network router and check if this influences the data flow of a circuit. By identifying the routers influencing the circuit, it is possible to track the full path of a circuit. While their approach works well in the controlled setting, it was harder to detect the traffic fluctuations in the real Tor network.

An implementation specific attack against Tor was proposed in [14]. Here the authors delayed cells in a way such that it is possible to influence the number of cells in an IP packet. They used this to embed a hidden signal at the exit node of a circuit and showed that an attacker at the entry node can recover this hidden signal. Thus an attacker controlling the first and exit node can deanonymize a user with help of this technique.

In [15], the authors describe how a local attacker can deanonymize a user with a replay attack/tagging attack in Tor. Here an attacker needs to observe the exit as well as the entry node of the Tor network. The attack exploits that replays are only detected at the end points of a circuit. An attacker who controls the entry and the exit node can replay a packet at the first node. Eventually the replayed packet will be discovered at the exit node. Thereby an attacker can map the entry node and the exit node to the same circuit.

Another method to confirm that an entry node and an exit node participate in the same circuit was proposed in [16]. In the paper, the authors link the entry node and the exit node of a circuit with each other by correlating the *CREATE CELLS* with help of their dispatch and arrival times at the corresponding nodes.

In [3], the authors inspected the cryptographic protocols used in AN.ON. Based on the flaws found in the cryptographic protocols, they proposed some attacks exploiting them.

An overview of general attacks against low-latency anonymity networks is given in [17,18].

## 7 Conclusion

In this paper we exploit the lack of replay protection and integrity protection in AN.ON to mount two different attacks. Our attacks demonstrate that the lack of these protection mechanisms introduces a high risk. Thereby we only assumed a local attacker who controls the connection between the user and the first mix. Although the first attack is not capable of deanonymizing a user, e.g., the attack does not identify the web site that an attacked user intends to visit with AN.ON, it introduces a risk. It is easy-to-mount and has a good chance of success in disclosing some web sites a user has once visited. The latter can be important for follow-up attacks like the presented replay attack. This replay attack requires more resources than the redirect attack and is also more difficult to mount. However, if the attacker manage to delay the responses of the correct anticipated web server, the attack is capable of deanonymizing the user. In combination, the attacks introduce a risk that is unacceptable for a low-latency anonymity network.

Currently, the AN.ON team is working to integrate both protection mechanisms, e.g., an integrity protection mechanism is currently in the testing phase.

**Acknowledgements.** We like to thank Pern Hui Chia and Svein Knapskog for their valuable feedback. We also thank the JonDonym and AN.ON team for their answers to our questions and the anonymous referees for their helpful suggestions. Finally, we want to thank our shepherd Roger Dingledine.

## References

1. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: USENIX Security Symposium, USENIX, pp. 303–320 (2004)
2. 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)
3. Westermann, B., Wendolsky, R., Pimenidis, L., Kesdogan, D.: Cryptographic Protocol Analysis of AN.ON. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 114–128. Springer, Heidelberg (2010)
4. Köpsell, S.: Entwicklung und Betrieb eines Anonymisierungsdienstes für das WWW. PhD thesis, TU Dresden University (2010)
5. Köpsell, S.: AnonDienst - Design und Implementierung. Technical report, TU Dresden University (2004)
6. Janc, A., Olejnik, L.: Web Browser History Detection as a Real-World Privacy Threat. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 215–231. Springer, Heidelberg (2010)
7. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol: HTTP/1.1. Internet Engineering Task Force: RFC 2616 (June 1999)

8. Stamm, S.: Mozilla security blog: Plugging the css history leak (March 2010), <http://blog.mozilla.com/security/2010/03/31/plugging-the-css-history-leak/> (visited September 30, 2010)
9. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, pp. 31–42. ACM, New York (2009)
10. Evans, N., Dingledine, R., Grothoff, C.: A practical congestion attack on Tor using long paths. In: USENIX Security Symposium, USENIX, pp. 33–50 (2009)
11. Murdoch, S.J., Danezis, G.: Low-cost traffic analysis of Tor. In: IEEE Symposium on Security and Privacy, pp. 183–195. IEEE Computer Society (2005)
12. Hopper, N., Vasserman, E.Y., Chan-TIN, E.: How much anonymity does network latency leak? ACM Transactions on Information and System Security 13(2), 1–28 (2010)
13. Chakravarty, S., Stavrou, A., Keromytis, A.D.: Traffic Analysis Against Low-Latency Anonymity Networks Using Available Bandwidth Estimation. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 249–267. Springer, Heidelberg (2010)
14. Ling, Z., Luo, J., Yu, W., Fu, X., Xuan, D., Jia, W.: A new cell counter based attack against tor. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM Conference on Computer and Communications Security, pp. 578–589. ACM (2009)
15. Pries, R., Yu, W., Fu, X., Zhao, W.: A new replay attack against anonymous communication networks. In: IEEE International Conference on Communications, pp. 1578–1582. IEEE (2008)
16. Bauer, K.S., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.C.: Low-resource routing attacks against tor. In: Ning, P., Yu, T. (eds.) WPES, pp. 11–20. ACM (2007)
17. Back, A., Möller, U., Stiglic, A.: Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137, pp. 245–257. Springer, Heidelberg (2001)
18. Raymond, J.-F.: Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 10–29. Springer, Heidelberg (2001)

# Absolute Pwnage: A Short Paper about the Security Risks of Remote Administration Tools

Jay Novak, Jonathan Stribley, Kenneth Meagher, and J. Alex Halderman

The University of Michigan  
`{novakjs,strib,meagherk,jhalderm}@umich.edu`

**Abstract.** Many IT departments use remote administration products to configure, monitor, and maintain the systems they manage. These tools can be beneficial in the right hands, but they can also be devastating if attackers exploit them to seize control of machines. As a case study, we analyze the security of a remote administration product called Absolute Manage. We find that the system’s communication protocol suffers from serious design flaws and fails to provide adequate integrity, confidentiality, or authentication. Attackers can exploit these vulnerabilities to issue unauthorized commands on client systems and execute arbitrary code with administrator privileges. These blatant vulnerabilities suggest that remote administration tools require increased scrutiny from the security community. We recommend that developers adopt defensive designs that limit the damage attackers can cause if they gain control.

## 1 Introduction

Remote administration products allow system administrators to manage collections of machines from a central location. These tools carry inherent security risks. If an attacker can exploit them to issue unauthorized commands, he may be able to take control of client machines. The question is, do the designers of remote administration software take adequate steps to protect the security of their users?

As a case study, we analyzed the security of Absolute Manage [1], a remote administration tool by Absolute Software. Absolute Manage has been deployed by companies, universities, and school districts throughout the U.S. [1]. It has been in the news since February 2010, when a school district in Pennsylvania was alleged to be using it to spy on students at home via their laptop webcams [10]. We selected it for our study after this controversy brought it to our attention. We had no reason to believe its security would be particularly weak or strong.

We began with black-box testing, through which we determined that an attacker with control of the network could subvert local clients and run arbitrary code. Following this initial result, we used the IDA Pro disassembler to understand the software’s communication protocol and security mechanisms, which we found to contain significant design flaws. These vulnerabilities allowed us to develop attacks that require less control over the network while still providing complete adversarial control over the client.

The problems we found in Absolute Manage suggest lessons for remote administration products more broadly. We observe that the design flaws we uncovered were elementary mistakes that would not have gone unnoticed in even the most basic security review. Given the magnitude of the risks these products pose when they are vulnerable, we recommend that developers adopt defensive design and programming practices. The goal should be to ensure that, even if an attacker is able to issue unauthorized commands through the software, the damage he can cause will be limited.

*Outline* Section 2 introduces Absolute Manage and the software’s communications protocol. Section 3 describes serious vulnerabilities we found in its encryption and authentication. Section 4 explains ways attackers could exploit these flaws to take control of clients. Section 5 discusses ways to mitigate the problems and draws broader security lessons. We survey related work in Section 6, and we conclude in Section 7.

We present additional details in the extended version of this paper, available at <http://www.cse.umich.edu/~jhaldem/>.

## 2 Background

Absolute Manage is a product of Absolute Software, which purchased it from Pole Position Software in December 2009 for \$12.1 million and 500,000 shares of common stock [2]. Using server-side tools, administrators can instruct clients to install programs, apply software updates, run scripts, take screenshots, or execute code, among other functions. Clients also report status information to the server at regular intervals (by default, every 15 minutes). The client software supports Windows and Mac OS X.

One place where Absolute Manage is used is Lower Merion School District in eastern Pennsylvania, which installed it on laptops issued to around 1800 high school students. In February 2010, one of those students, Blake J. Robbins, sued the district in federal court, alleging that school officials violated students’ privacy rights by secretly using the laptop cameras to photograph them in their homes [10], using the Absolute Manage TheftTrack feature.

The Absolute Manage software suite includes Absolute Manage Agent, which runs on client machines, and Absolute Manage Server. Both sides accept TCP connections; by default, clients listen on port 3970 and servers listen on port 3971. The server can issue commands as a response to the heartbeat or by directly contacting the client. If a command cannot be delivered, it is queued and resent in response to the next heartbeat.

The body of each message is an XML-formatted property list. The properties include:

- *AgentSerial*. A unique identifier for the client that is randomly generated on installation.
- *AdminUUID*. A unique identifier for the server that is randomly generated on installation.

- *CommandUUID*. A unique identifier for the command that is randomly generated when it is issued.
- *CommandID*. A number that specifies the command to be executed and determines the format of the *CommandParameters* structure.
- *SeedValue*. A 192-bit binary value used to authenticate the server.

Prior to transmission, all messages are compressed with `zlib` and encrypted using the Blowfish cipher [11] operating in ECB mode.

## 3 Vulnerabilities

Due to a number of design flaws, the Absolute Manage protocol fails to provide adequate integrity, confidentiality, or authentication.

### 3.1 Defective Encryption

The Absolute Manage developers opted for a simple cryptographic design: all clients and servers use the same hard-coded secret keys every time, for every message. We were able to discover the keys by examining the client program with IDA Pro. There are at least four keys used for different purposes: two are based on a German phrase and differ only in punctuation, one is a minor corruption of a common colloquial expression, and one appears to be a snide remark about a design choice. All can be exposed by running the `strings` command on the client binary.

Using hard-coded secret keys is highly risky, since an attacker who manages to extract them from one copy of the software can then attack all the other copies. In the case of Absolute Manage, an attacker who learned the keys could decrypt intercepted protocol messages, including messages containing sensitive private data or proprietary software; he could act as a man-in-the-middle and arbitrarily modify the contents of messages in transit; or he could generate new encrypted messages from scratch and pass them to servers and clients. Another important flaw is that the protocol uses ECB mode, so blocks are encrypted independently and deterministically. This lets attackers compromise the protocol even without knowing the keys.

### 3.2 Defective Authentication

Since remote administration software gives parties the ability to control the machine, it is essential to ensure that only *authorized* third parties can do so. Unfortunately, Absolute Manage uses an extremely weak authentication mechanism for its client-server communication.

When the client receives a command message, it tries to confirm that it originated from an authorized server. Each server has a unique *SeedValue* that it includes in all its command messages, and clients discard commands that do not have the expected value. In contrast, the *AdminUUID*, *AgentSerial*, and *CommandUUID* properties can have arbitrary values.

The `SeedValue` property is a random-looking 192-bit string, so one might expect it to be difficult to guess. In fact, it carries very little entropy. If we decrypt our test server’s `SeedValue` using Blowfish in ECB mode and a different hard-coded key, we get the following bytes:

```
00 00 00 0E 00 31 00 34 00 30 00 31
00 34 00 37 00 35 00 00 00 00 00 00
```

The first four bytes are a length specifier, and the trailing zeroes are padding. After removing these, we are left with the UTF-16 encoding of a 7 character string: 1401475. This is the server’s “serial number,” which was provided by Absolute Software along with the product activation key when we purchased our license. If all server serial numbers are 7 digits like ours, and they are randomly assigned, then the `SeedValue` property contains about 23 bits of entropy. We suspect the assignment is nonrandom, so the actual entropy may be much less.

Furthermore, in certain situations, clients do not apply any authentication at all. One example is upon client initialization. The client discovers its server’s `SeedValue` by *asking the server*. It sends a heartbeat message with the `NeedSeedValue` property set to true. Until the client receives a `NeedSeedValue` response from the server, it defaults to accepting commands with any `SeedValue`. Clients do not store the `SeedValue` to disk, so they need to ask the server again every time the software starts.

An additional vulnerability is that the servers do not authenticate messages from clients. Any correctly formatted message sent to the server is processed as a valid message, no matter where it originated or who sent it. Any client can send the server a heartbeat message with `NeedSeedValue` set to true, and the server will respond with its `SeedValue`.

## 4 Attacks

There are a variety of ways that attackers could exploit the vulnerabilities we identified to issue unauthorized commands to Absolute Manage clients, including commands that silently install and run arbitrary code with administrative permissions. We demonstrated several of these attacks in a testbed network and measured their performance; for details, see the extended version of this paper.

### 4.1 Sniffing Attacks

An attacker who can observe Absolute Manage traffic can use a number of techniques to identify clients to target and to learn the server’s `SeedValue`, with which he can issue arbitrary commands to all the server’s clients. The most basic attack is to listen for connections initiated by the server, which can be recognized by the default client port number. The destination IP address identifies a potential victim, and, since server-originated commands always contain the `SeedValue`, the attacker can decrypt them to learn it.

## 4.2 Guessing Attacks

Adversaries anywhere on the Internet can use other attacks to target any Absolute Manage client with a publicly accessible IP address. For instance, given the address of a target, the attacker could use a brute-force attack to quickly guess the expected SeedValue.

We experimentally measured the performance of a SeedValue guessing attack in a laboratory environment. Our attacker software, a multithreaded Python program, attempts to impersonate the server and repeatedly sends benign commands, each time embedding a different guess for SeedValue. The Absolute Manage client responds with an acknowledgment message containing the CommandResultError property, which has a nonzero value if the SeedValue guess is wrong. Our program attempts different random guesses until the client indicates a successful guess.

Using the multithreaded approach, we were able to test an average of 330 guesses per second on a fast network connection. The bottleneck appeared to be the client's software and TCP stack. We found there was very little performance increase when the number of threads exceeded 8 or when attacking from multiple machines. Under these conditions, the expected time to successfully guess the seed would be about four hours, if we assume that the server's serial number contains seven or fewer digits.

## 4.3 Global Attacks

An attacker who wants to cast a wide net can use a different attack to efficiently target all clients at once. The first stage of the attack is to build a dictionary of server SeedValues. Servers running Absolute Manage have a unique signature that can be identified using a port scanner such as `nmap` [5]. With such a tool, the attacker can perform an Internet-wide scan for publicly addressable servers. Whenever he finds a server, the attacker sends it a heartbeat message asking it to return its SeedValue, and he adds the response to his dictionary. In the second stage of the attack, the attacker performs further network scanning to locate clients and uses the SeedValue dictionary to mount rapid guessing attacks against them. Using this attack, an adversary can take control of a large fraction of the publicly addressable machines running the Absolute Manage client.

## 5 Defenses and Lessons

Absolute manage released a new version of the product in mid 2010 that modifies the protocol and adds an SSL-based transport. While SSL is likely to be a significant improvement over the system's current encryption and authentication methods, there are many things that could still go wrong, particularly with certificate management.

Absolute Manage users running older versions need to take immediate steps to protect themselves from the attacks we have described. For the rest of us, the problems in Absolute Manage carry lessons about security risks in remote administration software more generally and about patterns of security failure.

### 5.1 Risks of Remote Administration Tools

Remote administration products like Absolute Manage carry large risks because they intentionally create mechanisms that allow remote parties to take control of a machine. There will always be a risk of *abuse* by authorized parties, as alleged in the students' lawsuit against Lower Merion School District, but correctly designed technology should at least prevent unauthorized third-party attacks by making sure only authorized parties can issue commands. This requires getting authentication right—exactly what Absolute Manage failed to do.

Because of these inherent dangers, remote administration software warrants careful security scrutiny during design, implementation, and testing. Furthermore, remote administration software should be designed defensively in order to minimize the harm to users if the authentication does fail. For example, clients could default to allowing only a minimal set of low-risk operations, and enabling additional operations could require physical access by an administrator. Or, if the client was intended to allow software installation but not remote desktop control, it could be designed to only allow the installation of binaries signed with a secondary key controlled by the administrators.

When remote actions do take place, clients should give users prominent notification and even a chance to cancel or postpone the activity. Howell and Schechter [6] recently proposed a UI paradigm for giving users control over sensor data that might be applicable in this context. Keeping users informed about what is happening to their computers would make attacks—and abuse—easier to detect and avoid.

### 5.2 Hard-Coded Keys as a Vulnerability Pattern

The problems with Absolute Manage are part of pattern of security failures involving hard-coded cryptographic keys. Using hard-coded secrets often negates the benefits of cryptography. It is widely recognized as a recurring vulnerability pattern, and was named one of the CWE/SANS “Top 25 Most Dangerous Programming Errors” for 2010 [4].

Why do developers keep making this mistake? Given that the broader security community treats the use of hard-coded keys with deserved contempt, we might conclude that it is a symptom of broader problems, indicative of companies that are devoid of security culture, process, or training. Yet perhaps this instead reflects a rational choice on the part of developers in light of the “weakest link” nature of security. Suppose you are a small developer with the resources to invest, at most, 50% of the cost of building strong security. Since investing 50% will likely leave the system just as vulnerable as if you had invested 1%, why waste the extra money? Retrofitting security is much harder than building it into a product from the start. If our hypothetical developer later sells the system to a larger company that can afford security in its home-grown products, the purchaser may nevertheless be unable to afford the investment needed to make the system secure.

Under this theory, hard-coded keys are a manifestation of a bimodal phenomenon that causes some rational developers to invest heavily in security and

others to essentially give up on it. This suggests that the only way to prevent problems like those in Absolute Manage is to change developers’ incentives, either by making security much cheaper for them to build or by increasing the odds that insecurity will harm their profits.

## 6 Related Work

*On Absolute Manage.* To our knowledge, the first published analysis of Absolute Manage appeared in a blog post by security consultants Aaron Rhodes and stryde.hax [13] in February 2010. Other prior work came after we had finished our investigation but before we disclosed our results to the vendor. In late May, the Threat Level blog reported [15] that researchers from the Leviathan Security Group had discovered the hard-coded keys and demonstrated how they could be used to attack clients on local networks. The researchers have not published the details of their findings, but we infer that their attacks cannot target remote victims.

*Other Administration Tools.* Another remote administration tool by Absolute Software reportedly contains security problems related to authentication. Comptrace [1], a theft tracking and recovery tool, uses proprietary code in the system BIOS to resist removal. This support can apparently be exploited by an attacker to make malware harder to detect and remove [9].

Other tools that provide remote desktop control have adopted design philosophies that differ from the approach used in Absolute Manage. Apple Remote Desktop [3] and the Windows Remote Desktop Connection feature [8] normally display prominent notifications that the system is under remote control.

In contrast, Back Orifice [12] is a remote administration program that is designed to hide from users. Due to its potential for malicious use, a number of antimalware programs have added it to their blacklists. Indeed, the distinction between remote administration tools and malware like spyware, back doors, and bots can be a fine line—sometimes, the only difference is who is intended to be in control.

Lest we forget, the Internet’s oldest tool for remote administration, `telnet` [7], used no cryptography at all. Incredibly, it took nearly 30 years for a secure alternative [14] to catch on.

## 7 Conclusions

In this paper, we revealed critical security vulnerabilities in Absolute Manage and demonstrated how attackers could harness them to cause widespread damage. We used these problems as a case study to discuss the broader risks of remote administration software and how they might be mitigated. The blatant vulnerabilities in Absolute Manage suggest that this class of remote administration programs requires greater security scrutiny, particularly in light of the danger such software can pose when commanded by unauthorized third parties. Secure authentication is a necessity, of course, but such products should be further strengthened by employing defensive design and programming techniques.

**Acknowledgments.** The authors are grateful to Scott Wolchok for his many insights and invaluable technical contributions, and for submitting the paper after the rest of us had fallen asleep. We thank the anonymous reviewers for their constructive feedback, and Jaeyeon Jung for shepherding the work to publication. We also wish to thank Ari Feldman, Ed Felten, Bob Lougheed, Gordon Lyon, Hovav Shacham, Eric Wustrow, and the students of Michigan’s winter 2010 Computer and Network Security class for helpful discussions and other assistance.

## References

1. Absolute Software. Absolute Manage Web Site,  
[http://www.absolute.com/en\\_GB/products/absolute-manage](http://www.absolute.com/en_GB/products/absolute-manage)
2. Absolute Software. Absolute Software Acquires LANrev (December 3, 2009),  
<http://www.absolute.com/company/pressroom/news/2009/12/lanrev>
3. Apple. Remote Desktop 3, <http://www.apple.com/remotedesktop/>
4. CWE/SANS. 2010 Top 25 Most Dangerous Programming Errors,  
<http://cwe.mitre.org/top25/>
5. Lyon, G.F.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure, USA (2009)
6. Howell, J., Schechter, S.: What You See is What They Get: Protecting Users from Unwanted Use of Microphones, Cameras, and Other Sensors. Web 2.0 Security and Privacy (2010)
7. Postel, J., Reynolds, J., Reynolds, J.: Telnet protocol specification. STD 8, RFC 854 (May 1983)
8. Microsoft. Connect to Another Computer Using Remote Desktop Connection,  
<http://windows.microsoft.com/en-us/windows-vista/Connect-to-another-computer-using-Remote-Desktop-Connection>.
9. Ortega, A., Sacco, A.: Deactivate the Rootkit: Attacks on BIOS Anti-Theft Technologies. Blackhat (2009)
10. Robbins, B.J., et al.: Complaint Against Lower Merion School District (February 16, 2010),  
<http://docs.justia.com/cases/federal/district-courts/pennsylvania/paedce/2:2010cv00665/347863/1/>
11. Schneier, B.: Description of a new variable-length key, 64-bit block cipher (Blowfish) In: Fast Software Encryption, pp. 191–204 (1993)
12. Sir Dystic. Back Orifice, <http://www.cultdeadcow.com/tools/bo.html>
13. stryde.hax and Aaron Rhodes. The Spy At Harrington High (February 2010),  
<http://strydehax.blogspot.com/2010/02/spy-at-harrington-high.html>
14. Ylonen, T.: SSH-secure login connections over the Internet. In: Proceedings of the 6th USENIX Security Symposium, pp. 37–42 (1996)
15. Zetter, K.: School Spy Program Used on Students Contains Hacker-Friendly Security Hole. Threat Level (May 2010),  
<http://www.wired.com/threatlevel/2010/05/lanrev/>

# A Protocol for Anonymously Establishing Digital Provenance in Reseller Chains

## (Short Paper)

Ben Palmer, Kris Bubendorfer, and Ian Welch

School of Engineering and Computer Science  
Victoria University of Wellington  
`{ben,kris,ian}@ecs.vuw.ac.nz`

**Abstract.** An increasing number of Internet traders exclusively sell digital products. These digital products can include media files, licenses, services, or subscriptions. We consider the concept of digital provenance in reseller chains. The goal of this work is to provide an honest customer with a guarantee on the origin and ownership history for a digital item *even when the reseller they are dealing with is untrusted*. We introduce a protocol called the Tagged Transaction protocol which uses a third party called the Tag Generation Centre (TGC) to provide a method for honest customers to check they are purchasing a legitimate item, anonymity for customers and resellers, a method for customers to resell items they have purchased to other customers, and verification of the TGC.

## 1 Introduction

Amazon, iTunes, and domain name resellers, such as GoDaddy, only exist as on-line traders with no physical stores. These digital products and services can include digital media or more abstract products such as an access ‘right’, a license, a service, or a subscription.

We consider the concept of digital provenance in reseller chains. The goal of this work is to provide an honest customer with a guarantee on the origin and ownership history for a digital item *even when the reseller they are dealing with is untrusted*. To check the origin of a digital item we need to provide the customer with a guarantee that the item has originally been purchased from the correct supplier. Checking the ownership history for a digital item involves checking that at every step in the reseller chain it has correctly been purchased and only sold on to one reseller or customer. We look at methods for establishing digital provenance anonymously to prevent any party involved in the protocol (or an observer) from building up detailed records of the identities of customers and resellers.

Most Internet resellers use a digital certificate to prove their identity and to provide information on their physical location and contact details. The digital certificate does not provide any mechanism or guarantees though which the provenance of goods might be established.

A simple approach to achieving digital provenance in reseller chains is to introduce a license server that acts as a trusted third party. This license server can check at every step in the transaction that the item is legitimate and that it has not been sold to multiple customers. This license server would then have control over a large amount of data both on the details of transactions conducted and the identities of the parties involved. A better option is to provide verification of the actions of any third party in the protocol without reducing privacy.

To establish digital provenance anonymously in reseller chains we have developed a protocol called the Tagged Transaction Protocol. The protocol does *not* provide enforcement of licenses. The tagged transaction protocol uses ‘tags’ to establish provenance. If a reseller can provide a customer with a valid tag, the customer can have confidence that the reseller has sold them a properly licensed item. The tagged transaction protocol uses a Tag Generation Centre (TGC) to sign and check tags. The main four contributions of the Tagged Transaction protocol are: (1) a method for customers to check they are purchasing a legitimate item, even from an untrusted reseller; (2) selectable anonymity for customers, resellers, and suppliers; (3) mechanisms to verify the actions of the TGC so it is not required to be a trusted third party; and (4) customers can act in the role of a reseller and on-sell items.

## 2 The Tagged Transactions Protocol

The tagged transaction protocol provides a mechanism for establishment of the provenance of a digital item while preserving the anonymity of resellers and customers (and optionally suppliers). We use a Tag Generation Centre (TGC) to generate and sign tags. The tagged transaction protocol does not involve payment and we assume payment is made through an external third party.

### 2.1 Threat Model

A malicious reseller has several ways to try and defraud both the supplier and the customer. We have informally grouped these actions in to the following categories. Spoofing is where the reseller claims to be the supplier or tries to subvert the protocol to make it appear that they are the supplier. Counterfeiting is where the reseller sells the customer an item but never buys it from the supplier. Counterfeiting can be further divided into: fabrication where the reseller tries to forge a license for an item from scratch (or based on the structure of other licenses), cloning where the reseller tries to sell a license they have purchased from the supplier to multiple customers, and network sniffing where the reseller replays a legitimate license. We also class identity revelation where the customer learns the identity of one of the resellers (or optionally the supplier) that is not its neighbour in the chain as a category of attack.

We assume the reseller is a polynomially bounded active adversary. The customer and supplier are also assumed to be polynomially bounded. If the reseller is selling a customer item  $x$ , then we assume the reseller cannot collude with

the supplier for  $x$ , but can try to impersonate the supplier for  $x$ . We assume the customer does not collude with the reseller.

## 2.2 Definitions and Techniques

The modified El-Gamal signature scheme for digital signatures is used because it has been proved secure in the random oracle model against adaptive chosen message attacks [1]. We use the notation of  $\{A\}_{sk_B}$  to denote the message  $A$  signed using the key  $sk_B$ . All mathematical operations are computed modulo a large prime  $p$  in the group of integers  $\mathbb{Z}_p$  closed under multiplication unless otherwise stated. We use the notation of  $pk$  to represent a public key and  $sk$  to represent a private key where  $pk = g^{sk} \bmod p$ . The value  $g$  is a generator for the group  $\mathbb{Z}_p$  and  $q$  is some large prime where  $q|p - 1$  ( $q$  divides  $p - 1$ ). The parameters  $p$ ,  $q$ , and  $g$  are global parameters. The TGC also generates its private key  $sk_{TGC}$  and public key  $pk_{TGC} = g^{sk_{TGC}} \bmod p$ .

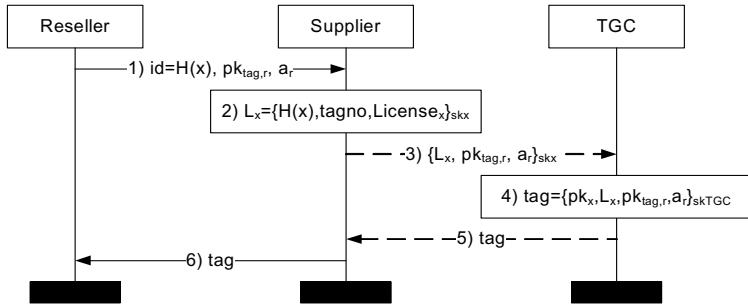
A tag is a 4-tuple  $\{A = pk_x, B = L_x, C = pk_{tag,r} = g^{sk_{tag,r}} \bmod p, D = a = g^z \bmod p\}$  with elements:  $A = pk_x$  is the public key for the item,  $B = L_x = \{id = H(x), tagno, License\}_{sk_x}$  is a license signed with the secret key for the item,  $C = pk_{tag,r} = g^{sk_{tag,r}} \bmod p$  is the one time public key for the reseller  $r$  and tag  $tag$ , and  $D = a = g^z \bmod p$  is the commitment value used in the zero knowledge proof of knowledge of the one time private key for the reseller.

The license  $B = L_x = \{id = H(x), tagno, License\}_{sk_x}$  contains the identity of the item  $id = H(x)$  and the unique tag number. The identity of the item  $id = H(x)$  is calculated using a well known hash function  $H$ . In this paper, we assume that an item can be uniquely identified by its hash value. To prevent the TGC being able to link actions done by a single reseller together, the reseller will use a separate one time private and public key pair for every tag. We denote this one time tag key as public key  $pk_{tag,r}$  and secret key  $sk_{tag,r}$  for reseller  $r$  and tag  $tag$ .

## 2.3 Stage 1 - Supplier Generating Tag with TGC

Before an item tag may be generated a one time registration phase must be completed. The supplier calculates the identity of the item  $id = H(x)$  and a public ( $pk_x$ ) and secret ( $sk_x$ ) key for the item and registers the item and public key with the TGC. The TGC may convince itself with out of band checks that the party registering the item is indeed the rights owner. Where suppliers wish to remain anonymous, registration messages are sent via an anonymous communication channel. The TGC cannot verify ownership due to the anonymous channel and uses a first-in first-registered default. Anonymous channels are shown as the dotted lines in Figures 1 and 2.

The generation of a new tag for an item by the supplier takes place in the six steps shown in Figure 1, specifically: (1) The reseller sends a purchase request to the supplier containing the identity of the item they wish to purchase  $id = H(x)$ , the one time public key for the tag  $pk_{tag,r}$ , and a commitment value  $a_r = g^{z_r} \bmod p$ ; (2) the supplier then creates a signed tag request containing the

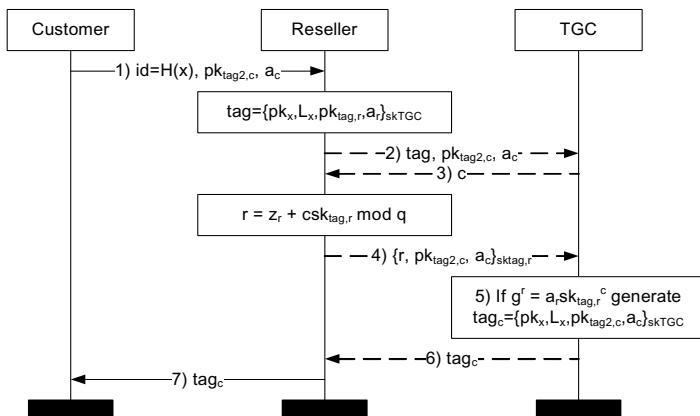


**Fig. 1.** Supplier Generating Tag with TGC

license  $L_x$  for the item signed using  $sk_x$ , the one time public key  $pk_{tag,r}$ , and the commitment value  $a_r$  all signed by  $sk_x$ ; (3) the supplier sends the signed tag request to the TGC; (4) the TGC checks the tag number contained in the signed license  $tagno$  has not been used for this item before and then constructs and signs  $tag = \{pk_x, L_x, pk_{tag,r}, a_r\}_{sk_{TGC}}$ ; (5) the tag is now sent from the TGC to the supplier; and (6) the supplier passes it on to the reseller. The reseller checks the tag has been signed by the TGC, that the license is for the correct item, and that the tag contains the correct one time public key and commitment value.

#### 2.4 Stage 2 - Reseller Instantiating Tag with TGC

The instantiation by a reseller of a new tag takes place in the seven steps shown in Figure 2, specifically: (1) the customer sends a purchase request comprised of the identity of the item they wish to purchase, the one time public key for



**Fig. 2.** Reseller Passing Tag to Customer

the tag  $pk_{tag,c}$ , and a commitment value  $a_c$ ; (2) the reseller sends the one time public key for the tag  $pk_{tag,c}$  and the commitment  $a_c$  to the TGC along with their tag for the item; (3) the TGC then checks that the one time public key has not previously been used for this item and tag number, the TGC and reseller then take part in a zero knowledge proof of knowledge of a discrete logarithm [2] where the prover (reseller) proves to the verifier (TGC) that they know the value  $sk_{tag,r}$  such that  $pk_{tag,r} = g^{sk_{tag,r}}$  using the commitment value  $a_r = g^{z_r}$  (from the tag) and the TGC sends the challenge value  $c$  to the reseller; (4) the reseller calculates  $r = z_r + csk_r \bmod q$  and sends  $\{r, pk_{tag2,c}, a_c\}sk_{tag,r}$  to the TGC; (5) the TGC checks that the message is signed using  $sk_{tag,r}$  and that  $g^r = a_r pk_{tag,r}^c$  as  $g^r = g^{z_r + csk_{tag,r} \bmod q} = g^{z_r} g^{csk_{tag,r}} = a_r pk_{tag,r}^c$ .

The response value  $r, pk_{tag2,c}, a_c$  is signed using the one time secret key for the tag  $sk_{tag,r}$  to prove to the TGC that the reseller owns this tag. The TGC checks the tag sent to it by the reseller was signed using its private key  $sk_{TGC}$  and the values  $pk_{tag2,c}, a_c$  and then uses the zero knowledge proof to detect if the tag has been replayed. If the tag has not been submitted to the TGC by a reseller before, the TGC saves the identity of the item  $H(x)$ , the public key  $pk_{tag,r}$ , the commitment  $a_r$  and the proof transcript  $c, r$  of the zero knowledge proof. If this tag is used a second time, the TGC will have two proof transcripts  $c_1, r_1$  and  $c_2, r_2$  that have been used with the same commitment value  $a_r$ . The TGC can then extract the one time secret key  $sk_{tag,r}$  by computing  $g^{r_1}/g^{r_2} = g^{z_r} pk_{tag,r}^{c_1}/g^{z_r} pk_{tag,r}^{c_2} = pk_{tag,r}^{c_1 - c_2}$  and  $log(gpk_{tag,r}) = r_1 - r_2/c_1 - c_2 = sk_{tag,r}$ . The TGC can prove that the reseller has replayed the tag to any third party by presenting the two proof transcripts.

If the TGC does not detect replay, it generates a new tag that contains the public key  $pk_{tag,c}$  and commitment value  $a_c$  of the customer or second reseller. This tag is then sent to the reseller (6) and then the reseller forwards it to the customer (7). The customer then checks that the tag was signed by the TGC and that the license  $L_x$  is valid and signed by  $sk_x$ . This step can be repeated by the customer to resell this item to another party.

### 3 Security Analysis

**Spoofing:** An adversary could try and spoof the supplier in one of two ways. It could try and forge a request from the supplier to create a new license or register the item with the TGC. Forging a request for the supplier to the TGC is equivalent to the fabrication attack described below. To prevent an adversary in control of the network from modifying the registration message it is encrypted using the public key of the TGC  $pk_{TGC}$ . The checks done by the TGC when a new item is registered should prevent an adversary from being able to register an item they do not control.

**Fabrication:** If the TGC receives a message signed by  $sk_x$  requesting a new tag that was not sent by the supplier, then either the adversary knows  $sk_x$ , or the adversary has been able to forge a message signed by  $sk_x$ , or the adversary is replying a previously sent message. If the adversary replays a previously sent

message, the TGC will not generate a new tag as the tag number for the item has already been used. The chance of an adversary being able to forge a message signed by  $sk_x$  is negligible due to the signature scheme used.

**Cloning:** If an adversary has run the protocol twice with the same input  $tag$  and produced two different tags  $tag_1$  and  $tag_2$  with different one time keys  $pk_1$  and  $pk_2$  and different commitments  $a_1$  and  $a_2$  where  $tag$ ,  $tag_1$ , and  $tag_2$  are all signed by the TGC then either the TGC has generated two tags or the reseller has tampered with the input  $tag$  to change the one time key or commitment value. The TGC will not generate two tags as it will be able to detect replay with two separate runs of the zero knowledge proof with the same one time key  $pk_{tag}$  and commitment  $a_{tag}$  and different challenge values  $c_1$  and  $c_2$ . The chance of the reseller being able to tamper with the input tag  $tag$  to change the one time key or commitment value is negligible due to the signature scheme used.

**Network Sniffing:** The adversary could intercept a tag that has already been generated by the TGC and send it to a customer or try and generate a new tag from the tag they have intercepted. In the first case, the chance of the tag containing the correct one time key  $pk_c$  and commitment value  $a_c$  chosen by the customer is negligible. In the second case, the adversary would have to be able to generate the message  $\{r, pk_{tag2,c}, a_c\}_{sk_{tag,r}}$  and send it to the TGC as part of the zero knowledge proof. If the adversary does not know the secret key  $sk_{tag,r}$  then the chance of it being able to create the message is negligible.

## 4 Anonymity and Verification of the TGC

We cannot always assume that our trusted TGC is trustworthy because it maybe subverted by the owners or third-parties through bribery, seizure or being compromised by attackers. An un-trustworthy TGC could be used to reveal the identify of parties using the TGC or allow violation of the security properties provided by our protocol.

Identity can be protected by allowing communication with the TGC via an anonymity service such as TOR [3]. This would prevent even the TGC from knowing the identity of the suppliers and resellers limiting the amount of information it could maliciously reveal. A downside would be that selective revelation of identity could not be easily provided.

Verification that the TGC faithfully implements the security protocol can be achieved by requiring the TGC to publish all its actions to a public bulletin board. When a customer gets sent a tag from the reseller, it verifies that the tag has been correctly generated by the TGC by checking the operations the TGC has done to generate the tag. It also follows the actions of the TGC on the tag that the reseller had before generating the tag for the customer and so on down the chain of resellers until it reaches the initial tag creation by the supplier. Privacy is maintained because the only information leaked to the customer is the number of resellers the tag has passed through from the supplier to it.

## 5 Performance

The Tagged Transaction protocol makes use of digital signatures and zero knowledge proofs. We examine the computational complexity of the Tagged Transaction protocol based on the number of modular operations in Table 1 where  $n$  is the complexity of modular exponentiation and  $m$  is the complexity of modular division. This table does not take in to account any extra computations required to verify the actions of the TGC.

The Tagged Transaction protocol relies on the use of the TGC to generate and sign tags. As the actions of the TGC can be verified, there is no requirement for the TGC to be operated by a trusted party and we envisage many TGCs operating. The supplier for the item can select a TGC when it first generates the tag for an item based on previous relationships, results of verification of the TGCs past actions, availability, or some other metric.

**Table 1.** Complexity of Operations in the Tagged Transaction Protocol

Operation	Customer	Reseller	Supplier	TGC	Total
Generating Tag		$5n$	$5n + 2m$	$7n + m$	$17n + 3m$
Reseller Generating Tag	$8n$	$7n + m$		$9n + m$	$24n + 2m$
Verifying Tag	$6n$				$6n$

## 6 Related Work

The Paradiso system lets customer purchase not only the songs and videos from content providers but also reseller rights [4]. To prevent malicious behaviour, a Trusted Computing Module (TCM) is used to store encryption keys and to perform private key operations in secure memory while the Tagged Transaction Protocol does not rely on any trusted hardware.

The IEEE working group P1817 has produced a document suggesting a standard which is similar to the Paradiso system where customers can resell items they have purchased as they can with physical products [5]. While the P1817 standard does provide options for customers to resell content it relies on a trusted player to store cryptographic keys.

Serban, Chen, Zhang, and Minsky introduce the concept of a decentralised electronic marketplace (DEM) where transactions are subject to a set of trading rules [6] implemented using a mechanism called Law Governed Interaction (LGI). In LGI, a law is formulated using an event-condition-action pattern. Apart from the agents taking part in transactions in the marketplace, there are also a set of trusted controllers that enforce the law of the marketplace. Although these controllers are distributed, there is no method to verify their actions.

The Idemix system [7] developed by Camenisch and Van Herreweghen is an implementation of an anonymous credential system [8]. Both the Tagged Transaction protocol and one-show anonymous credentials provide replay detection

but there are several differences. In anonymous credentials, when using multiple verifiers, replay detection is performed after the fact whereas in the Tagged Transaction protocol it is done live by the TGC. In the Tagged Transaction protocol a reseller could transfer a tag from themselves to another reseller but this is not possible using anonymous credentials. The tagged transaction protocol also provides optional supplier anonymity and allows the resellers to generate tags when the supplier is offline.

## 7 Conclusions and Future Work

In this paper we have presented the Tagged Transaction Protocol for establishing digital provenance anonymously in reseller chains. The tagged transaction protocol provides a method for honest customers to check they are purchasing a legitimate item, provides selective anonymity for customers and resellers (and optionally suppliers), provides mechanisms to verify the actions of the TGC, and allows customers to resell items. Future work includes completing a security analysis using the FDR model checker and modification of the protocol to prevent the leakage of information regarding the distance from the supplier of the reseller when the TGC is verified. We also intend to look for applications of the tagged transaction protocol in other areas.

## References

1. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: Maurer, U.M. (ed.) *EUROCRYPT* 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
2. Schnorr, C.-P.: Efficient Identification and Signatures for Smart Cards. In: Quisquater, J.-J., Vandewalle, J. (eds.) *EUROCRYPT* 1989. LNCS, vol. 434, pp. 688–689. Springer, Heidelberg (1990)
3. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium (2004)
4. Nair, S.K., Popescu, B.C., Gamage, C., Crispo, B., Tanenbaum, A.S.: Enabling drm-preserving digital content redistribution. In: CEC 2005: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, pp. 151–158. IEEE Computer Society, Washington, DC (2005)
5. P1817, I.W.G.: Initial technical description of the p1817 standard. Technical report. IEEE (2010)
6. Serban, C., Chen, Y., Zhang, W., Minsky, N.: The concept of decentralized and secure electronic marketplace. *Electronic Commerce Research* 8(1-2), 79–101 (2008)
7. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: CCS 2002: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 21–30. ACM, New York (2002)
8. Camenisch, J.L., Lysyanskaya, A.: An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) *EUROCRYPT* 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

# Impeding Individual User Profiling in Shopper Loyalty Programs

Philip Marquardt, David Dagon, and Patrick Traynor

Converging Infrastructure Security (CISEC) Laboratory

Georgia Tech Information Security Center (GTISC)

School of Computer Science, Georgia Institute of Technology

Atlanta, GA 30332

{pmarq, dagon, traynor}@cc.gatech.edu

**Abstract.** Shopper loyalty club programs are advertised as a means of reducing prices for consumers. When making a purchase, a customer simply scans their keyring tag along with the items they intend to buy and is granted a reduction in the total price. While the use of these cards results in a visible reduction in price, customers are largely unaware of the privacy implications of such discounts. In particular, the ability to link all purchases made by an individual customer allows retailers to develop detailed profiles that may reveal sensitive information, especially if leaked or sold to third parties. In this paper, we present ShopAnon, a mobile phone-based infrastructure designed to help consumers partake in shopper loyalty programs without allowing their transactions to be linked by a retailer. ShopAnon displays legitimate but random barcodes for specific retailers on each execution, and provides a number of operational modes that respond to the changing availability of resources and the specific privacy concerns of the user. Communications between the application and the database storing the barcodes occurs using an Oblivious Transfer protocol to prevent our system from exposing the barcode received by a requester. We design, implement and characterize the behavior of our application on the iPhone mobile platform, and demonstrate its practical efficiency (i.e., the ability to render random tags in less than 0.25 seconds via 802.11 links and approximately 3.9 seconds via a 3G cellular connection). Through this, we provide a powerful tool through which customers can improve their privacy in a retail environment.

## 1 Introduction

Businesses have launched a number of efforts in order to encourage customers to select themselves over the competition. Many stores offer a variety of indulgences (e.g., coffee stands, gourmet food, complimentary personal shoppers) in attempts to sell a “shopping experience” instead of just the individual items on their shelves. More recently, a large number of companies have tried to retain their customer base through the use of shopper loyalty programs. These marketing efforts are structured to provide regular shoppers with small discounts on their purchases once they become members of the program. Customers claim these discounts by presenting a scannable barcode, usually in the form of key tags/fobs, that links their purchases to their identity. Because presenting a membership tag results in a tangible savings, customers voluntarily sign

up for and participate in such programs without considering the potential consequences. In particular, by allowing a store to aggregate the purchases shoppers make, these programs enable businesses to profile and track their customers. As these profiles often contain sensitive information about individual customers (e.g., medications, products that may cause embarrassment) and may potentially be leaked or resold to third parties, these programs represent significant threats to consumer privacy.

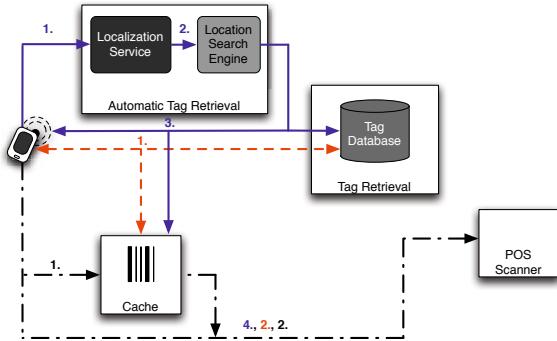
In this paper, we present a means by which customers can participate in shopper loyalty programs with a greatly reduced risk of being profiled. We argue that a customer can be rewarded for their loyalty to a particular retailer without the wholesale surrender of their privacy. We achieve these ends through *ShopAnon*, an application designed for mobile phones that anonymously downloads loyalty club tags for its users. ShopAnon does not simply provide a trusted third party through which barcodes can be filtered. Instead, it allows users to automatically receive barcodes based on their current physical location while implementing an *Oblivious Transfer* protocol to prevent our infrastructure from learning the identity of the card served to the requester. Through these mechanisms, ShopAnon allows customers that wish to remain anonymous to be rewarded for patronizing the retailer of their choice. In so doing, we make the following contributions: 1) Develop an architecture capable of impeding the profiling of individuals in shopper loyalty programs, 2) Create a number of modes so as to provide consumers with operational robustness, 3) Implement, test and characterize our application on the iPhone platform.

Note that our system is robust against campaigns by retailers designed to identify and remove tags used in our system. Specifically, *we experimentally demonstrate that currently deployed systems rely on offline databases, allowing us to use cards for months after they have been deactivated*. Without the large-scale replacement of the cash registers and backend database systems with online systems, our approach will be able to assist users in protecting their privacy.

## 2 Threat Model

ShopAnon fundamentally relies upon the ability of consumers to use their mobile phone while making a purchase. There is no incentive for a store to support the ShopAnon program. Fortunately, other applications already allow users to store all of their shopper loyalty cards on their mobile phones [2]. These programs differ from ShopAnon in that the user remains traceable because the tag presented at each transaction is the same. However, the existence of these other programs allows customers to run ShopAnon without necessarily appearing to be avoiding profiling. If a retailer decides to prevent all such applications or devices from being used in their stores, our application is unlikely to be able to help protect the user. However, we believe that such a rule would be generally unenforceable and is more likely to irritate customers.

Retailers can download our software and potentially learn the values of some of the tags used in our system. Because of cost and highly distributed nature of current infrastructure, the databases responsible for processing shopper purchases function as an offline, batch-processing system. All updates to the system are completed by executing a bulk update during off-peak times. This results in the ability for the scanner to accept



**Fig. 1.** A high-level overview of the ShopAnon architecture. The phone determines its location and queries a search engine for nearby businesses within a fixed radius. The response is filtered through the list of known shopper loyalty companies and then a request is sent to the tag database. The client then performs oblivious transfers with the database and receives the appropriate tag. Note that a client may skip the search phase by identifying the store manually (dashed red line) or by using a cached card (dot-dashed black line.) Barcodes are then displayed on the screen of the mobile phone and read by the Point-of-Sale (POS) scanner.

any shopper card that displays the correct barcode formatting for the store’s shopper program. Supporting the instantaneous revocation of a card for any reason would require that these systems operate in an online fashion; however, such a change would be prohibitively expensive as it would require enabling all cash registers to access the database during each transaction involving a loyalty club card. Section 4.3 experimentally demonstrates the offline nature of current systems.

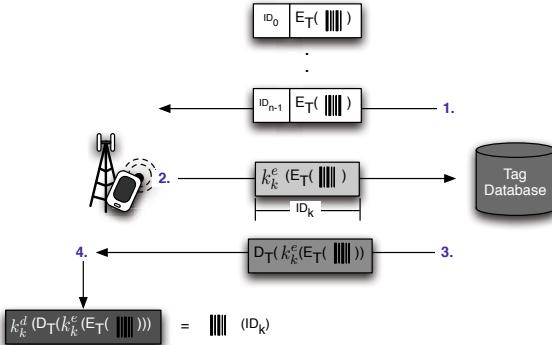
### 3 System Architecture

In this section, we present the details of the ShopAnon architecture, a high-level description of which is shown in Figure 1.

**Location-Based Search.** Knowledge of one’s physical location is a necessary precondition for automatically delivering relevant barcodes. Such information must be translated into the identity of a store before it can be used. We rely on location-based search engines (e.g., Google Maps, Yahoo! Maps, MapQuest) to assist in this process; however, there are a number of ways in which such systems can be used depending on the privacy concerns of the user [16,11,12]. In general, Shopanon will request a user’s GPS coordinates and query the local-based search engine with a specific or general query provided by the user for finding stores within the specific search radius.

#### 3.1 Tag Retrieval

Preventing a customer from being tracked by a loyalty rewards program has little practical value if we simply shift responsibility for tag dissemination to a “trusted



**Fig. 2.** A 1-out-of- $n$  oblivious transfer as used in ShopAnon. The client 1) receives  $n$  encrypted barcodes from the Tag Database. Using the prepended identifier, the client determines which tag it would like to receive, 2) removes the identifier, encrypts the tag with its key (modular multiplication with random  $k$ ) and returns it to the tag database. The Tag Database then 3) decrypts the received tag, and sends the resulting ciphertext to the client. Finally, 4) the client decrypts the received ciphertext (modular division by  $k$ ) and recovers the desired barcode. Note that the Tag Database can not determine the identity of the tag selected by the client because it can not fully decrypt any of the ciphertexts.

third party". For instance, nothing prevents such an entity from changing their terms of service and eventually selling the data collected regarding user request patterns. Moreover, a system built upon this assumption fails to prevent insiders, intruders or those with access to server logs from similarly exfiltrating such data. We aim to provide a service in which correlation between a user and a tag is difficult for *anyone* except the user.

We address these concerns through the use of *Oblivious Transfer* (OT). This cryptographic primitive allows a server Alice to offer a receiver Bob  $k$ -out-of- $n$  pieces of data without allowing Alice to determine the identity of the  $k$  pieces delivered to Bob while preventing Bob from learning more than  $k$  pieces of data. *This approach is advantageous as it not only prevents our server from learning the actual tag delivered to a mobile for a given transaction, but also because it prevents an adversary capable of compromising the server from using log data to learn about a client's past behaviors.*

We selected an OT scheme proposed by Huang et al. [7]. This work is based on RSA and relies on the security guarantees of this algorithm. Unlike the above definition, Alice now sends Bob a series of messages  $m_0, \dots, m_{n-1}$ . Alice encrypts these messages such that  $X_0 = m_0^e, \dots, X_{n-1} = m_{n-1}^e$ . Bob then selects  $k$  of the  $n$  received messages, selects  $k$  random numbers and encrypts the selected subset to create the set  $Y_0 = X_{k0} \cdot k_0^e, \dots, Y_{k-1} = X_{kk-1} \cdot k_{k-1}^e$ . Alice receives the  $k$  messages from Bob, which she can not uniquely identify from the set  $M$ , and decrypts them such that  $C_0 = Y_0^d, \dots, C_{k-1} = Y_{k-1}^d$ . Bob then receives these messages, removes his secret and is able to retrieve the contents of the  $k$  selected messages. Figure 2 provides an overview of the 1-out-of- $n$  scheme used in this work.

The scheme proposed by Huang et al. is attractive for a number of reasons. First, the intuitive nature of this scheme makes its implementation relatively straightforward.

- |  |   |
|--|---|
| 0. $C \rightarrow D : V_0, \dots, V_{n-1}$   | $C, D, V$ : Client, Database Server, Vendor                             |
| 1. $D \rightarrow C : V_0   \langle B_{V_{0,i}}, t \rangle_{k_D^+}, \dots, V_{n-1}  $    | $B, Y$ : Barcode, Client secret value                                   |
|  | $k_D^+, k_D^-$ : Public and private key for Database $D$                |
| 2. $C \rightarrow D : \langle B_{V_{j,i''}} \rangle_{k_D^+} * Y$                         | $\langle B \rangle_{k_D^+}$ : $B$ encrypted with public key $k$ for $D$ |
| 3. $D \rightarrow C : \langle \langle B_{V_{j,i''}} \rangle_{k_D^+} * Y \rangle_{k_D^-}$ |   |
| 4. $C \rightarrow V_j : B_{j,i''}$   |   |

**Fig. 3.** The ShopAnon protocol and variable definition. We formally define the interaction between the application and the Tag Database. Each message corresponds to the messages in Figure 2. Note that Message 0, the initial request, is not included in the previous figure.

Moreover, unlike related OT schemes based on Diffie-Hellman, we can parameterize our implementation to be very efficient based on small values for  $e$  in RSA while remaining resistant to known small exponent attacks. For a  $k$ -out-of- $n$  system based on this cryptosystem, a client is required to perform  $4k$  modular multiplications, whereas the server performs  $k$  modular exponentiations and  $2n$  modular multiplications. Given that other related schemes generally require  $O(kn)$  modular exponentiations by the server and  $O(k)$  modular exponentiations by the client [5,9,13,15], the selected scheme is more appropriate for potentially constrained mobile devices.

## 4 System Evaluation and Analysis

### 4.1 Experimental Setup

We used the following development, software, and hardware environments to develop and test the ShopAnon application. The client application was developed for the iPhone 3G OS v3.1.2. Xcode v3.1.2 was used as the development environment for both the client and server applications. During the development phase, iPhone simulator v3.1 was used for verifying correct functionality. All development software was run on an Apple MacBook Pro with 2.2GHz Intel Core 2 Duo processor, 4 GB 667 MHz DDR2 SDRAM, running Mac OS X v10.6.2 (Snow Leopard). The ShopAnon application is written in Objective-C with some C functionality and compiled using GCC v4.2. We cross-compiled and used GMP v4.3.1 for cryptographic functions and pseudorandom number generation. Figure 4 shows the working application.

### 4.2 Microbenchmarks

Our first set of experiments ran the tag retrieval portion of ShopAnon over an 802.11g wireless connection. We ran a total of 10 experiments for each of the above data points and calculated 95% confidence intervals of at least one order of magnitude smaller than the mean. ShopAnon rendered the received barcode in an average of  $0.229 \pm 0.0166$  seconds after selecting the vendor for which a barcode was needed. The cryptographic overhead related to this delivery was also extremely low - the application was able to



**Fig. 4.** The ShopAnon user interface. In these images, the user selects “Find By Store Name”, selects a particular retailer and automatically receives one of the barcodes stored for that vendor in the Tag Database.

perform encryption and decryption in  $0.023 \pm 0.002$  seconds and  $0.008 \pm 0.0002$  seconds, respectively. Accordingly, our fears of significant overhead in order to perform OT operations were alleviated. The performance of our protocol over the 3G UMTS cellular data link was significantly different from 802.11g. All network operations conducted via the cellular link were at least an order of magnitude more expensive than WiFi, with a total protocol execution time of  $3.939 \pm 0.560$  seconds. These results are expected based on the architecture of the network. In particular, the high cost of connection establishment [14] and the use of scheduling algorithms such as Proportional Fair that favor short high bandwidth bursts as opposed to uniform distribution of traffic with low latency [6].

#### 4.3 Field Testing Experience

**Card Collection Experience.** The barcodes made available in ShopAnon represent real loyalty club card accounts. We have not attempted to reverse-engineer the algorithms used to generate customer identifiers and have therefore not included any “forged” cards in our system. Instead, individuals in our research group visited a range of stores offering these cards and retrieved them. Our experiences were mixed, even across multiple locations within the same chain. Whenever possible, group members requested receiving tags without any name associated with them. While some of the stores we visited provided such cards without question, the majority of stores required that we provide some identifying information. Most employees, however, knowingly allowed us to enter false information. This reaction was expected and has been observed by other parties concerned about privacy in this space [3]. Only one out of the five chains we visited required that we provide a state-issued identification card, such as a driver’s license. However, we were able to acquire a card from the same chain online without the need to provide this verifiable information. At no time were our requests for cards rejected. Moreover, regardless of the information that was entered at the time of activation, all of the cards that we have collected continue to allow us to receive the

discounts at each of the respective stores. We use these collected cards to populate our Tag Database and, as mentioned above, plan to allow users to submit their own cards to the system as well.

**Card Invalidation Experience.** To gain insight on how the barcodes are processed during the checkout time, we chose two barcodes from two different retailers and tried to manually invalidate them. Our hypothesis is the POS systems are not connected to an online central database, so either the cards can not be deactivated or all deactivated cards are downloaded by all stores periodically making deactivation processing times lengthy. Such invalidation problems are well known in the field of certificate revocation [8].

Retailers were contacted and provided with the barcodes requiring invalidation. One operator informed us the deactivation process would take four to six weeks while another operator indicated immediate processing. As expected, both cards were tested for six months and continued to succeed in receiving discounts to qualified purchases. This testing proves that the system being used to process discount cards is completely offline, validating the assumption made in our threat model. Given that the cost of retrofitting all of the cash registers in a shopping chain and the backend database to support an system capable of processing thousands of transactions per second is very high, ShopAnon is resistant to deactivation attacks. While vendors may eventually purchase such highly-capable infrastructure, our solution is an important first step in protecting consumer privacy in this space.

**Application Usage Results.** Having collected a number of valid shopper loyalty club tags and entered them into the Tag Database, we then sought to determine how well such tags could be read by barcode scanners. While other tag storage applications have already shown that it is possible to read barcodes from an iPhone screen [2], users regularly complain that such applications suffer from poor detection and accuracy. These tests accordingly attempt to better quantify such issues. We note that like many other mobile phones, the iPhone tested in these experiments used a scratch-resistant plastic film on the screen. Our experiments were conducted both with and without this cover.

Our attempts to detect barcodes were extremely successful with handheld scanning guns. We specifically ran this set of tests using the Metrologic Ms1690 Focus handheld scanner, which is also used by a number of major retailers on a national scale. In these tests, barcodes were readable ten out of ten times when the screen protector was removed; however, our results were not as consistent when the screen protector was in place with only seven out of ten attempts being successful. We also tested the performance of our barcodes against a UniComp PCT2 Price Checker and found that these devices were able to read the barcode rendered by the ShopAnon application nine out of ten times when the screen protector was in place and ten out of ten times when the screen was unobscured.

We also measured traditional flatbed barcode scanners by testing the NCR RealPOS High Performance Bi-Optic Scanner/Scale [10], which is used by major chain stores throughout the country. In spite of multiple attempts from a variety of angles, orientations, distances and in the presence and absence of the screen protector, we were unable to scan the barcode displayed by ShopAnon. This was expected given previously documented complaints [1,4]. Fortunately, most of these systems come with an optional hand scanner, which a customer can request their cashier use on the displayed barcode.

## 5 Conclusion

Shopper loyalty club programs offer their members demonstrably lower prices than those given to non-members. Unfortunately, in the pursuit of these savings, users often unknowingly sacrifice their privacy and allow themselves to be profiled. In this paper, we develop ShopAnon, an infrastructure that allows consumers to receive the benefits associated with such programs while allowing their transactions to remain unlinkable. Using an application running on their mobile phone, consumers download random legitimate barcodes to be presented at the checkout. Such barcodes can be downloaded automatically or with manual assistance and cached to allow for potential offline operation in the future. To demonstrate that this system is practical, we design, implement, measure and field test our application on the iPhone platform and show that users can execute this protocol in a matter of seconds (approximately 3.9 seconds over 3G UMTS links and 0.2 seconds via 802.11g). We then provide a number of options by which more advanced users can further limit their exposure. In so doing, we provide a robust tool and an important first step by which consumer privacy can be protected in a retail environment.

**Acknowledgments.** This work was supported in part by the US National Science Foundation (CNS-0916031). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

1. AppShouter. CardStar iPhone App Review (2009),  
<http://appshouter.com/iphone-app-review/iphone-app-review-cardstar/>
2. CardStar, Inc. CardStar (2009), <http://www.mycardstar.com/>
3. Consumers Against Supermarket Privacy Invasion and Numbering (CASPIAN). Is Big Brother in Your Grocery Cart? (2009), <http://www.nocards.org/>
4. Fischer, B.: AppAdvice: CardStar (2009),  
<http://appadvice.com/appnn/2009/02/review-cardstar/>
5. Gilles, B., Claude, C., Stefan, W.: Oblivious Transfers and Privacy Amplification. *Journal of Cryptology* 16(4) (2003)
6. Holma, H., Toskala, A. (eds.): HSDPA/HSUPA for UMTS. John Wiley & Sons, Ltd. (2006)
7. Huang, H.-F., Chang, C.-C.: A new design for efficient t-out-n oblivious transfer scheme. In: International Conference on Advanced Information Networking and Applications, vol. 2, pp. 499–502 (2005)
8. McDaniel, P., Rubin, A.D.: A Response to Can We Eliminate Certificate Revocation Lists? In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 245–258. Springer, Heidelberg (2001)
9. Naor, M., Pinkas, B.: Efficient Oblivious Transfer Protocols. In: Proceedings of SIAM Symposium on Discrete Algorithms (SODA) (2001)
10. National Cash Register (NCR). NCR RealPOS High Performance Bi-Optic Scanner/Scale (2009), [http://www.ncr.com/products\\_and\\_services/point\\_of\\_sale/pos\\_scanners/index.jsp](http://www.ncr.com/products_and_services/point_of_sale/pos_scanners/index.jsp)
11. Popa, R.A., Balakrishnan, H., Blumberg, A.J.: Vpriv: Protecting privacy in location-based vehicular services. In: Proceedings of the USENIX Security Symposium (2009)

12. Shankar, P., Ganapathy, V., Iftode, L.: Privately Querying Location-based Services with SybilQuery. In: Proceedings of the International Conference on Ubiquitous Computing (2009)
13. Stern, J.P.: A New and Efficient All-or-Nothing Disclosure of Secrets Protocol. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 357–371. Springer, Heidelberg (1998)
14. Traynor, P., McDaniel, P., La Porta, T.: On Attack Causality in Internet-Connected Cellular Networks. In: Proceedings of the USENIX Security Symposium (SECURITY) (2007)
15. Tzeng, W.-G.: Efficient 1-Out-n Oblivious Transfer Schemes. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 159–171. Springer, Heidelberg (2002)
16. Zhong, G., Goldberg, I., Hengartner, U.: Louis, Lester and Pierre: Three Protocols for Location Privacy. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 62–76. Springer, Heidelberg (2007)

# Beyond Risk-Based Access Control: Towards Incentive-Based Access Control

Debin Liu<sup>1</sup>, Ninghui Li<sup>2</sup>, XiaoFeng Wang<sup>1</sup>, and L. Jean Camp<sup>1</sup>

<sup>1</sup> School of Informatics and Computing  
Indiana University, Bloomington, Indiana, US

<sup>2</sup> Department of Computer Science  
Purdue University, West Lafayette, Indiana, US

**Abstract.** In recent years, risk-based access control has been proposed as an alternative to traditional rigid access control models such as multi-level security and role-based access control. While these approaches make the risks associated with exceptional access accountable and encourage the users to take low-risk actions, they also create the disincentives for seeking necessary risky accesses. We introduce novel incentive mechanism based on Contract Theory. Another benefit of our approach is avoiding accurate estimate of the risk associated with each access. We demonstrate that Nash Equilibria can be achieved in which the user's optimal strategy is performing the risk-mitigation efforts to minimize her organization's risk, and conduct human-subject studies to empirically confirm the theoretical results.

**Keywords:** Insider Threat, Access Control, Risk Management, Incentive Engineering, Human-Subject Experiment.

## 1 Introduction

Access control is used pervasively for security and privacy protection in computer and information systems. Traditionally, access control policies are encoded as rigid rules. One example is the multi-level security (MLS) policy [1] used in the United States government. A user is assigned a security clearance and a set of compartments, a document is assigned a security level and a set of compartments, and access is allowed only if the clearance of the user dominates the level of the document and the user's set of compartments is a superset of the document's. A 2004 report by the JASON defense advisory group [2] discussed many of the shortcomings of such policies, which are often too restrictive to meet modern needs.

**Risk-Based Access Control.** Several proposals for risk-based access control [2,3,4,5] have been proposed to address these limitations. In these approaches, initially each user is assigned certain amount of risk tokens, called risk budget. For each access, the access control system estimates the amount of risk for that access, and the user has to pay in the form of risk tokens in order to gain access. The user's aggregated risk is controlled by the risk budget, and the user is incentivized to choose low-risk accesses because it costs less risk tokens and the surplus of the budget brings her benefits. Such an approach,

however, suffers from several limitations. First, it unintentionally discourages the accesses that involve spending risk tokens, as the user has an incentive to avoid accesses and save her tokens for personal gains. Essentially, this creates a situation in which the user has a conflict of interest between personal rewards and accesses necessary for better completing her job, and as a result, discourages necessary risk-taking behaviors. This goes against the intention of enabling more accesses. Second, this places an additional burden for users, as it creates a new critical resource, risk budgets, the users need to manage for fulfilling their duties. They have to keep track of the amount of risk tokens they have, and be constantly debating whether to save them or spend them. Finally, this approach requires as a first step, a relatively precise quantification of the risk of accesses, which is notoriously difficult.

**Incentive-Based Access Control.** We believe that these problems can be mitigated if the user is willing to use exceptional accesses in the best interest of her organization. In this paper, we propose incentive-based access control (IBAC) as a solution. Our idea is to separate the two objectives of limiting aggregated risks and incentivizing users to adopt risk-mitigation methods to reduce risk, and achieve them through different mechanisms.

## 2 Overview of Incentive-Based Access Control

We had three goals while designing our incentive-based access control. First, it should provide access flexibility to respond to unforeseeable situations by allowing exceptional accesses to bypass controls. Second, it should enable the organization to manage aggregated risks, and identify not only potentially malicious insiders but also those who are simply risk-seeking. Third, users should face consistent incentives to adopt risk-mitigation efforts reducing organizational risk whenever feasible.

### 2.1 Basic Concepts

*Exceptional access* When a user requests an access, the organization first roughly estimates the risk,  $k$ , for that request depend on the users' profile, such as security clearance and roles, where applicable. Based on the quantified estimate, the access request falls into one of the following categories: “*allowed access*” when the risk is low enough; “*denied access*” when the risk is too high; or “*exceptional access*” when the risk is in-between.

*Risk budget* Organization set risk budget for each individual employee. Risk budget enables its owner to obtain exceptional access to override the control when necessary, at the expense of risk points. This provides the flexibility an access model needs to work in a practical environment. On the other hand, risk budget restricts the number of exceptional accesses and prevents exception abuse.

*Risk-mitigation efforts* We assume that for each exceptional access, there is a set  $E$  of risk mitigation methods available for a user to adopt. Different  $e \in E$  may represent

different access method; they may also represent different additional technical and non-technical efforts for mitigating risks. In the rest of this paper, we use the term “access method” and the term “risk-mitigation effort” interchangeably. For ease of discussion, we assume that a special method  $e_0 \in E$  represents the case that user takes an exceptional access without making any risk-mitigation effort, implicitly the highest resulted risk.

*Contract-based rewards* We consider two types of incentive contracts to provide users rewards and align their interests with their organization. A contract of the first type is designed for the situations where the organization can observe and verify the risk-mitigation effort  $e$  taken by the user. The contract specifies a *reward* function  $r(e)$  which determines the reward that the user will get for choosing  $e$ . The second type of contracts is used when the organization is unable to determine the user’s risk-mitigation effort. This happens when the effort cannot be directly observed by the organization, or is too expensive to monitor. Instead, such a contract is specified over a set of observable outcomes  $T$  that indirectly reflect the effort  $e$  the user takes, and therefore called *consequence-based contracts*.

*Personal costs* Given an exceptional access  $a$ , each risk-mitigation effort  $e$  will incur some cost for the user; this is determined by the cost function  $c(e)$  which reflects the expenses incurred in terms of usage of storage/computational resources, opportunities, interference in the normal work flow, and others. We assume that  $c(e_0) = 0$ . In this paper, we measure the costs of risks, personal costs of effort and incentive rewards using the same metric, the *IBAC token*. We use the domain of real numbers  $R$  to denote this domain. Therefore the risk function is denoted as  $k : T \rightarrow R$ ; the cost function is denoted as  $c : E \rightarrow R$ ; and the reward function is denoted as  $r : E \rightarrow R$ .

## 2.2 Putting Things Together

When a user requests an access, IBAC makes access decision to allow any low-risk request and deny any request that has too high a risk. If the risk is in-between, that request is considered as an exceptional access and priced at the risk estimate. User can obtain an exceptional access only if his risk budget is sufficient to cover the expense of risk points for the price. The risk budget deduction is conducted spontaneously without any requirement of human response.

Once an exceptional access is granted, IBAC will launch a contract-based mechanism to incentivize users to perform risk-mitigation efforts. When multiple risk-mitigation efforts exist, a discrepancy of incentives emerges between the organization and the employee: the former prefers the effort that minimizes its risk, while the latter wants to choose what minimizes her cost. This problem is tackled by the incentive contracts.

## 3 Effort-Based Contract

**Contract game.** When the user’s effort is observable, the contract is determined by a reward function  $r : E \rightarrow R$  that specifies the reward the organization offers to the

user based on her observable or verifiable risk-mitigation effort. Under a contract  $r$ , the user attempts to choose the mitigation effort  $e$  that incurs the minimum amount of total personal cost and the maximum amount of the reward, by solving the following optimization problem:

$$\min_e [c(e) - r(e)]$$

On the other hand, the organization seeks a low risk and a low reward associated with an access. Thus, its combined optimization goal is described as follows:

$$\min_r [k(e) + r(e)]$$

Here we call  $c(e) - r(e)$  an *adjusted cost* for the user and  $k(e) + r(e)$  an *adjusted risk* for the organization.

From the above equations, it is evident that none of these parties can unilaterally achieve its optimization objective: the user's selection of  $e$  depends on the organization's choice of  $r$ , and vice versa. This essentially forms a game, which we call *contract game*. Solving this pair of optimization problems gives us one or more *Nash Equilibria*<sup>1</sup> in which neither player (the user or the organization) has an incentive to deviate from its equilibrium strategy (choice of  $e$  or  $r$ ), as this does not give it a better payoff (a lower outcome for its optimization target). Here we denote an equilibrium contract by  $r^*$  and an equilibrium effort by  $e^*$ .

### 3.1 Cooperative User and Deterministic Cost

We assume users are *cooperative* and *rational*. A cooperative rational user will choose among all  $e$ 's such that  $c(e) - r(e)$  is minimum, and when there exists more than one such  $e$ , choose the one that results in the least perceived risk: i.e.,  $k(e)$  is smallest. Again, such a perception can be inaccurate, based upon, for example, some qualitative concepts such as high or low risks.

Let  $E$  be the set of risk-mitigation efforts for an access. The organization constructs its contract  $r(e)$  in the following way: for  $e^* = \arg \min_{e \in E} [k(e) + c(e)]$ , we have  $r(e^*) = c(e^*)$ ; for other  $e \in E$  except  $e_0$ ,  $r(e)$  can be set to any value in  $[0, c(e)]$ . We call this contract *compensation contract*, as it makes up for the user's cost for choosing the effort in the best interest of the organization, i.e.,  $e^*$ . Here we show that this contract is optimal.

**Theorem 1.** *The compensation contract and selection of  $e^*$  form a Nash Equilibrium in the contract game.*

A Nash Equilibrium is a strategy pair in which neither party (the organization or the user) can be better off by switching to another strategy, given that the other party sticks to its equilibrium strategy. This outcome is optimal in the sense that when the users

---

<sup>1</sup> The contract game is actually sequential in which the organization moves first and the user moves next, and the Nash Equilibrium we describe throughout the paper all refers to the sub-game perfect Nash Equilibrium [6].

are cooperative and rational, the equilibrium strategy (i.e, the compensation contract) achieves the lowest adjusted risk  $k(e) + r(e)$  for the organization. We prove this result as follows.

*Proof.* Let us first show that under the compensation contract, a cooperative and rational user will choose  $e^*$ . The adjusted cost of  $e \neq e^*$  is greater than  $c(e_0)$ , which is zero. Comparing it with the adjusted cost of  $e^*$ , we have:  $c(e) - r(e) \geq 0 = c(e^*) - r(e^*)$ . Note that the equality holds only when  $e = e_0$ , which makes  $c(e) = 0$ . When this happens, a cooperative user will still choose  $e^*$ , which incurs a lower risk, according to its definition.

Consider a contract under which a cooperative and rational user chooses  $e'$  and gets a reward  $x$ . This only happens when  $e'$  is no less attractive than  $e_0$ , that is  $c(e') - x \leq c(e_0) = 0$ . Therefore,  $x \geq c(e')$ . For the organization, this choice gives it an adjusted risk at least  $k(e') + c(e')$ , which is no better than what is offered by the compensation contract, according to its definition.

We note that the best option from the organization's point of view minimizes the combined cost of the organization and the user  $k(e) + c(e)$ , rather than simply the organization's risk  $k(e)$ . Hence even from minimizing the organization's adjusted risk, the best option maximizes the combined social welfare.

**Example 1.** Consider a bank that utilizes the effort-based contract, which encourages its tellers to reduce organization's risk when accessing sensitive customer information. For a read access, we assume the access-control mechanism offers the options with or without the copy-paste function. The one with it incurs a risk at 1000 IBAC tokens while the one without incurs a risk at 200 tokens. On the other hand, the operational cost the teller has to undertake in the absence of the function is quantified as 20 tokens. This example is described in table 1.

**Table 1.** Example 1

Effort	$r(e)$	$c(e)$	$k(e)$	$(c - r)$	$(k + r)$
$e_0$	0	0	1000	0	1000
$e_1$	$r$	20	200	$20 - r$	$200 + r$

Organization computes  $k(e_0) + c(e_0)$  to be 1000, and  $k(e_1) + c(e_1)$  to be 220, and determines that  $e_1$  is preferred, and set  $r(e_1) = 20$ , and  $r(e_0) = 0$ . The compensation contract the bank offers to the teller becomes: “*You can receive 20 IBAC tokens as reward if you access the database without copy-paste functionality. Do you want to accept this offer?*”

### 3.2 Considering Non-deterministic Cost

The deterministic model above assumes that  $c(e)$  is a constant across all users and observable to the organization, which has several limitations. First, different users may have different personal costs. Second, under different circumstances, the same person's

cost may vary. Third, a user may not be able to accurately estimate her cost, which leads to the seemingly randomness in effort selection, and alternatively, the organization's estimate of her cost can be inaccurate.

We model all these by using a random cost  $c(e)$ . Specifically, we assume that  $c(e)$  is a random variable with a Gaussian distribution. Such a distribution can be estimated in practice through performing public surveys or sampling. In our model, an individual is viewed as being randomly drawn from a population that produces that cost distribution. Her personal cost, therefore, is also treated as a random draw from the distribution.

The user's objective, again, is to minimize her adjusted cost. The organization, however, cannot observe her specific cost, but the distribution of the cost, and thus intends to minimize the expected adjusted risk, as follows:

$$\sum_e \Pr[e \text{ is chosen}] (k(e) + r(e))$$

Due to the uncertainty of the cost, design of an optimal contract becomes very complicated. For simplicity, here we only consider two risk-mitigation efforts: the low effort  $e_0$  and the high effort  $e_1$ . Since the low-effort  $e_0$  represents no risk mitigation effort, we assume that  $c(e_0)$  is constant at 0, and  $c(e_1)$  follows a Gaussian distribution  $N(\mu, \sigma^2)$ .

Because changing both  $r(e_0)$  and  $r(e_1)$  by the same amount does not affect the user's incentive to choose  $e_1$ , and the organization would like to minimize  $r$ . The effort-based contract should set  $r(e_0) = 0$ , and we need to determine  $r(e_1)$ .

The user chooses  $e_0$  whenever the following inequality holds:

$$c(e_1) - r(e_1) \geq c(e_0) - r(e_0) = 0$$

As  $c(e_1)$  follows a Gaussian distribution  $N(\mu, \sigma^2)$ , this happens with the following probability:

$$p = 1 - \Phi\left(\frac{r - \mu}{\sigma}\right),$$

where  $\Phi$  is the cumulative distribution function (cdf) of the standard Gaussian distribution.

The organization intends to minimize the adjusted cost, and hence would set  $r(e_1)$  as follows:

$$\begin{aligned} r^*(e_1) &= \arg \min_r [p \cdot k(e_0) + (1 - p) \cdot (k(e_1) + r)] \\ &\Rightarrow \arg \min_r \left[ \left(1 - \Phi\left(\frac{r - \mu}{\sigma}\right)\right) (k(e_0) - k(e_1) - r) + r \right] \end{aligned}$$

The above can be solved using standard function minimization techniques. We note that when the risk difference between  $e_0$  and  $e_1$ , i.e.,  $k(e_0) - k(e_1)$ , is large, the reward for taking  $e_1$  should also grow, to reduce the probability  $p$ . However, the value  $r(e_1)$  is not sensitive to inaccuracies of estimating  $k(e)$ .

**Theorem 2.** *The above contract and the user's choice of  $e$  minimizing its adjusted cost forms a Bayesian Nash Equilibrium in the contract game.*

In a Bayesian Nash Equilibrium [6], the strategy of each player (the organization or the user) maximizes its expected payoff against the other's strategy, given its belief about the other player (probability  $p$  of choosing  $e_0$ ). The correctness of the proof directly follows the above analysis.

**Example 2.** In Example 1, consider that an access without the copy-and-paste function incurs a probabilistic cost to the user, which follows a Gaussian distribution with a mean of 20 and a stand deviation of 10. Then, the optimal contract is built as follows:

$$r^*(e_1) = \min_r \left[ \left( 1 - \Phi \left( \frac{r - 20}{10} \right) \right) (1000 - 200 - r) + r \right]$$

where  $\Phi$  is the cdf of a standard normal distribution.

Using the Matlab built-in function of *fminbnd*,  $r^*$  is found to be 46. The effort-based contract for this example is then constructed as: “*You can receive 46 IBAC tokens as reward if you access the database without copy-paste functionality. Do you want to accept this offer?*”

## 4 Consequence-Based Contract

Effort-based contract can be used only when it is possible for the organization to observe the action taken by the user. We now show that even when the organization is unable to observe the actions, the organization can nonetheless compute an optimal effort-based contract and translate that into a consequence-based contract.

We assume that there is a set  $T$  of possible outcomes, and the organization has a damage estimation function  $k : T \rightarrow R$ , that maps each outcome to a quantitative estimate of damage. A consequence-based contract is given by a function  $r : T \rightarrow R$ , which gives the reward the user gets for each of possible outcome. Given such a contract  $r$ , the user will choose an action that minimizes her expected cost. The user's cost for each action  $e \in E$  if the personal cost  $c(e)$ , minus the expect reward the user gets. Let  $T = \{t_1, t_2, \dots, t_n\}$ . Suppose that the probability each  $t_i$  when taken action  $e$  is  $p_i^e$ , then the user's adjusted cost for action  $e$  is given by:

$$uac(e) = c(e) - \sum_{1 \leq i \leq n} p_i(e)r(t_i)$$

The user will choose an  $e$  that minimizes the above cost function. The organization needs to generate a contract such that the user's optimal choice will minimizes its expected cost as well. When a user chooses action  $e$ , the organization's adjusted cost is given by

$$oac(e) = \sum_{1 \leq i \leq n} p_i(e)k(t_i) + \sum_{1 \leq i \leq n} p_i(e)r(t_i)$$

We note that if we use  $k(e)$  to denote the risk to the organization when the user takes action  $e$ , then

$$k(e) = \sum_{1 \leq i \leq n} p_i(e)k(t_i)$$

We further note that for every consequence-based contract, one could construct an equivalent effort-based contract, by computing

$$r'(e) = \sum_{1 \leq i \leq n} p_i(e)r(t_i),$$

and setting

$$r(e) = r'(e) - \min(r'(e)).$$

The goal of the organization is to choose  $r(t_i)$  such that the same  $e$  minimizes both  $uac(e) = c(e) + r(e)$  and  $oac(e) = k(e) - r(e)$ , which is exactly the same as in the effort-based case. The only difference is that in the effort-based case, the organization could directly set  $r(e)$ , but in the consequence-based case, the organization can only *indirectly* set  $r(e)$  by choosing  $r(t_i)$ 's.

Hence if there exists a consequence-based contract  $C$  such that its equivalent effort-based contract is optimal for the effort-based case, then  $C$  is the optimal contract for the consequence-based contract case as well. Suppose that  $C$  is not, and another contract  $C'$  gives better utility for the organization, then its corresponding effort-based contract must also be better, contradicting the assumption that  $C$ 's corresponding effort-based contract is optimal.

Given an optimal effort-based contract  $r$ , one can try to solve for the corresponding consequence-based contract by first solving the following systems of linear equations to compute  $r'(t_i)$ , which may be negative:

$$r(e_j) = \sum_{1 \leq i \leq n} p_i(e_j)r'(t_i).$$

And then set

$$r(t_i) = r'(t_i) - \min(r'(t_i)).$$

**Example 3.** An employee wants to download a file from the Internet. His browser can be adjusted to high security setting or low security setting. We assume that the browser security setting upgrade has a cost of 20 IBAC tokens to the user, and a drive-by-download attack will cause the organization a loss of 1000 tokens. The system cannot detect what security level the user's browser has. Assuming the statistics that a high security setting browser can prevent 90% of drive-by download, and a low security setting browser can only prevent 40% of the same attack, this example is presented in the table 2.

**Table 2.** Third Example

	Consequence $k$	$c + r$
	attacked	unattacked
Low setting	60%	40%
		$60\%r(a) + 40\%r(u)$
High setting	10%	90%
		$20 + 10\%r(a) + 90\%r(u)$
$r(k)$	$r(a)$	$r(u)$

From previous section, we have the optimal effort-based contract is

$$\begin{aligned} r^*(e_0) &= 0, \\ r^*(e_1) &= 20. \end{aligned}$$

We can construct the corresponding consequence-based contract by solving the following linear equations:

$$\begin{aligned} r^*(e_0) &= 0 = 60\%r'(a) + 40\%r'(u), \\ r^*(e_1) &= 20 = 10\%r'(a) + 90\%r'(u). \end{aligned}$$

We further set  $r(t_i) = r'(t_i) - \min(r'(t_i))$ , and provides the employee the following contract: “*You can receive 40 IBAC tokens if there is no virus detected after your file is downloaded. Upgrade your browser setting to high security level will greatly reduce the probability of being infected.*”

## 5 Human-Subject Evaluation

We performed a human-subject experiment to evaluate the efficacy of our access control model. The main goal was to understand how users, who may not be perfectly rational, choose their access choices under our contract-based incentive mechanism. We are also interested in understanding how well our model helps suppress the risk the organization undertakes.

### 5.1 Subject Recruitment

We recruited 36 volunteers. More than 90% of the participants use computers more than 5 hours per day. An interesting finding from their background survey is their different attitudes towards the security protections for their organization’s and their own computing systems: 61% of the participants chose to scan their personal computers immediately upon seeing a virus warning, while only 52% of them did so to their organization’s computers. This echoes our hypothesis about the existing misalignment between employees’ incentives and their organizations’ interests.

### 5.2 Experimental Design

The participants were randomly and equally divided into three groups. Group 1 was treated as the benchmark. Group 2 and 3 worked under IBAC. All groups were given the same tasks of sending ten documents, each of which was attached to a different email. They were told that with a certain probability, these emails could be intercepted by untrusted parties. In order to reduce the risk of information leaks, we suggested, but did not require, that they encrypt the emails or the documents, or both. The participants did not have any obligation to make any of these mitigation efforts. We treated encrypting both email and document as the high effort (Level 3), encrypting only the document as the medium high effort (Level 2), encrypting only the outgoing email as the medium low effort (Level 1), and no encryption as the low effort (Level 0).

**Document Classifications.** These documents were classified into four categories: Secret, Confidential, Restricted, and Unclassified. The risk associated with the document in one category was deemed as one order of magnitude higher than that of a document in its immediate lower category. For example, a secret document was assigned a risk magnitude of 1000, and a confidential one was given 100, and so on. Among these ten documents, two were unclassified, three restricted, two confidential, and three secret.

**Experiment Settings.** In the experiment, Members in Group 2 and 3 need to interact with IBAC model: the budget-based mechanism kept records of their risk points deduction, and the contract-based mechanism incentivized them to select a risk-mitigation effort. Their effort choices determined their reward. In order to make the compensation scheme easy to understand, we combined the budget-based control and the contract-based incentive mechanism together, and set the set of interactions as a scenario of purchasing: we first gave each of them 1,000 points. They had to pay a price using their points for sending a document. The leftover points were later reimbursed as every 300 points for \$5. Therefore, the prices must be carefully designed so that the rewards generated follow an effort-based contract for Group 2, and a consequence-based contract for Group 3.

Before starting the experiment, we gave the participants an exercise of encrypting email and document. We took that chance to measure the time each of them used. Given the time factor in determining compensation scheme, we translate the obtained time measurement into that participant’s personal cost of mitigation effort. For example, we pay Alice \$10 compensation if she completed the experiment in 10 minutes. Thus, if Carol needs 1 minute to perform a document encryption, we set  $c_{\text{encryption}} = \$1 = 60$  points as Carol’s personal cost of encrypting document.

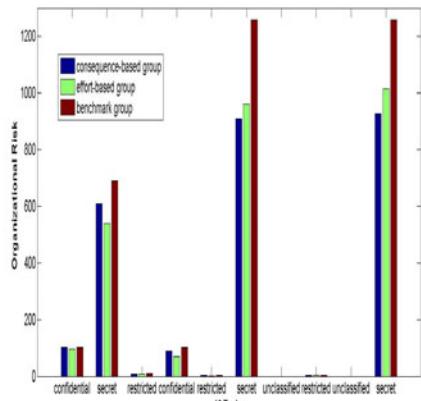
The participants in Group 1 (the benchmark) always received compensation when they completed the tasks. The amount of the compensation was determined solely by the time they spent on the tasks: the less time was used, the higher it became. The participants in the other two groups got compensation, which also depended on time, plus rewards: the more risk points they spent, the less rewards they got.

### 5.3 Reducing Organizational Risks

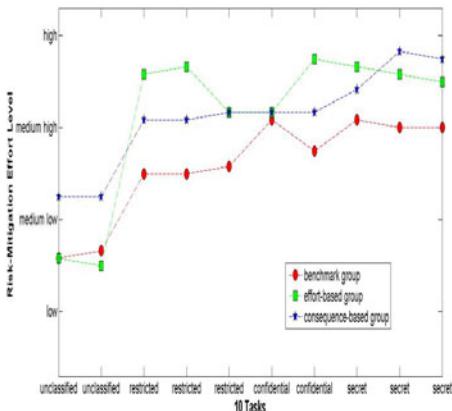
Figure 1(a) shows the average risks incurred by each group. Both incentive contracts helped reduce the organization’s risk exposure. Particularly when secret documents were transmitted, our contracts cut the risk by 30%.

### 5.4 Encouraging Risk-Mitigation Effort

To understand how well our model could incentivize the individuals to perform risk-mitigation efforts, we first sorted the tasks from the lowest rank to the highest one, i.e., unclassified to secret, and then calculated the average risk-mitigation effort level for each task and each group. Figure 1(b) presents the results, in which red dots represent the average levels chosen by the benchmark group for all ten tasks. The green dots describe these by Group 2 (effort-based contract) and the blue dots by Group 3 (consequence-based contract). We can see that both contracts successfully encouraged users to improve their effort levels, which is in stark contrast to those in the benchmark group. We also performed paired t-statistical tests for the effort choices in these three



(a) Organization's Risk Postures



(b) Average Personal Risk Control Effort Levels

groups. Paired with the benchmark group, Group 2 and 3 generated t-values of 3.72 and 7.93 respectively. The outcomes indicate that both groups made statistically significant improvements on risk-mitigation effort, with a 99% confidence level.

## 6 Conclusions

In this paper, we designed a new incentive-based access control mechanism that encourages the users to make necessary accesses, while discouraging them from taking unnecessary risks. This has been achieved through a novel contract-based incentive mechanism that rewards the users for the access that is in the best interest of their organization. We analyzed the IBAC mechanism using game theory and identified the optimal contracts that motivate both the organization and the users to play Nash-Equilibrium strategies. We also performed human-subject experiments that demonstrate the effectiveness of our approach in mitigating the organization's risk.

**Acknowledgements.** This work is supported in part by CNS-0716292.

## References

1. Elliott Bell, D., LaPadula, L.J.: Secure computer systems: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, Mitre Corporation (March 1976)
2. MITRE Corporation. Horizontal integration: Broader access models for realizing information dominace. Technical Report JSR-04-132, JASON Defense Advisory Panel Reports (2004)
3. Molloy, I., Cheng, P., Rohatgi, P.: Trading in risk: Using markets to improve access control. In: New Security Paradigms Workshop, Olympic, California. Applied Computer Security Associates (September 2008)
4. Yemini, A., Dailianas, D., Florissi, Huberman, G.: Marketnet: Market-based protection of information systems. In: The 12th Int. Symp. on Dynamic Games and Applications (2006)
5. Liu, D., Wang, X., Camp, L.J.: Mitigating Inadvertent Insider Threats with Incentives. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 1–16. Springer, Heidelberg (2009)
6. Osborne, M.J., Rubenstein, A.: A Course in Game Theory. The MIT Press, Cambridge (1994)

# Authenticated Key Exchange under Bad Randomness

Guomin Yang<sup>1</sup>, Shanshan Duan<sup>2</sup>, Duncan S. Wong<sup>3</sup>,  
Chik How Tan<sup>1</sup>, and Huaxiong Wang<sup>4,\*</sup>

<sup>1</sup> National University of Singapore  
`{tslyg,tslch}@nus.edu.sg`

<sup>2</sup> University of California San Diego  
`shduan@cs.ucsd.edu`

<sup>3</sup> City University of Hong Kong  
`duncan@cityu.edu.hk`

<sup>4</sup> Nanyang Technological University  
`hxwang@ntu.edu.sg`

**Abstract.** We initiate the formal study on authenticated key exchange (AKE) under bad randomness. This could happen when (1) an adversary compromises the randomness source and hence directly controls the randomness of each AKE session; and (2) the randomness repeats in different AKE sessions due to reset attacks. We construct two formal security models, Reset-1 and Reset-2, to capture these two bad randomness situations respectively, and investigate the security of some widely used AKE protocols in these models by showing that they become insecure when the adversary is able to manipulate the randomness. On the positive side, we propose simple but generic methods to make AKE protocols secure in Reset-1 and Reset-2 models. The methods work in a modular way: first, we strengthen a widely used AKE protocol to achieve Reset-2 security, then we show how to transform any Reset-2 secure AKE protocol to a new one which also satisfies Reset-1 security.

**Keywords:** Authenticated Key Exchange, Resettable Cryptography, Bad Randomness.

## 1 Introduction

An Authenticated Key Exchange (AKE) protocol consists of a tuple of randomized algorithms that enable two parties communicating over an insecure network to establish a common session key. These algorithms consume random coins which are typically generated by pseudo-random number generators (PRNGs). Practical PRNGs, such as ANSI X9.17 PRNG and FIPS 186 PRNG, can generate bit-strings which are computationally indistinguishable from truly random strings provided that the seeds of the PRNGs are fresh and truly random [16].

---

\* The work of H. Wang is supported in part by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

In practice (e.g. OpenSSL), seeds are formed by collecting random data from an entropy pool, which can be created from a hardware source (e.g., sound/video input, disk drives, etc.) and/or a non-hardware source (e.g., the timing and content of events such as mouse movement, keystroke, network traffic, etc.) [19,30].

In some practical situations however, the entropy pool could be controlled by an adversary and the seeds may no longer be fresh or truly random. For example, if an adversary has physical access to a hardware source and/or can control the events of a non-hardware source [19], the adversary may be able to manipulate the data in the entropy pool. Also, in some operating systems such as Linux, random data from the entropy pool may be pre-generated and stored in a buffer for later use. If the buffer is not well-protected, an adversary may be able to modify these pre-generated random data.

Adversarial reset of machines could make an AKE protocol reuse the same random coins in different sessions. It has been known as a real threat to some computing devices such as smart cards [12,6]. Recently, Ristenpart and Yilek [34] showed that the adversarial reset is also a serious security threat to Virtual Machines (VMs). Generally, a system administrator can take snapshots of the current system state of a VM from time to time as regular backups. The VM can later be reverted back to a previous state using the snapshots. To perform an adversarial reset, an adversary can first make a system on a VM crash, e.g., via a Denial of Service (DoS) attack. Then when the system administrator reverts the VM back to a “good state”, some random coins that have occurred before the machine being reset would be reused after the VM is reverted [34].

We categorize the threats above into “Reset-1” and “Reset-2” attacks, respectively. In Reset-1 attack, the adversary controls the random coins used by the AKE algorithms, while in Reset-2 attack, the adversary can reset a device to make algorithms reuse some random coins. The practical interests of these attacks bring a natural question: would the existing AKE protocols, especially those widely used ones, be still secure under these attacks?

In this paper, we conduct the first formal study on AKE under Reset-1 and Reset-2 attacks. Below are the three aspects to which we contribute.

**SECURITY MODELS.** We propose two formal security models for Reset-1 and Reset-2 attacks respectively. We build our models based on the existing Bellare-Rogaway (BR) [7] and Canetti-Krawczyk (CK) [13] security models by providing the adversary additional capabilities. In the Reset-1 model, the adversary directly picks random coins for the AKE participants, while in the Reset-2 model, the adversary does not pick random coins directly but can reset a participant so that the same random coins will be used in multiple AKE sessions. In addition, to capture Kaliski’s online *Unknown Key Share* (UKS) attacks [11], our models allow the adversary to register malicious users with public keys of its own choice.

In the Reset-1 model, since the adversary already controls the randomness, if the adversary also learns the long-lived key of either participant, it can trivially compute the session key. Hence the model cannot allow the adversary to corrupt the long-lived key of either participant, and *Forward Secrecy* (FS) cannot be

captured. On the other side, the Reset-2 model does not have this restriction. This also indicates that the two models are incomparable and explains the reason why we need two models.

**SECURITY OF EXISTING PROTOCOLS.** Based on the two new models we defined, we show that some well-known AKE protocols (e.g. ISO [24,13], SIGMA<sup>1</sup> [14,27], JFK [2], SKEME [28], HMQV [26]), which are proven secure in their original security models will become insecure when the adversary is allowed to manipulate the randomness in the way defined in Reset-1 and/or Reset-2.

**DESIGNING NEW PROTOCOLS.** We then present techniques to build AKE protocols that are secure in both Reset-1 and Reset-2 models. We present a generic way to efficiently transform a Reset-2 secure AKE protocol to a new one which is secure in both Reset-1 and Reset-2 models. Our idea is to generate good “internal” randomness within the protocol, we do this by applying a pseudo-random function (PRF) to the “raw” randomness at the very beginning of a protocol so that after this treatment, computationally “good” randomness are used in the remaining steps of the protocol. And we prove that this simple (but useful) idea indeed works. However, we remark that some additional requirement on the PRF is needed in order to make the transformed protocol still be Reset-2 secure.

Our transformation provides a “modular approach” to the construction of Reset-1 and Reset-2 secure AKE protocols: (1) we first build a Reset-2 secure AKE protocol  $\Pi$ ; (2) then we apply the transformation to  $\Pi$  and get a new protocol  $\Pi'$  which maintains Reset-2 security and in addition to this, it also satisfies Reset-1 security. We illustrate this modular approach by proposing a new SIG-DH based AKE protocol which is transformed from the ISO protocol. The modifications we made are simple and efficient, and can be deployed easily to existing implementations of the protocol.

## 2 Related Work

Authenticated key exchange (AKE) protocols have been extensively studied within the cryptographic community. The first complexity-theoretic treatment of the security notion of AKE is due to Bellare and Rogaway [7]. Their model (referred to as the BR model) and its variants (e.g., [8,9,13]) then became the *de facto* standard for analyzing AKE protocols. In [13], Canetti and Krawczyk combined previous work and presented a new security model (referred to as the CK model). They showed that AKE protocols secure in their model can be composed with symmetric key encryption and authentication functions to provide provably secure communication channels. The CK model was used to demonstrate the security of many popular AKE protocols such as the ISO protocol [24,13], the SIGMA protocol [14,27], the HMQV protocol [26] and many more. Recently, LaMacchia et al. [29] extended the CK model to a new model (referred to as eCK model). In their model, the adversary is allowed to compromise either

---

<sup>1</sup> SIGMA serves as the basis of the signature-based mode of IKE [23] and IKEv2 [25].

the long-live keys or the ephemeral keys (the latter are related to the randomness) of the participants of a protocol session. There have been many discussions on the strength of the two (CK and eCK) models [29,32,10]. In [10], Boyd et al. suggested that these two models are incomparable. In this paper, we follow the definitional approach of Canetti and Krawczyk, and we will see later that in fact our Reset-2 model can be considered as Resettable CK model.

None of the existing security models for AKE considers the bad randomness scenarios that we describe in this paper. Although for AKE protocols resilience to the leakage of ephemeral secret key has been studied by Krawczyk [26] and by LaMacchia et al. [29], their work is different from ours: firstly, ephemeral secret key is related but not equivalent to the randomness required by an AKE protocol, in particular, randomness may be required in other parts of the protocol; secondly, in the case of ephemeral secret key leakage, the adversary can only *passively* learn the ephemeral secret key, but not control its value. Another piece of work that is “somehow” related to ours is the work by Aiello et al. [2] in which the JFK (Just Fast Keying) AKE protocol was proposed and discussions about the reuse of Diffie-Hellman (DH) exponents in multiple JFK AKE sessions were made. However, this is different from the Reset-2 scenario we discussed earlier. The reuse of DH exponent is initiatitively implemented by a participant of the protocol for reducing the number of costly modular exponentiation operations. But in a reset attack, all the components of the protocol use unfresh/used randomness. To see the difference more clearly, if an AKE protocol (such as JFK) uses a randomized digital signature scheme (such as Digital Signature Standard - DSS [1]), then reusing the same randomness to sign different messages may allow an adversary to derive the secret signing key. However, merely reusing the DH exponent may not cause such a serious consequence.

**RESETTABLE CRYPTOGRAPHY.** Resettable security have been considered for other cryptographic protocols before, such as resettable Zero-Knowledge (rZK) proof [12] and resettable Identification (rID) protocols [6]. Recently, Goyal and Sahai [22] studied the problem of resetably secure two-party and multi-party computation for general functionalities, and in [35], Yilek studied resetably secure public key encryption. Although AKE can be considered as a two-party computation function, our work is different from that of Goyal and Sahai [22]. We focus on examining and enhancing the existing AKE protocols that have been widely used in the real practice.

**HEDGED CRYPTOGRAPHY.** Our paper is not the first paper to treat bad randomness for cryptographic operations. The Hedged Cryptography [4,34] preprocesses randomness together with other inputs (messages, keys, etc.) of a cryptographic operation to provide (pseudo)randomness for the cryptographic operation. In particular, in [34], Ristenpart and Yilek presented the hedged RSA key transport and authenticated Diffie-Hellman key exchange protocols used in TLS without formal security models. Their heading technique is different from our treatment to the randomness presented in Sec. 5 and their hedged protocols cannot provide Reset-1 security.

### 3 Security Models and Definitions

#### 3.1 AKE Protocol Descriptions

An Authenticated Key Exchange (AKE) protocol consists of two probabilistic polynomial time algorithms: the Long-Lived Key generation algorithm  $\text{SKG}$  and a protocol execution algorithm  $P$ . In this paper, we focus on the public key setting where the algorithm  $\text{SKG}$  returns a public key and a private key upon each invocation.

**PROTOCOL PARTICIPANTS.** We initialize a nonempty set  $\mathcal{U}$  of parties. Each party  $U \in \mathcal{U}$  is named by a unique string, and that string has some fixed length. We use another set  $\mathcal{MU}$  to denote malicious parties who are added into the system by an adversary after the initialization phase. Each malicious party  $M \in \mathcal{MU}$  is also named by a distinct and fixed-length string which has never been used to name another party inside the system.

**LONG-LIVED KEYS.** Each party  $U \in \mathcal{U}$  holds a public/private key pair  $(\text{pk}_U, \text{sk}_U)$  that is generated according to the Long-Lived Key generation algorithm  $\text{SKG}$ . However, for each party  $M \in \mathcal{MU}$ , its public key  $\text{pk}_M$  can be set to any value except that  $\text{pk}_M$  has never been used as the public key of another party inside the system.

**INSTANCES.** A party may run many instances concurrently. We denote instance  $i$  of party  $U$  by  $\Pi_U^i$ . At the time a new instance is created, a unique instance number within the party is chosen, a sequence of random coins are tossed and feeded to that instance, and the instance enters the “ready” state.

**PROTOCOL EXECUTION.** A protocol execution algorithm is a probabilistic algorithm taking strings to strings. This algorithm determines how instances of the parties behave in response to signals (messages) from their environment. Upon receiving an incoming signal (message)  $M_{\text{in}}$ , an instance runs the protocol  $P$  and generates

$$(M_{\text{out}}, \text{acc}, \text{term}_U^i, \text{sid}_U^i, \text{pid}_U^i, \text{ssk}, St_U^i) \leftarrow P(1^k, U, \text{pk}_U, \text{sk}_U, St_U^i, M_{\text{in}}).$$

The first component  $M_{\text{out}}$  corresponds to the responding message, the second component  $\text{acc}$  denotes the *decision* the instance has made, and the third component  $\text{term}_U^i$  indicates if the protocol execution has been terminated. A session id ( $\text{sid}_U^i$ ), and partner id ( $\text{pid}_U^i$ ) may be generated during the protocol execution. When the decision is *accept*, the instance holds a session key (ssk) which is to be used by upper layer applications. For all the protocols we analyze in this paper, we assume the state information  $St_U^i$  is erased from the memory of  $U$  once  $\text{term}_U^i$  becomes true.

**PARTNERSHIP.** The partnership between two instances is defined via parter ID ( $\text{pid}$ ) and session ID ( $\text{sid}$ ). The  $\text{pid}$  names the party with which the instance believes it has just exchanged a key, and the  $\text{sid}$  is an identifier which uniquely labels the AKE session. We say two instances  $\Pi_U^i$  and  $\Pi_V^j$  are partners if  $\text{pid}_U^i = V, \text{pid}_V^j = U$  and  $\text{sid}_U^i = \text{sid}_V^j$ .

```

procedure Initialize
For all  $U \in \mathcal{U}$ 
 $(\text{pk}_U, \text{sk}_U) \leftarrow \text{SKG}(1^k, U)$ ;  $T_U \leftarrow \emptyset$ 
Timer  $\leftarrow 0$ ;  $b \leftarrow \{0, 1\}$ ;  $\mathcal{MU} \leftarrow \emptyset$ 
return  $\{\text{pk}_U\}_{U \in \mathcal{U}}$ 

procedure Register( $U, \text{pk}$ )
If ( $U \in (\mathcal{U} \cup \mathcal{MU}) \vee \text{pk} \in \{\text{pk}_V\}_{V \in \mathcal{U} \cup \mathcal{MU}}$ ) then
    return Invalid
 $\mathcal{MU} \leftarrow \mathcal{MU} \cup \{U\}$ 
return true

procedure NewInstance( $U, i, N$ )
If ( $U \notin \mathcal{U} \vee i \in T_U$ ) then return Invalid
 $T_U \leftarrow T_U \cup \{i\}$ ;  $N_U^i \leftarrow N$ ;  $St_U^i \leftarrow (N_U^i, \text{ready})$ 
 $\text{acc}_U^i \leftarrow \text{false}$ ;  $\text{term}_U^i \leftarrow \text{false}$ 
 $\text{sid}_U^i \leftarrow \perp$ ;  $\text{pid}_U^i \leftarrow \perp$ ;  $\text{ssk}_U^i \leftarrow \perp$ 
return true

procedure Send( $U, i, M_{\text{in}}$ )
If ( $U \notin \mathcal{U} \vee i \notin T_U \vee \text{term}_U^i$ ) then return Invalid
 $(M_{\text{out}}, \text{acc}, \text{term}_U^i, \text{sid}_U^i, \text{pid}_U^i, \text{ssk}, St_U^i)$ 
 $\leftarrow P(1^k, U, \text{pk}_U, \text{sk}_U, St_U^i, M_{\text{in}})$ 
If ( $\text{acc} \wedge \text{not acc}_U^i$ ) then
     $\text{ssk}_U^i \leftarrow \text{ssk}$ ;  $\text{acc}_U^i \leftarrow \text{true}$ 
return  $(M_{\text{out}}, \text{acc}, \text{term}_U^i, \text{sid}_U^i, \text{pid}_U^i)$ 

procedure Reveal( $U, i$ )
If ( $U \notin \mathcal{U} \vee i \notin T_U$ ) then return Invalid
Timer  $\leftarrow \text{Timer} + 1$ ;  $\text{Time}[\text{Reveal}, (U, i)] \leftarrow \text{Timer}$ 
return  $\text{ssk}_U^i$ 

procedure Corrupt( $U$ )
If  $U \notin \mathcal{U}$  then return Invalid
Timer  $\leftarrow \text{Timer} + 1$ 
 $\text{Time}[\text{Corrupt}, U] \leftarrow \text{Timer}$ 
return  $\text{sk}_U$ 

procedure Test( $U^*, i^*$ )
If  $U^* \notin \mathcal{U}$  then return Invalid
If ( $\text{not acc}_{U^*}^{i^*}$ ) then return Invalid
 $K \leftarrow \text{KeySpace}$ 
If  $b = 0$  then return  $K$ 
Else return  $\text{ssk}_{U^*}^{i^*}$ 

procedure Finalize( $b'$ )
 $V^* \leftarrow \text{pid}_{U^*}^{i^*}$ 
If  $V^* \notin \mathcal{U}$  then return false
If ( $\text{Time}[\text{Corrupt}, U^*]$ 
 $\vee \text{Time}[\text{Corrupt}, V^*]$ )
    return false
If ( $\text{Time}[\text{Reveal}, (U^*, i^*)]$ )
    return false
If ( $(\exists i, i \neq i^* \wedge N_{U^*}^i = N_{U^*}^{i^*})$ )
    return false
If ( $(\exists j^* \in T_{V^*}, \text{pid}_{V^*}^{j^*} = U^*$ 
 $\wedge \text{sid}_{V^*}^{j^*} = \text{sid}_{U^*}^{i^*})$ )
    If ( $(\exists j, j \neq j^* \wedge N_{V^*}^j = N_{V^*}^{j^*})$ )
        return false
    If ( $\text{Time}[\text{Reveal}, (V^*, j^*)]$ )
        return false
return  $(b = b')$ 

```

**Fig. 1.** Game RAKE-1

### 3.2 Security Models

We define two security models to capture the two scenarios (namely, Reset-1 and Reset-2) where the randomness of an AKE protocol goes bad. However, we assume that the long-lived keys of all the honest party in the set  $\mathcal{U}$  are securely generated using fresh random coins.

**RESET-1 MODEL.** In this model, we consider the scenario where the randomness of each instance is completely controlled by the adversary. The formal definition is given in Figure 1 where in total six types of oracle queries are defined to capture the adversarial capabilities. In the following we explain those oracle queries in detail.

**Register**( $U, \text{pk}_U$ ) This oracle query allows the adversary  $A$  to register a new user  $U$  with public key  $\text{pk}_U$ . Here we only require that neither the user identity  $U$  nor the public key  $\text{pk}_U$  exists in the system. In particular, we do not require the adversary to provide a proof of knowledge on the secret key with regard to  $\text{pk}_U$ .

**NewInstance**( $U, i, N$ ) This oracle query allows  $A$  to initialize a new instance  $\Pi_U^i$  within party  $U$  with a binary string  $N$  which serves as the random tape of  $\Pi_U^i$ .

**Send**( $U, i, M_{\text{in}}$ ) This oracle query invokes instance  $i$  of  $U$  with message  $M_{\text{in}}$ . The instance then runs  $P(1^k, U, \text{pk}_U, \text{sk}_U, St_U^i, M_{\text{in}})$  and sends the response back to the adversary. Should  $\Pi_U^i$  terminate or accept will be made available to  $A$ . The session id  $\text{sid}_U^i$  and partner id  $\text{pid}_U^i$  are also made available to  $A$  once they are available.

**Reveal**( $U, i$ ) If oracle  $\Pi_U^i$  has accepted and generated a session key  $\text{ssk}_U^i$ , then  $\text{ssk}_U^i$  is returned to the adversary.

**Corrupt**( $U$ ) By making this oracle query, adversary  $A$  obtains the long-lived secret key  $\text{sk}_U$  of party  $U$ .

**Test**( $U^*, i^*$ ) By making this oracle query,  $A$  selects a challenge instance  $\Pi_{U^*}^{i^*}$ . If  $\Pi_{U^*}^{i^*}$  has accepted, holding a session key  $\text{ssk}_{U^*}^{i^*}$ , then the following happens. If the coin  $b$ , flipped in the **Initialize** phase, is 1, then  $\text{ssk}_{U^*}^{i^*}$  is returned to the adversary. If  $b = 0$ , then a random session key is drawn from the session key space and returned to the adversary. This query is only asked once during the whole game.

The success of an adversary is measured by its ability to distinguish a real session key from a random key in the session key space. However, some oracle queries will render session keys exposed. By issuing these queries the adversary can trivially win the game. To exclude these trivial attacks, we consider the adversary to be successful only if it specifies a *fresh* oracle in the **Test** query.

First of all, the adversary can trivially derive a session key if one of the parties involved in that session is the adversary itself (i.e. one party is created by the adversary via a **Register** query).

The adversary will learn a party's long lived key by making a **Corrupt** query. Since in the Reset-1 model, randomness is completely controlled by the adversary, once a party is corrupted, the adversary is able to derive all state information and session keys ever generated by the party. So there is no security guarantee on session keys of any corrupted party. In other words, we don't consider the notion of *forward secrecy* in the Reset-1 model.

The adversary can certainly learn the value of a session key via a **Reveal** query. In the reset setting, the adversary can also derive a session key by mounting the *reset-and-reply* attack. Specifically, the adversary first activates a protocol execution between instance  $\Pi_U^i$  with random tape  $N_U$ , and instance  $\Pi_V^j$  with random tape  $N_V$ . Then it activates another instance  $\Pi_U^{i'}$  with the same random tape  $N_U$ . By replaying messages from  $\Pi_V^j$ , the adversary makes  $\text{ssk}_U^{i'} = \text{ssk}_U^i$ . In this case, revealing  $\text{ssk}_U^i$  (or using it in a upper layer application) will automatically render  $\text{ssk}_U^{i'}$  insecure, and vice versa. This type of attacks imply that as long

<b>procedure Initialize</b>	<b>procedure Test(<math>U^*, i^*</math>)</b>
The same as in Game RAKE-1	The same as in Game RAKE-1
<b>procedure Register(<math>U, \text{pk}</math>)</b>	<b>procedure Finalize(<math>b'</math>)</b>
The same as in Game RAKE-1	$V^* \leftarrow \text{pid}_{U^*}^{i^*}$
<b>procedure NewInstance(<math>U, i, j</math>)</b>	If $V^* \notin \mathcal{U}$ then return false
If ( $U \notin \mathcal{U} \vee i \in T_U$ ) then return Invalid	If ( $\exists i, i \neq i^* \wedge R_{U^*}^i = R_{U^*}^{i^*}$ )
If $j \neq \perp \wedge j \notin T_U$ then return Invalid	return false
If $j = \perp$ then $R_U^i \leftarrow \text{RandomCoins}$	If (Time[Reveal, $(U^*, i^*)$ ])
Else $R_U^i \leftarrow R_U^j$	return false
$T_U \leftarrow T_U \cup \{i\}; St_U^i \leftarrow (R_U^i, \text{ready})$	If ( $\exists j^* \in T_{V^*}, \text{pid}_{V^*}^{j^*} = U^*$
$\text{acc}_U^i \leftarrow \text{false}; \text{term}_U^i \leftarrow \text{false}$	$\wedge \text{sid}_{V^*}^{j^*} = \text{sid}_{U^*}^{i^*}$ )
$\text{sid}_U^i \leftarrow \perp; \text{pid}_U^i \leftarrow \perp; \text{ssk}_U^i \leftarrow \perp$	If ( $\exists j, j \neq j^* \wedge R_{V^*}^j = R_{V^*}^{j^*}$ )
return true	return false
<b>procedure Send(<math>U, i, M_{\text{in}}</math>)</b>	If (Time[Reveal, $(V^*, j^*)$ ])
The same as in Game RAKE-1	return false
<b>procedure Reveal(<math>U, i</math>)</b>	Else
The same as in Game RAKE-1	If (Time[Corrupt, $V^*$ ])
<b>procedure Corrupt(<math>U</math>)</b>	return false
The same as in Game RAKE-1	return $(b = b')$

**Fig. 2.** Game RAKE-2

as the random tape of one instance  $\Pi_U^i$  is used by another instance  $\Pi_U^{i'}$ , there is no security guarantee on the session keys generated by these two instances. So when defining the freshness of an instance, we require that its random tape is never used by another instance. Our goal is to design AKE protocols such that reset attacks would not affect the security of session keys generated by those un-reset instances.

**Definition 1.** Let  $\mathcal{AKE}$  be an AKE protocol. Let  $A$  be a Reset-1 adversary against  $\mathcal{AKE}$  and  $k$  a security parameter. The advantage of  $A$  is defined as

$$\mathbf{Adv}_{\mathcal{AKE}, A}^{\text{rake-1}}(k) = \Pr[\text{RAKE-1}_{\mathcal{AKE}, A}(k) \Rightarrow \text{true}] - 1/2.$$

We say  $\mathcal{AKE}$  is secure in the Reset-1 model if

1. in the presence of a benign adversary who faithfully conveys messages, then two partnering instances output the same session key; and
2. for any PPT adversary  $A$ ,  $\mathbf{Adv}_{\mathcal{AKE}, A}^{\text{rake-1}}(k)$  is negligible.

**RESET-2 MODEL.** In this model, we consider the scenario where the adversary is able to perform reset attacks, but unable to directly set the value of the random

coins. The game RAKE-2, described in Figure 2, is used to define the security of AKE protocols in the Reset-2 setting. The definitions of oracle queries **Register**, **Send**, **Reveal**, **Corrupt** and **Test** are the same as those in game RAKE-1. But differently, when initializing a new user instance  $\Pi_U^i$  via a **NewInstance** query, the adversary does not set the random coins directly. Instead, it can specify another instance  $\Pi_U^j$  that has already been initialized, and instance  $\Pi_U^i$  would use the same random coins that  $\Pi_U^j$  has used. The adversary can also let  $\Pi_U^i$  use fresh random coins by setting  $j = \perp$ .

This adversarial model enables us to define *forward secrecy*. Recall that forward secrecy requires that compromising two users' long-lived secret keys should not allow the adversary to compromise any already established session key. We say an instance  $\Pi_U^i$  ( $U \in \mathcal{U}$ ) is fs-unfresh in the Reset-2 model if any of the following conditions is true:

1.  $\text{pid}_U^i$  is created by the adversary via a **Register** query.
2.  $A$  reveals the session key of  $\Pi_U^i$ .
3. There exists another instance of  $U$  whose random tape is the same as that of  $\Pi_U^i$  (i.e. a reset attack against  $\Pi_U^i$  has occurred).
4. Condition 2 is true regarding the partner-oracle of  $\Pi_U^i$  (if it exists).
5. Condition 3 is true regarding the partner-oracle of  $\Pi_U^i$  (if it exists).
6.  $\Pi_U^i$  has no partner instance, and  $A$  corrupts  $\text{pid}_U^i$ .

Otherwise, we say  $\Pi_U^i$  is fs-fresh.

**Definition 2.** Let  $\mathcal{AKE}$  be an AKE protocol. Let  $A$  be a Reset-2 adversary against  $\mathcal{AKE}$  and  $k$  a security parameter. The advantage of  $A$  is defined as

$$\mathbf{Adv}_{\mathcal{AKE},A}^{\text{rake-2}}(k) = \Pr[\text{RAKE-2}_{\mathcal{AKE},A}(k) \Rightarrow \text{true}] - 1/2.$$

We say  $\mathcal{AKE}$  is secure in the Reset-2 model if

1. in the presence of a benign adversary who faithfully conveys messages, then two partnering instances output the same session key; and
2. for any PPT adversary  $A$ ,  $\mathbf{Adv}_{\mathcal{AKE},A}^{\text{rake-2}}(k)$  is negligible.

**Strong Corruption.** So far we only consider the so called “weak corruption model”. To define strong corruption in our models, we follow the approach of Canetti and Krawczyk [13] and introduce a new query called **RevealState** query.

```

procedure RevealState( $U, i$ )
  If ( $U \notin \mathcal{U} \vee i \notin T_U$ ) then return Invalid
  Timer  $\leftarrow$  Timer + 1 ; Time[RevealState,  $(U, i)$ ]  $\leftarrow$  Timer
  return  $S_U^i$ 
```

Now an additional restriction to the adversary  $A$  is that  $A$  cannot ask the **RevealState** query to the instance  $\Pi_{U^*}^{i^*}$  or its partner  $\Pi_{V^*}^{j^*}$  (if the latter exists).

```

procedure Finalize( $b'$ )
...
If (Time[Reveal,  $(U^*, i^*)$ ) ∨ Time[RevealState,  $(U^*, i^*)$ )
    return false
...
If (Time[Reveal,  $(V^*, j^*)$ ) ∨ Time[RevealState,  $(V^*, j^*)$ )
    return false
...

```

It is also worth noting that by adding the **RevealState** query, our Reset-2 model can be considered as Resettable CK model.

## 4 Resettable Security of Existing AKE Protocols

It is obvious to see that many widely deployed AKE protocols such as ISO [24], SIGMA [27,14], JFK [2] and SKEME [28] are insecure in the Reset-1 model since for these protocols, the secrecy of the session key solely relies on the secrecy of the ephemeral secrets. Some of these protocols are insecure in the Reset-2 model either. As we have briefly mentioned before, for SIG-DH protocols, if they are implemented using DSS (or any signature scheme under the Fiat-Shamir paradigm [20]), then they are insecure in either of our reset models as the adversary can retrieve the long-lived signing key of an honest user by letting the user sign two different messages using the same randomness.

The HMQV protocol, proposed by Krawczyk in [26], is currently one of the most prominent AKE protocols. Besides achieving proven security and high efficiency, the HMQV protocol has several extra features, such as resilience to leakage of the DH exponents. However, according to a recent result by Menezes and Ustaoglu [31], an adversary can derive the long-lived secret key of an honest user if the adversary can make the user use the same randomness in different sessions, which indicates HMQV is insecure in either of our reset models.

## 5 From Reset-2 Security to Reset-1 and Reset-2 Security

In this section, we show that though the Reset-1 and Reset-2 models are incomparable, we can do a simple transformation on a Reset-2 secure AKE protocol to derive a new protocol that is secure in both Reset-1 and Reset-2 models.

**The Transformation.** Given a protocol  $\Pi = (\text{SKG}, \text{P})$  that is secure in the Reset-2 model, and a pseudo-random function family  $\mathbb{F} = \{\mathsf{F}_K : \{0, 1\}^{\rho(k)} \rightarrow \{0, 1\}^{\ell(k)} | K \in \{0, 1\}^{\delta(k)}\}$  where  $\rho(k)$ ,  $\ell(k)$  and  $\delta(k)$  are all polynomials of  $k$ , and  $\ell(k)$  denotes the maximum number of random bits needed by a party in an execution of  $\text{P}$ , we construct a new protocol  $\Pi' = (\text{SKG}', \text{P}')$  as follows:

- $\text{SKG}'(1^k)$ : run  $\text{SKG}(1^k)$  to generate  $(\text{pk}, \text{sk})$ , select  $K \leftarrow_s \{0, 1\}^{\delta(k)}$ . Set  $\text{pk}' = \text{pk}$  and  $\text{sk}' = (\text{sk}, K)$ .
- $\text{P}'$ : get a  $\rho(k)$ -bit random string  $r$ , then compute  $r' \leftarrow \mathsf{F}_K(r)$  and run  $\text{P}$  with random coins  $r'$ .

**Theorem 1.** *If  $\Pi$  is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model, and  $\mathbb{F}$  is a secure pseudo-random function family, then  $\Pi'$  is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-1 model.*

The detailed proof is deferred to the full paper.

The pseudo-random function (PRF) family  $\mathbb{F}$  is the central tool for our transformation. However, the security of a pseudo-random function  $F_K(\cdot)$  relies on the secrecy of the key  $K$ . When the key is known to the adversary, then we cannot assume the output of the function is still computationally indistinguishable from truly random strings. So a problem arises regarding our transformation: the resulting protocol “seems” no longer secure in the Reset-2 model. Recall in the Reset-2 model, the adversary is allowed to corrupt the long-lived key  $sk'_{U^*} = (sk_{U^*}, K_{U^*})$  of the user  $U^*$  that output by the adversary in the Test query, then even given a truly random string  $r$ , we cannot guarantee  $F_{K_{U^*}}(r)$  is random from the viewpoint of the adversary who knows  $K_{U^*}$ .

Fortunately, this problem can be resolved, but we need an extra requirement on  $\mathbb{F}$ , that is, we require  $\mathbb{F}$  to be a Strong Randomness Extractor (SRE) [17]. In [15], Chevassut et al. showed that those very strong (i.e. the adversary has very small winning advantage) pseudo-random function families are also good strong randomness extractors. For real implementation, the HMAC function [5], which is widely used in the real practice (e.g., TLS and IKE), is a good candidate for our purpose [3,18,33].

**Theorem 2.** *If  $\Pi$  is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model, and  $\mathbb{F}$  is a pseudo-random function family and a strong randomness extractor, then  $\Pi'$  is secure in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model.*

The proof is deferred to the full paper.

## 6 A New SIG-DH Protocol

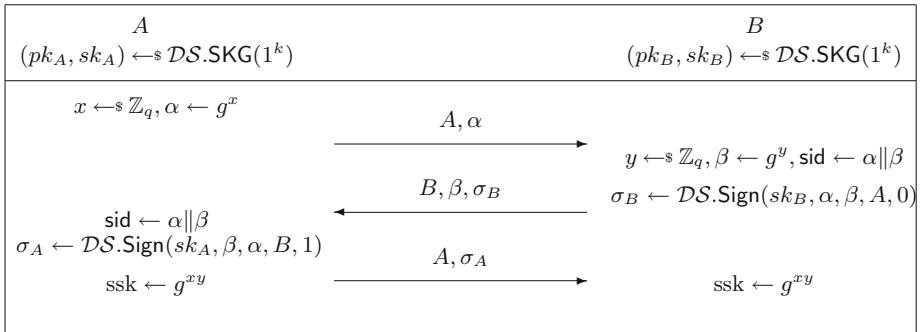
In this section, we modify the ISO protocol [24,13] to obtain a new SIG-DH protocol that is secure in both Reset-1 and Reset-2 models.

We first construct a variant of the ISO protocol, denoted by ISO-R2 (Fig. 3), that is secure in the Reset-2 model. The protocol uses a digital signature scheme  $\mathcal{DS} = (\mathcal{DS}.SKG, \mathcal{DS}.Sign, \mathcal{DS}.Vf)$  that is deterministic (i.e., the signing algorithm  $\mathcal{DS}.Sign$  is deterministic) and existentially unforgeable under adaptive chosen-message attack (uf-cma) [21].

**Theorem 3.** *The ISO-R2 protocol is secure in the Strong Corruption Reset-2 model if  $\mathcal{DS}$  is a uf-cma secure deterministic digital signature scheme, and the DDH assumption holds in the underlying group.*

The proof is deferred to the full paper.

Given the ISO-R2 protocol, we can then apply the transformation in Sec. 5 to obtain a new protocol that is secure in both Reset-1 and Reset-2 models. We omit the transformed protocol here.

**Fig. 3.** The ISO-R2 Protocol

**Acknowledgements.** We thank Mihir Bellare for his suggestions on the modeling part of this paper. We also thank the reviewers for their comments and suggestions.

## References

1. Digital signature standard. National Institute of Standards and Technology, NIST FIPS PUB 186 (May 1994)
2. Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A.D., Reingold, O.: Just fast keying: Key agreement in a hostile Internet. ACM Trans. Inf. Syst. Secur. 7(2), 242–273 (2004)
3. Bellare, M.: New Proofs for NMAC and HMAC: Security without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
4. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged Public-Key Encryption: How to Protect Against Bad Randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
5. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
6. Bellare, M., Fischlin, M., Goldwasser, S., Micali, S.: Identification Protocols Secure Against Reset Attacks. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 495–511. Springer, Heidelberg (2001)
7. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
8. Bellare, M., Rogaway, P.: Provably secure session key distribution — the three party case. In: 28th ACM STOC, pp. 57–66
9. Blake-Wilson, S., Menezes, A.: Entity authentication and authenticated key transport protocols employing asymmetric techniques. In: Security Protocols Workshop, pp. 137–158 (1997)

10. Boyd, C., Cliff, Y., Gonzalez Nieto, J.M., Paterson, K.G.: Efficient One-Round Key Exchange in the Standard Model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 69–83. Springer, Heidelberg (2008), <http://eprint.iacr.org/2008/007>
11. Kaliski Jr., B.S.: An unknown key-share attack on the MQV key agreement protocol. ACM Trans. Inf. Syst. Secur. 4(3), 275–288 (2001)
12. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge. In: 32nd ACM STOC, pp. 235–244
13. 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), <http://eprint.iacr.org/2001/040/>
14. Canetti, R., Krawczyk, H.: Security Analysis of iKE’s Signature-Based Key-Exchange Protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002), <http://eprint.iacr.org/2002/120/>
15. Chevassut, O., Fouque, P.-A., Gaudry, P., Pointcheval, D.: Key derivation and randomness extraction. Cryptology ePrint Archive, Report 2005/061 (2005), <http://eprint.iacr.org/>
16. Desai, A., Hevia, A., Yin, Y.L.: A Practice-Oriented Treatment of Pseudorandom Number Generators. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 368–383. Springer, Heidelberg (2002)
17. Dodis, Y.: Exposure-resilient cryptography. PhD Thesis, MIT (2000)
18. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004)
19. Eastlake, D., Crocker, S., Schiller, J.: IETF RFC 1750: Randomness Recommendations for Security (1994)
20. 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)
21. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
22. Goyal, V., Sahai, A.: Resettable Secure Computation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 54–71. Springer, Heidelberg (2009)
23. Harkins, D., Carrel, D.: The Internet Key Exchange (IKE). RFC 2409 (1998)
24. Entity authentication mechanisms - Part 3: Entity authentication using asymmetric techniques. ISO/IEC IS 9798-3 (1993)
25. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol. RFC 4306 (2005)
26. 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)
27. Krawczyk, H.: SIGMA: The ‘SIGn-and-mAc’ Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003)
28. Krawczyk, H.: SKEME: A versatile secure key exchange mechanism for Internet. In: NDSS, pp. 114–127 (1996)
29. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Provable Security, pp. 1–16 (2007)
30. Matthews, T.: Suggestions for random number generation in software. RSA Laboratories Bulletin # 1 (January 1996)

31. Menezes, A., Ustaoglu, B.: On reusing ephemeral keys in Diffie-Hellman key agreement protocols. International Journal of Applied Cryptography (to appear), <http://www.math.uwaterloo.ca/~ajmeneze/research.html>
32. Okamoto, T.: Authenticated Key Exchange and Key Encapsulation in the Standard Model. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474–484. Springer, Heidelberg (2007), <http://eprint.iacr.org/2007/473>
33. Fouque, P.-A., Pointcheval, D., Zimmer, S.: HMAC is a randomness extractor and applications to TLS. In: ASIACCS, pp. 21–32 (2008)
34. Ristenpart, T., Yilek, S.: When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In: Network and Distributed System Security Symposium (NDSS) (2010)
35. Yilek, S.: Resettable public-key encryption: How to encrypt on a virtual machine. In: Topics in Cryptology - CT-RSA, pp. 41–56 (2010)

# Oblivious Outsourced Storage with Delegation

Martin Franz<sup>1</sup>, Peter Williams<sup>2</sup>, Bogdan Carbunar<sup>3</sup>, Stefan Katzenbeisser<sup>1,4</sup>,  
Andreas Peter<sup>4</sup>, Radu Sion<sup>2</sup>, and Miroslava Sotakova<sup>2</sup>

<sup>1</sup> Center for Advanced Security Research Darmstadt - CASED

<sup>2</sup> Computer Science, Stony Brook University

<sup>3</sup> Applied Research Center, Motorola Labs

<sup>4</sup> Technische Universität Darmstadt

**Abstract.** In the past few years, outsourcing private data to untrusted servers has become an important challenge. This raises severe questions concerning the security and privacy of the data on the external storage. In this paper we consider a scenario where multiple clients want to share data on a server, while hiding all access patterns. We propose here a first solution to this problem based on Oblivious RAM (ORAM) techniques. Data owners can delegate rights to external new clients enabling them to privately access portions of the outsourced data served by a curious server. Our solution is as efficient as the underlying ORAM constructs and allows for delegated read or write access while ensuring strong guarantees for the privacy of the outsourced data. The server does not learn anything about client access patterns while clients do not learn anything more than what their delegated rights permit.

## 1 Introduction

As data management is increasingly being outsourced to third party “cloud” providers such as Google, Amazon or Microsoft, enabling secure, distributed access to outsourced data becomes essential. This raises new requirements concerning the privacy of the outsourced data with respect to the external storage, network traffic observers or even collaborators who might have access to parts of the outsourced database. In this scenario, a data owner  $O$  outsources his data items to a server  $S$ . At a later time, he wishes to delegate read- or write access to individual data items to third party clients  $C_1, \dots, C_n$ . Since the data is potentially privacy sensitive, strong confidentiality and privacy guarantees should be in place. Clients should only be able to access those items they are given access to. Moreover, potential adversaries should be unable to derive information from the observed access patterns to the outsourced database. This is necessary, as even the observation that one item is accessed more frequently than others or the fact that one item is accessed by multiple clients, might leak sensitive information about this particular item.

In the special case where the owner is the sole client accessing the data stored on the server, the problem can be solved by applying techniques from Oblivious RAMs [10,21,18]. An ORAM structure preserves not only data confidentiality

but also provides privacy for client data accesses. So far, the problem of hiding access patterns in outsourcing database scenarios containing multiple (distrusted) clients is open. In this paper we show that ORAM techniques can be adapted to this scenario as well. To this end, we introduce a new ORAM feature: delegated access. Data owners can delegate controlled access to their outsourced database to third parties, while preserving full access privacy and data confidentiality. Achieving this turns out to be non-trivial: in addition to preserving the owner's access privacy, we also need to ensure that (i) the server is unable to learn the access patterns of any of the clients, (ii) no client is able to learn or modify any information of items she cannot access and (iii) no client can learn the access patterns of items which she cannot access herself.

### 1.1 Applications

We now describe several applications that can be built on the constructions we propose in this paper.

**Anonymous Banking.** The numbered accounts supported by several banks claim to provide user privacy. However, by allowing banks to trace the currency flow and build access pattern statistics, they can be used to learn undesired information, ultimately compromising privacy. The solution proposed in this paper can be used toward preventing such leaks: Account numbers and details are stored as records by the bank and account owners can delegate access rights to other clients as desired. Since the data is accessed obliviously, the bank can learn neither which records are being accessed nor the access rights associated with users.

**Oblivious Document Sharing.** Document sharing applications such as Google Docs suffer from obvious security and privacy shortcomings. Not only is the central storage able to access the cleartext documents but it can also learn access privileges as well as access patterns and exact contributions from individual users. Our solution is the perfect fit for this problem: Users can store encrypted documents, privately outsource read and write privileges and obliviously and efficiently access desired documents as allowed by their permissions.

**Rating Agency Access.** Privacy is of paramount importance in financial markets. Public knowledge of investor interest can influence the ratings and prices of company shares in undesired ways. The natural question to ask is, can an investor privately obtain desired information about companies of interest? The solution we provide in this paper answers this question affirmatively. A rating agency maintains an ORAM with records containing ratings and general information for individual companies. Each company owns its own records and can delegate write access to specialized rating assessing companies and, at the same time, an on-demand read-only access to clients that pay to privately access them.

## 1.2 Contributions

We devise delegated ORAM privacy and security properties, expressing the fact that clients cannot learn any information about items which they are not allowed to access. We provide a first construction of an ORAM with delegation that satisfies this property while preserving the original ORAM privacy properties. The construction relies on a new type of read, write and insert capabilities issued by data owners for items that clients should be able to access. We also show how data owners can efficiently revoke access rights.

## 2 Building Blocks and Related Work

In this section we describe cryptographic primitives used in our solution, the concept of ORAMs and other related work.

### 2.1 Cryptographic Primitives

In the solution presented in this paper, we make use of the following primitives:

*Symmetric encryption.* We will use a symmetric encryption scheme to encrypt items which are stored in the database. In particular, we employ an IND-CPA secure, anonymous, verifiable, symmetric encryption scheme  $(E, D, K)$  where  $E$  and  $D$  are the encryption and decryption algorithms and  $K$  is the key generation algorithm which outputs the secret key [14].

*Signatures.* We also use an existentially unforgeable identity-based signature scheme which consists of four algorithms  $(\mathbb{G}, \mathbb{K}, \mathbb{S}, \mathbb{V})$ .  $\mathbb{G}$  outputs public operating parameters as well as a keypair containing a master public key  $\mathcal{M}_P$  and a master secret key  $\mathcal{M}_S$ ;  $\mathbb{K}_{\mathcal{M}_S}(id)$  outputs a private signing key  $s_{id}$  for an identity  $id$ ;  $\mathbb{S}(s_{id}, message)$  and  $\mathbb{V}_{\mathcal{M}_P}(signature, id, message)$  are the signature generation and verification algorithms. A concrete instantiation of an identity-based signature scheme can be found in [17].

### 2.2 Oblivious RAM

Oblivious RAM [10] provides access pattern privacy to clients (or software processes) accessing a remote database (or RAM). The database is considered to be a set of  $n$  encrypted pairs of the form  $(id, value)$ , denoting an item *value* stored under a searchable tag *id*, and supports read and write operations. Client access privacy is obtained by maintaining two invariants: (i) never reveal the *id* values of interest in a query and (ii) never look twice in the same spot for the same *id*. Since we base our work on the “square root” solution [10], we briefly recall it here:

**The “square root” solution.** In addition to the  $n$  locations reserved for items of the database, the server maintains  $2\sqrt{n}$  additional memory locations.  $\sqrt{n}$  of them store dummy items (used to preserve access privacy as discussed below).

The remaining  $\sqrt{n}$  locations serve as a “cache” buffer. To hide the virtual access pattern, the client first obliviously shuffles the database items together with the  $\sqrt{n}$  dummy items, using a permutation chosen uniformly at random. The suggested way to do that is the following: We assign all  $m := n + \sqrt{n}$  items a tag, chosen at random from a space of size  $m^2/\epsilon$ , yielding a collision probability of  $\epsilon$ . Then the client sorts the items according to their tags obliviously, using a universal sorting network (such as a Batcher network). Once the database is shuffled,  $\sqrt{n}$  database accesses are possible by the client before another reshuffle has to take place. To access an item  $id$ , the client first reads the *entire* buffer. If  $id$  is not found there, the client retrieves it from the database by performing a binary search for the element indexed by the random tag which was associated to  $id$  upon the last reshuffle over all  $n + \sqrt{n}$  real- and dummy items stored on the server. Notice that the location at which the item has been found does not need to be kept hidden. This is because from the perspective of the server, any database location can potentially store any item. If, on the other hand,  $id$  is found in the cache buffer, the client retrieves a previously unread *dummy* item. This is necessary to hide from the server whether the desired item  $id$  was found in the buffer, and thus hide access patterns and inter-query correlation. Finally, the client places the retrieved and re-encrypted item in the cache buffer. When the buffer becomes full, the client obliviously reorganizes the items in the database together with the ones in the cache buffer (while also generating new dummy items), and the process is ready to repeat.

Clearly, from the server’s point of view, the database locations are accessed in a random order and each of them at most once. Per each access, the procedure achieves an (amortized) overhead of  $O(\sqrt{n} \log^2 n)$ . As discussed in [10], the result can be optimized to achieve an  $O(\sqrt{n} \log n)$  computational overhead.

### 2.3 Related Work

**Private Information Retrieval.** Another set of existing mechanisms handle access pattern privacy (but *not data confidentiality*) in the presence of *multiple clients*. Private Information Retrieval (PIR) [5] protocols aim to allow (arbitrary, multiple) clients to retrieve information from public or private databases, without revealing to the database servers which records are retrieved.

In initial results, Chor et al. [5] proved that in an information theoretic setting, any single-server solution requires  $\Omega(n)$  bits of communication. PIR schemes with only sub-linear communication overheads, such as [5], require multiple non-communicating servers to hold replicated copies of the data. When the information theoretic guarantee is relaxed single-server solutions with better complexities exist; an excellent survey of PIR can be found online [8,9].

Recently, Sion and Carbunar [19] showed that due to computation costs, use of existing non-trivial single-server PIR protocols on current hardware is still orders of magnitude more time-consuming than trivially transferring the entire database. Their deployment would in fact *increase* overall execution time, as well as the probability of *forward* leakage, when the present trapdoors become eventually vulnerable – e.g., today’s queries will be revealed once factoring of

today's values will become possible in the future. Their result goes beyond existing knowledge of mere "impracticality" under unfavorable assumptions. On real hardware, *no* existing non-trivial single server PIR protocol could have possibly had out-performed the trivial client-to-server transfer of records in the past, and is likely not to do so in the future either. This negative result is due to the fact that on any known past general-purpose Von Neumann hardware, it is simply more expensive to PIR-process one bit of information than to transfer it over a network.

**Hardware-aided PIR.** The recent advent of tamper-resistant, general-purpose trustworthy hardware such as the IBM 4764 Secure Co-Processor [12] has opened the door to efficiently deploying ORAM privacy primitives for PIR purposes (i.e., for arbitrary public or private data, not necessarily originated by the current client) by deploying such hardware as a trusted server-side client proxy.

Trusted hardware devices however are not a panacea. Their practical limitations pose a set of significant challenges in achieving sound regulatory-compliance assurances. Specifically, heat dissipation concerns under tamper-resistant requirements limit the maximum allowable spatial gate-density. As a result, general-purpose secure coprocessors are significantly constrained in both computation ability and memory capacity, being up to one order of magnitude slower than host CPUs.

Asonov was the first to introduce [1] a PIR scheme that uses a secure CPU to provide (an apparent)  $O(1)$  online communication cost between the client and server. However, this requires the secure CPU on the server side to scan portions of the database on every request, indicating a computational complexity cost of  $O(n)$ , where  $n$  is the size of the database.

An ORAM-based PIR mechanism is introduced by Iliev and Smith [13], who deploy secure hardware to achieve a cost of  $O(\sqrt{n} \log n)$ . This is better than the poly-logarithmic complexity granted by ORAM for the small database sizes they consider. This work is notable as one of the first full ORAM-based PIR setups.

An improved ORAM-based PIR mechanism with  $O(n/k)$  cost is introduced in [20], where  $n$  is the database size and  $k$  is the amount of secure storage. The protocol is based on a careful scrambling of a minimal set of server-hosted items. A partial reshuffle costing  $O(n)$  is performed every time the secure storage fills up, which occurs once every  $k$  queries. While an improvement, this result is not always practical since the total database size  $n$  often remains much larger than the secure hardware size  $k$ . For  $k = \sqrt{n}$ , this yields an  $O(\sqrt{n})$  complexity (significantly greater than  $O(\log \log n \log n)$  for practical values of  $n$ ).

In [22] Williams et al. introduced a faster ORAM variant which also features correctness guarantees, with computational complexity costs and storage overheads of only  $O(\log n \log \log n)$  (amortized per-query), under the assumption of  $O(\sqrt{n})$  temporary client storage. In their work, the assumed client storage is used to speed up the reshuffle process by taking advantage of the predictable nature of a merge sort on uniform random data.

**Oblivious Transfer with Access Control.** Camenisch et al. [4] study the problem of performing  $k$  sequential oblivious transfers (OT) between a client and a server storing  $N$  values. The work makes the case that previous solutions tolerate selective failures. A selective failure occurs when the server may force the following behavior in the  $i$ th round (for any  $i = 1, \dots, k$ ): the round should fail if the client requests item  $j$  (of the  $N$  items) and succeed otherwise. The paper introduces security definitions to include the selective failure problem and then propose two protocols to solve the problem under the new definitions.

Coull et al. [6] propose an access control oblivious transfer problem. Specifically, the server wants to enforce access control policies on oblivious transfers performed on the data stored: The client should only access fields for which it has the credentials. However, the server should not learn which credentials the client has used and which items it accesses.

Camenisch et al. [3] propose another solution that makes use of capabilities to enable clients to obliviously transfer items from a server. Regardless of the outcome of the interaction between a client and a server, the server does not learn which capabilities the client has. Moreover, the client retrieves the item only if it has enough capabilities to do so. Note however that this is different from our solution, since our solution also allows clients to obliviously write/modify items they can access. Thus, an oblivious transfer is not sufficient.

### 3 Model

Let  $O$  be a database owner and  $S$  be a server that stores the database. In its simplest form, the database is stored as a set of  $n$  pairs,  $D = \{(id_1, v_1), \dots, (id_n, v_n)\}$ , where  $id$  denotes a unique identifier and  $v$  is the value stored under it. We will assume that the data owner knows all the IDs of items stored in his database (or has a an efficient way to compute them directly when needed). A set of *clients*  $\mathcal{C} = \{C_1, \dots, C_c\}$  is given access to items from  $D$ . In our approach, the database owner  $O$  delegates the rights to access items by handing out certain capabilities. We focus on the management of individual items, where a client is provided with access to a single item at a time. While this model can be extended to handle multiple items (e.g., request access to a contiguous range between  $id_1$  and  $id_2$  or to all items in a table column whose values exceeds  $v_1$ ), we prefer our model for simplicity of exposition. We further assume that each client has a secure communication channel to communicate with the data owner. We give a construction for an oblivious database  $D$ -*ORAM* which supports the following operations:

- *Setup()*: Operation called by the owner to generate the initial  $D$ -*ORAM*.
- *Store(id, v)*: Operation that allows the owner to insert a new  $(id, v)$  pair into the  $D$ -*ORAM*.
- *DStore(id, C, ctr, ctr\_C)*: Operation that allows the owner to insert a new dummy item for client  $C$  into the  $D$ -*ORAM*.

- $\text{Delegate}(C, id, op)$ : Delegate to a client  $C$  the right to access an item  $id$  with operation  $op$  (*Read*, *Write* or *Insert*).
- $\text{Read}(id, cap)$ : Access the value of  $id$ , thereby using capability  $cap$ .
- $\text{Write}(id, newV, cap)$ : Modify the value stored under  $id$  to  $newV$ , thereby using capability  $cap$ .
- $\text{Insert}(id, v, cap)$ : Insert a value  $v$  under  $id$  using capability  $cap$ .
- $\text{Reshuffle}(ctr)$ : Reshuffle the  $D\text{-ORAM}$ , where  $ctr$  stores the number of reshuffles performed so far on the  $D\text{-ORAM}$ .

We note that it is further possible to revoke access rights; this can be done efficiently by changing the item key  $k_{id}$ . After changing  $k_{id}$ , the data owner sends the new key to all clients who were allowed to access the item and were not revoked access rights. To efficiently distribute the new key, we suggest to use broadcast encryption (see Section 5.2).

In our analysis, we assume an honest but curious server. The server is trusted to run any protocol correctly, while trying to collect additional information (access patterns or values accessed). We further assume the clients to be purely malicious: They can try to read items they cannot access, modify items even if they only have the right to read them, or learn about the access patterns of other clients. However, we guarantee these strong constraints only for items for which no permissions were given to a corrupted client. Note that this is a natural assumption: It is impossible to prevent a malicious client from publishing an item’s content via other channels or to reveal that an item has been accessed. Furthermore we assume that the owner is trusted – he knows which clients can access which items, and he has full control over the database if desired.

Before building a delegated ORAM, we need to define its security properties. In order to achieve security goals against the server, these need to capture all the security guarantees offered by the standard, single-client ORAM. We therefore require the  $D\text{-ORAM}$  to satisfy the security properties against a curious server as outlined in [10] and the following security properties against malicious clients:

- *Access Security*: An  $D\text{-ORAM}$  offers *Access Security*, if no client can read or write an item  $id \in D\text{-ORAM}$  without having proper capabilities.
- *Access Privacy*: We say that an item  $id$  in  $D\text{-ORAM}$  has been compromised, if there exists a corrupted client  $C_M$  with access to  $id$ . An  $D\text{-ORAM}$  offers *Access Privacy*, if for any item  $id \in D\text{-ORAM}$ , which has not been compromised, no client without access to  $id$  can tell with non-negligible probability whether the item has been accessed, or not.

## 4 The Delegated ORAM Solution

Our solution is built on the “square root” ORAM variant described in Section 2 and relies on a novel use of capabilities. The data owner  $O$  issues a capability allowing a client  $C$  to access a certain item in the  $D\text{-ORAM}$ . Recall that in the square root ORAM, the database stores  $n + \sqrt{n}$  items, where  $n$  items are “real”

and  $\sqrt{n}$  items are dummy values. Thus, in our solution, each client is assumed to have access to  $\sqrt{n}$  dummy or private items – real items that no other client can access. This increases storage at  $S$  by  $c\sqrt{n}$ , where  $c$  is the number of clients. In the following we will make extensive use of a buffer called “cache” buffer of size  $\sqrt{n}$  stored at  $S$ . The buffer starts empty.

Each item  $id$  stored in the *D-ORAM* has a key associated with it, denoted by  $k_{id}$ . The item is stored encrypted with  $k_{id}$ , providing confidentiality from  $S$ .  $O$  either stores all keys or is able to compute them on demand (e.g., using a private, general purpose database key and a pseudo-random generator). Each item is stored in the *D-ORAM* as a  $(tag, v, keybox)$  triplet, where  $tag$  is a public pseudo random string (derived from  $id$  as shown below) used to retrieve the item from the *D-ORAM*,  $keybox$  is an encrypted version of the item key  $k_{id}$  which will be used during the reshuffle and  $v$  denotes an encryption of the actual database item. The latter includes the item  $id$ , the actual value stored under this id, a version number for this item (which will be incremented upon each write operation) as well as a signature which allows to verify that the item-value was written correctly.  $C$  is allowed to access an item  $id$  only if it knows  $k_{id}$ . Thus, a capability for  $id$  needs to include  $k_{id}$ .

**Tag Generation.** Each item  $id$  in the database (including the dummy values) is assigned a tag, chosen pseudorandomly. Note that, if these tags were chosen uniformly at random, after each reshuffle  $O$  would have to notify each client  $C$  of the new tags assigned to items (including the dummy ones) it can access. To avoid this, we compute the tags as  $t(id) := h(id, ctr, k_{id})$ , where  $h$  is a publicly known pseudo random function,  $ctr$  is a counter, which counts the number of reshuffle operations performed so far, and  $k_{id}$  is the secret key corresponding to item  $id$ . This ensures that clients allowed to access item  $id$  (i.e., that know  $k_{id}$ ) will be able to determine its tag after a reshuffle.

Keeping track of the tags for dummy items is done similarly. Each client maintains a personal counter  $ctr_C$ , indicating the number of unused dummy items. Using a unique client dummy password  $d_C$ , the current value of the counter  $ctr$  (which counts the number of reshuffle operations) and the personal counter  $ctr_C$  we compute the tags as  $h(d_C, ctr, ctr_C)$ . The passwords  $d_C$  will be unique for each client, known to only client and the data owner.

## 4.1 D-ORAM Operations

The *Setup* operation is called by the owner before the first *D-ORAM* operation is performed to populate the RAM.

**Setup().** Initially,  $O$  calls the operation  $\mathbb{G}$  of an identity based signature scheme, which outputs a master secret  $\mathcal{M}_S$  and a public master key  $\mathcal{M}_P$ . He further sets  $ctr := 0$  and chooses a symmetric key  $k_O$  which will be used exclusively by  $O$ . Next, he calls the *Store* and *DStore*  $n + c\sqrt{n}$  times, once for each data or dummy item that needs to be stored in the *D-ORAM*. Notice that  $O$  will add these items in random order to the *D-ORAM* (to hide from the server which of them are dummy items). Furthermore, he allocates an empty cache buffer.

**Store( $id, v, C, ctr$ ).**  $Store$  is executed by  $O$  when a new data item is to be inserted into the  $D$ -ORAM.  $O$  generates a secret key  $k_{id} \in \{0,1\}^k$  and uses  $\mathbb{K}_{\mathcal{M}_S}$  to generate a private key  $s_{id}$  for the value  $id$ . Further,  $O$  generates the tag  $t(id)$  and outputs  $(t(id), ev, E_{k_O}(k_{id}))$ , where  $ev = E_{k_{id}}(id, v, 0, \mathbb{S}(s_{id}, (v, 0)))$  is the encryption of item  $id$  with version number 0. The value  $E_{k_O}(k_{id})$  will help  $O$  to recover the decryption key when presented with the encrypted item. Finally,  $O$  asks the server to insert this tuple in the database.

**DStore( $id, C, ctr, ctr_C$ ).** The owner executes this operation to insert a dummy item for some client  $C = C_i$ .  $DStore$  generates a tag  $tag = h(d_C, ctr, ctr_C)$  for client  $C$  and counters  $ctr, ctr_C$  and a string  $s$  having the same distribution as the output of  $E(\cdot)$ . Finally, the triplet  $(tag, s, E_{k_O}("dummy", C, ctr))$  is added to the  $D$ -ORAM.

We now describe the capability issuing operation, performed by  $O$ .

**Delegate( $C, id, op$ ).** This operation outputs a capability  $cap$  which can be used in *Read*, *Write* or *Insert* operations. First generate the value  $k_{id}$  just like in the *Store* operation. Next, if  $op = Read$ , output the tuple  $(id, k_{id})$  and return. If  $op = Write$ , generate the secret signing key  $s_{id}$  (just like in *Store*), output  $(id, k_{id}, s_{id})$  and return. If  $op = Insert$ , output  $(id, k_{id}, s_{id}, E_{k_O}(k_{id}))$  and return.

The *Read*, *Write* and *Insert* operations behave similarly to their basic ORAM variant. They are executed by a client  $C$ .

**Read( $id, cap$ ).** Given a capability  $cap = (id, k_{id})$ , scan all cache buffer items starting from the most recently added. Retrieve each element as  $(value, keybox)$ . Decrypt each element  $value$  using the key  $k_{id}$ . If any decryption has the format  $(id, v, ver, sig)$ , then check that  $\mathbb{V}_{\mathcal{M}_P}(sig, id, (v, ver))$  verifies correctly. Note that since we are using an identity based signature scheme, each signature can be verified by using the value  $id$  and the master public key  $\mathcal{M}_P$ . If the check does not verify, discard the item and continue with the next item. If no correct item is found, compute the tag  $t(id)$  and request the item with this tag from the  $D$ -ORAM database, obtain element  $(tag, value, keybox)$  and decrypt its second field  $value$ . If the desired element had been found in the cache buffer, request the next unused dummy item from the  $D$ -ORAM database and obtain  $(td, s, keybox)$  – discarding values  $td$  and  $s$  immediately while storing the value  $keybox$ . If all verifications pass and the decryption has been performed correctly, use the value  $v$  as the actual item value. Finally, re-encrypt the message  $m = E_{k_{id}}(id, v, ver, sig)$  using  $k_{id}$ . It is necessary to re-encrypt this message before storing the item back into the buffer, to hide from the server whether it was found in the buffer or had been retrieved from the main database. Insert the result into the cache buffer along with the value  $keybox$  of the item which was requested from the main  $D$ -ORAM database (note that we always use the value  $keybox$  derived from the main database – real or dummy item – in order to keep all  $D$ -ORAM accesses indistinguishable). Output  $v$  and return.

**Write**( $id, newV, cap$ ). Given a capability  $cap = (k_{id}, id, s_{id})$ , proceed as in *Read*, except that the value appended to the cache buffer is  $E_{k_{id}}(id, newV, ver + 1, \mathbb{S}(s_{id}, (newV, ver + 1)))$ , where  $ver$  is the version number of the most recent item  $id$  when scanning the buffer and the main database.

**Insert**( $id, v, cap$ ). For a given capability  $cap = (id, k_{id}, s_{id}, E_{k_O}(k_{id}))$ , append the tuple  $(m, keybox) = (E_{k_{id}}(id, v, 0, \mathbb{S}(s_{id}, (v, 0))), E_{k_O}(k_{id}))$  to the cache buffer. Note that, by adding items in this manner, the server will notice when an insert has occurred. This problem can be prevented by adding sufficiently many dummy items to the initial ORAM and replacing them with real items whenever *Insert* is called. In fact, this can as well be done incrementally: If it is known that the database on average grows by  $k$  items per epoch, the data owner can add additional  $k$  dummy items during each re-shuffle operation. This efficiently hides the time the *Insert* occurred. To support incremental inserts, some minor adaptions to our construction are necessary. In particular, the form of dummy elements needs to be changed slightly to allow the data owner to recover the correct value *keybox* during the reshuffle.

**Reshuffle**( $ctr$ ). The database *Reshuffle* operation is performed by  $O$ . The reshuffle is performed in five steps:

*Step 1:* Use  $E(\cdot)$  to encrypt each item in the *D-ORAM* (including the buffer) with a fresh session key  $k_{rs}$ , used exclusively to perform the reshuffle. Note that this essentially works like a second layer of encryption for items in the database. Thus, in steps 2 - 4 we will always assume that items are first decrypted using  $k_{rs}$  when accessed by the data owner, and encrypted again before stored back on the server.

*Step 2 (Clean the Cache):*  $O$  verifies the validity of each updated item: items which fail to verify correctly should never appear in the main database. Download each element  $(v, keybox)$  from the buffer, starting at the last inserted item. Perform the following actions:

- If  $D_{k_O}(keybox) = k_{id}$  and  $D_{k_{id}}(v)$  correctly decrypts and verifies to a valid item  $id$ , continue by scanning the earlier items in the buffer. Mark items with the same  $id$  for deletion.
- If  $D_{k_O}(keybox)$  contains “dummy”, scan the earlier items in the buffer until a valid value *keybox*, containing the correct key  $k_{id}$ , is found. Update the *keybox* encryption to  $E_{k_O}(k_{id})$  and continue.
- Otherwise (e.g. if the signature can not be verified or if no valid key  $k_{id}$  could be determined), mark the item as invalid and continue.

*Step 3:* Read each item stored in the *D-ORAM*, generate a new tag  $t(id)$  for it and store it back.

*Step 4:* Perform the re-shuffle operation as described in the basic square-root ORAM solution [10], i.e. obliviously update the database items’ values according to the buffers, re-encrypt them using the corresponding key  $k_{id}$ , and obliviously permute the database locations.

*Step 5:* Decrypt each item in the *D-ORAM* with the session key  $k_{rs}$ .

## 4.2 Security Analysis

We discuss the security of our construction as introduced in Section 3.

*Security against a curious server.* The proof is identical to the one for ORAM with only one client [10]. It is easy to verify that, from the server’s point of view, every time the *D-ORAM* is accessed, all operations are performed in precisely the same way. In particular, in the reshuffle the steps 1, 2, 3 and 5 are performed in the same deterministic way in each epoch, while step 4 consists of the reshuffle used in the single client ORAM [10]. All values the server can see are pseudo random and therefore do not reveal information to the server.

*Security against malicious clients.* The construction achieves D-ORAM *Access Security*: First, notice that an unauthorized attempt to overwrite an item can be detected and the original item’s value retrieved by any other client who is allowed to access this item. Even if a corrupted client knew  $k_{id}$ , the unforgeability property of the signature scheme ensures that without knowing  $s_{id}$ , he cannot produce a valid signature of a new item’s value  $v$ . Hence, if a client finds that supposedly a new item’s value is not correctly signed, he simply uses the last one that passes the signature verification in his computation. Also, in the reshuffle phase, the owner ensures that the items are updated to their correctly signed values. Notice further that IND-CPA security of the encryption scheme guarantees confidentiality for each item, in the sense that no collusion of clients without the capability to read this item can learn its value.

Furthermore, the solution also provides D-ORAM *Access Privacy*: To show that a client, who cannot access any of the items in a set, learns nothing about the computation on them, we use the same argument as in the case of the server: In case that no malicious client compromised the privacy of item  $id$ , every access to this item is indistinguishable from a random access to the *D-ORAM* for all clients who are not able to access item  $id$ .

## 4.3 Complexity

Using a Batcher network to shuffle a database containing  $n$  regular and  $d = c\sqrt{n}$  dummy items requires  $O((n + d) \log^2(n + d))$  comparisons. In addition, a reshuffle requires  $O(n + d)$  operations (encrypt/decrypt each item once, update the buffer). When manipulating one item,  $O(\sqrt{n})$  items need to be read. Hence, between two reshuffles,  $O((n + d) \log^2(n + d) + n + d) = O((n + d) \log^2(n + d))$  operations are needed. We therefore get the amortized complexity to be of order  $O(\frac{n+d}{\sqrt{n}} \log^2(n+d)) = O((\sqrt{n}+c) \log^2(n+d))$ , where we assume that  $c$  is a small constant compared to  $\sqrt{n}$ .

## 5 Discussion

### 5.1 Beyond a Curious Server

While above we considered the case of a curious yet otherwise honest server, here we discuss also some insights into malicious server behavior.

It makes little sense to handle outright denial of service behavior at this level, as the server has many natural avenues at his disposal to restrict service, including simply shutting down. More interesting to explore are attacks in which the server illicitly and *undetectably* manages to satisfy its curiosity by behaving incorrectly. We distinguish a set of scenarios, some of which are discussed in the following.

*Fork Consistency.* The server may attempt to partition the set of clients, and maintain separate versions of the database state (buffer, main database) for each partition. This partitioning attack has been examined in previous literature; if there are non-inter-communicating asynchronous clients, the best that can be guaranteed is fork consistency [15]. This is not as weak of a guarantee as it may appear, as once the provider has created a partition, the provider must block all future communication between partitioned clients, or else the partition will be immediately detected.

*Altering Responses.* Additionally, the server may attempt to substitute messages and previously read data for new requests. This can naturally be addressed by a combination of minimal client state based mechanisms that can checksum the server responses. For instance, the client could deploy Merkle tree based approaches coupled with item versioning to defend against such attacks. As numerous existing efforts already addressed such mechanisms we chose not to detail them here.

*Timing Attacks.* In such attacks, the server measures the time intervals taken by a client to parse the buffers and to access the ORAM database. This might enable the server to learn which type of operation was performed, and/or where the desired item was found. We suggest to prevent this attack by introducing the requirement that each access to an item stored on the server (buffer or database) takes the same time. This can be achieved by each client using additional timers to “uniformize” inter-request times.

## 5.2 Efficient Access Right Updates Using Broadcast Encryption Schemes

In the protocol construction, we omitted the details of access right updates. A naive solution to revoke the access rights of a set of clients to a particular item, is to change the item’s secret key  $k_{id}$  and broadcast a new key, encrypted with the public keys of all clients in the target set. If there are  $c$  clients, this solution potentially requires to encrypt the item’s secret key with  $\Theta(c)$  public keys.

A more efficient way to solve the problem is to use *broadcast encryption schemes* [2,7,16,11]. The main idea of broadcast encryption schemes is to associate keys to the subsets of clients and represent any set of privileged clients as a union of these subsets. In the naive solution, each client is given a unique key. A better result can be achieved by building a binary tree of keys with clients representing its leaves, and give each client all private keys on the path from the corresponding leaf to the root. The privileged set of clients is then covered by a set of subtrees and the (public key, private key)-pairs in the roots of these

subtrees are used for encryption/decryption of an item’s content. This scheme is still inefficient if access right updates involve revoking small sets of clients. For instance, to revoke a single client,  $\log c$  subtrees might be needed to cover all the remaining clients.

A better performance is achieved in [16,11]. Following the approach of Halevy and Shamir [11], any set of  $r$  clients can be revoked by the owner, broadcasting only  $O(r)$  (at most  $4r$ ) encryptions. In this scheme, each client is given  $O(\log^{1+\epsilon} c)$  private keys ( $O(\log^{3/2} c)$  in the basic case of practical interest) and performs  $O(\log c)$  decryption operations.

## 6 Conclusions

In this paper we study the problem of delegating access to an outsourced private database to multiple clients, while preserving the access privacy of all involved entities. Our solution extends existing ORAM flavors with the notion of capabilities, allowing data owners to delegate and revoke permissions and clients to privately read, write and insert items. We show that our solution provides reasonable security guarantees and protects the privacy of the involved parties. We further note that more efficient versions of ORAM can be constructed based on the so called “poly-log”-solution [10]. While in this paper we provide a basic ORAM solution which allows to give access to the ORAM to multiple parties, it might be interesting to investigate whether similar solutions could be applied to the more efficient “poly-log”-solution. We leave these, as well as further optimizations, as future work.

**Acknowledgments.** We thank Ian Goldberg for insightful discussions and the anonymous reviewers of FC’11 for helpful comments. Sion and Sotakova were supported by NSF through awards 0937833, 0845192, and 0803197, as well as by CA Technologies, Xerox, IBM and Microsoft Research. This work was supported by CASED (<http://www.cased.de>), the German Research Foundation (DFG) and DAAD.

## References

1. Asonov, D. (ed.): *Querying Databases Privately*. LNCS, vol. 3128. Springer, Heidelberg (2004)
2. Berkovits, S.: How to Broadcast a Secret. In: Davies, D.W. (ed.) *EUROCRYPT 1991*. LNCS, vol. 547, pp. 535–541. Springer, Heidelberg (1991)
3. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: CCS 2009: Proceedings of the 16th ACM Conference on Computer and Communications Security (2009)
4. Camenisch, J.L., Neven, G., Shelat, A.: Simulatable Adaptive Oblivious Transfer. In: Naor, M. (ed.) *EUROCRYPT 2007*. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
5. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: IEEE Symposium on Foundations of Computer Science, pp. 41–50 (1995)

6. Coull, S., Green, M., Hohenberger, S.: Controlling Access to an Oblivious Database Using Stateful Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 501–520. Springer, Heidelberg (2009)
7. Fiat, A., Naor, M.: Broadcast Encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
8. Gasarch, W.: A WebPage on Private Information Retrieval,  
<http://www.cs.umd.edu/~gasarch/pir/pir.html>
9. Gasarch, W.: A survey on private information retrieval,  
<http://citeseer.ifi.unizh.ch/gasarch04survey.html>
10. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. Journal of the ACM 43, 431–473 (1996)
11. Halevy, D., Shamir, A.: The LSD Broadcast Encryption Scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
12. IBM. IBM 4764 PCI-X Cryptographic Coprocessor (2007),  
<http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml>
13. Iliev, A., Smith, S.W.: Private information storage with logarithmic-space secure hardware. In: Proceedings of i-NetSec 2004: 3rd Working Conference on Privacy and Anonymity in Networked and Distributed Systems, pp. 201–216 (2004)
14. Jarecki, S., Shmatikov, V.: Handcuffing Big Brother: an Abuse-Resilient Transaction Escrow Scheme. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 590–608. Springer, Heidelberg (2004)
15. Li, J., Krohn, M., Mazières, D., Shasha, D.: Secure Untrusted Data Repository (SUNDR). In: OSDI 2004, pp. 121–136 (2004)
16. Naor, D., Naor, M., Lotspiech, J.B.: Revocation and Tracing Schemes for Stateless Receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
17. Paterson, K.G.: Id-based signatures from pairings on elliptic curves. Electronics Letters 38, 1025–1026 (2002)
18. Pinkas, B., Reiman, T.: Oblivious ram revisited. In: Proceedings of the 30th International Cryptology Conference (2010) (to appear)
19. Sion, R., Carbunar, B.: On the Computational Practicality of Private Information Retrieval. In: Proceedings of the Network and Distributed Systems Security Symposium (2007); Stony Brook Network Security and Applied Cryptography Lab Tech Report 2006-06
20. Wang, S., Ding, X., Deng, R.H., Bao, F.: Private Information Retrieval using Trusted Hardware. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 49–64. Springer, Heidelberg (2006)
21. Williams, P., Sion, R., Carbunar, B.: Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: CCS 2008: Proceedings of the 15th ACM Conference on Computer and Communications Security (2008)
22. Williams, P., Sion, R., Carbunar, B.: Building Castles out of Mud: Practical Access Pattern Privacy and Correctness on Untrusted Storage. In: ACM Conference on Computer and Communication Security CCS (2008)

# Homomorphic Signatures for Digital Photographs\*

Rob Johnson, Leif Walsh, and Michael Lamb

Stony Brook University

`rob@cs.sunysb.edu`, `rlwalsh@ic.sunysb.edu`, `mike@datagrok.org`

**Abstract.** We describe two homomorphic signature schemes for digital photographs such that an intermediate party in possession of a signed photograph can construct a scaled, cropped, and lossily compressed version of the photograph along with a new, valid signature, *without knowing the private signing key*. In other words, our signature schemes are simultaneously homomorphic with respect to cropping, scaling, and JPEG-like compression. Unlike prior ad-hoc schemes for photographic signatures, our first scheme is provably secure and quite practical. For example, a scaling-homomorphic signature scheme using our techniques requires less than 100KB of signature data for typical digital photographs. Our second signature scheme has weaker security but reduces typical signature sizes to 15KB. Both schemes extend naturally to authenticate movies and other digital media and use novel, multi-dimensional variations of Merkle hashing and GGM trees related to constructions used in computational geometry that may be of independent interest.

## 1 Introduction

We present homomorphic signature schemes that are simultaneously homomorphic with respect to cropping, scaling, and JPEG-like lossy compression. With such a signature, anyone possessing a signed digital image can perform any combination of these image edits and, by performing corresponding operations on the signature, create a new scaled, cropped, compressed image with a valid signature, without knowing the private signing key. Our signature schemes are efficient, requiring only one public-key operation and, for typical uses, under 100KB of signature data for digital photographs under 16 megapixels. Unlike previous ad-hoc attempts to create homomorphic signatures for digital photographs, our primary signature scheme is provably secure. The second signature scheme we present sacrifices some security, but reduces signature sizes to 15KB for typical digital images. Our homomorphic signatures extend easily to higher-dimensional data so, for example, we could create a movie signature scheme that is also homomorphic w.r.t. scene cuts and deletions.

Digital photographs are ubiquitous, so signature schemes homomorphic with respect to common image operations could have numerous applications. A digital camera equipped to produce a signature of each photograph it creates would

---

\* This research was supported by National Science Foundation grant CNS 0627645.

enable photographers to prove that their photographs are real and unaltered. These photographs could then be cropped and scaled as appropriate, and the final viewer could verify that the photograph they see is authentic. Scientific journals could require such signatures on photographic evidence in their submissions, preventing fraud such as the human cloning forgeries published in *Science*[32]. Online news sites could use such signatures to provide an end-to-end proof that photos accompanying their new stories are real, preventing photo-journalism fraud such as Reuters' digitally altered 2006 Lebanon war photographs[34]. Police could use these signatures to prove that crime-scene photos or security camera footage is authentic.<sup>1</sup>

Our signature scheme follows the redactable signature framework of Johnson, et al[12], but uses novel multi-dimensional variants of Merkle hashing (Section 4) and GGM trees (Section 5). Standard Merkle hashing computes a value,  $h$  for a data vector,  $x$ , such that, by presenting  $O(\log(|x| - |x'|))$  witnesses, one can prove that some contiguous subvector  $x'$  of  $x$  was part of the original data used to compute  $h$ . In our extension, one can prove that some hyperrectangular submatrix  $A'$  was part of an original matrix  $A$  used to compute the hash, although more witnesses are required. For example, in 2-dimensions, we require  $O((\log HW)^2)$  witnesses, where  $W$  and  $H$  are the width and height of the original matrix. This hashing scheme may be of independent interest. The GGM PRNG construction generates a sequence of  $n$  pseudorandom outputs such that one can reveal any contiguous subsequence by only communicating  $\log n$  PRNG seeds. We present two 2-dimensional analogs. The first uses a GGM tree and a space-filling curve to generate an  $H \times W$  matrix of pseudorandom outputs such that any  $h \times w$  submatrix of outputs can be revealed by transmitting  $O(h + w)$  PRNG seeds. The second construction is not a PRNG, but it suffices to construct cropping-, scaling-, and compression-homomorphic signature schemes that are secure against an adversary that makes 1 signing oracle query and it reduces the number seeds needed to  $O(\log hw)$ . We then use these building blocks to create two cropping-homomorphic signature schemes (Section 6).

We then describe how to convert any cropping-homomorphic signature scheme into a scaling-homomorphic signature scheme by observing that scaling an image is equivalent to cropping certain coefficients of its Discrete Cosine Transform (DCT) matrix[26] (Section 7). Thus, by representing the image as its DCT and signing the DCT representation with a cropable signature scheme, we can create an image format and accompanying signature that is homomorphic with respect to scaling. To support cropping and scaling simultaneously, we divide the image into blocks, take the DCT of each block, and then sign this data with a 4-dimensional cropping-homomorphic signature scheme. Cropping two of the four dimensions corresponds to deleting entire blocks, which is equivalent to cropping the original image. Cropping in the other two dimensions corresponds to cropping within all the DCT blocks, which corresponds to scaling each block of the original image, which is equivalent to scaling the original image. To enable

---

<sup>1</sup> This technology can not prevent all kinds of photographic forgery, but it can make forgeries significantly more expensive to produce.

lossy JPEG-like compression, we divide the DCT coefficients into their bitplanes and perform the above signatures on each bitplane independently. Thus, we represent the image as a 5-dimensional matrix in which cropping in each dimension corresponds to one of our supported image operations – cropping height, cropping width, scaling height, scaling width, and JPEG-like compression. We then describe a few tricks to simplify the signatures and make them more efficient.

Section 8 presents performance results for prototype implementations of our signature schemes. Our experiments show that, in the average case, our signatures can be substantially smaller than predicted by the worst-case analysis of Sections 4 and 5. We discuss open problems and make concluding remarks in Section 9.

## 2 Related Work

*Multimedia Authentication* Numerous authors have studied cryptographic methods of verifying the integrity of photographs and other digital media [15,23,5,14,37,36,35,39,38,33,9,10,28,30,29,20], but these schemes all have at least of the following deficiencies: they are insecure, they are less efficient, or they support fewer image operations than our signature scheme. The JPEG2000 security extension has also inspired a substantial amount of research[20,30,10,9,33,38]. Some of these schemes, particularly that of Peng, et al[20], use Merkle hash trees and thus may benefit by applying our multidimensional variant to reduce signature sizes or to support more image operations.

*Statistical Forgery Detection* Other researchers have developed statistical and other consistency tests to detect evidence of tampering in digital photographs [3,6,7,16,22,21,11,19]. Unfortunately, all these tests are vulnerable to an “oracle” attack: an attacker can apply the same tests to his candidate image, grooming it until it passes. Thus these tests may catch a casual attempt at forging a photograph, but they cannot stop a determined fraudster.

*Homomorphic Signatures* Homomorphic signatures were first proposed by Rivest in 2001[24], and Micali and Rivest presented the first such scheme, for graphs, in 2002[17]. Johnson, et al, published their redactable and set-homomorphic signature schemes at the same conference[12]. These initial schemes have inspired others[27,13,1]. In contrast to these other schemes that develop signatures homomorphic with respect to one operation, this paper presents techniques for creating efficient signatures that are homomorphic with respect to several document operations simultaneously.

## 3 Redactable Signatures

Our signature schemes builds on the redactable signature scheme of Johnson, et al[12], so we summarize that scheme here.

Given a vector  $x = (x_0, \dots, x_{n-1})$ , a vector  $x'$  is a redaction of  $x$  if  $x'$  is also of length  $n$  and for all  $0 \leq i < n$ ,  $x'[i] = x[i]$  or  $x'[i] = \perp$ , where  $\perp$  is a special symbol indicating the position  $i$  has been erased. A redactable signature scheme has three phases:

1. The private-key holder signs  $x$ , creating  $s = \text{Sig}(x)$ , and transmits  $(x, s)$  to the redactor.
2. The redactor replaces some positions of  $x$  with  $\perp$ , creating a redacted vector  $x'$ . Simultaneously, the redactor uses  $s$  to derive a new signature  $s'$  on  $x'$ . Note that the redactor does not have access to the private signing key used to generate  $s$ . The redactor then publishes  $(x', s')$ .
3. Some third party obtains  $(x', s')$  and uses the original signer's public key to verify that  $s'$  is a valid signature for  $x'$ .

As with normal signatures, redactable signatures should be unforgeable, but the notion of a forgery must be changed since anyone is allowed to construct signatures of redactions of signed vectors. A forger may also see several different signed redactions of the same original document. In general, given signed redactions  $x_1, \dots, x_n$  of some document  $x$ , let the *join*,  $x'$ , of  $x_1, \dots, x_n$  be

$$x'[i] = \begin{cases} x_j[i] & \text{for some } j \text{ if } x_j[i] \neq \perp \\ \perp & \text{if } x_j[i] = \perp \text{ for all } j \end{cases} \quad (1)$$

The forger should not be able to construct a signature on some document  $x^*$  that is not a redaction of  $x'$ .

To model this scenario, we give the adversary access to two oracles,  $S$  and  $R$ . The adversary can use oracle  $S$  to register vectors  $x_1, \dots, x_q$ . For each registered vector,  $S$  computes and stores a signature  $s_i$ , but does not return it to the adversary. When the adversary makes a query  $R(i, x)$  such that  $x$  is a redaction of  $x_i$ , the oracle uses  $s_i$  to compute a signature on  $x$  and returns it to the adversary. For each  $i$ , let  $W_i$  be the join of all vectors  $x$  that appear in a query of the form  $R(i, x)$ . We say the adversary has created an *existential forgery* if it produces a (possibly redacted) vector  $x^*$  with valid signature  $s^*$  such that  $x^*$  is not a redaction of  $W_i$  for any  $i$ .

**Definition 1.** A redactable signature scheme is  $(t, q, \epsilon)$ -secure against existential forgeries if, for all adversaries,  $A$ , running in time  $t$  and making at most  $q$  queries to  $S$ ,

$$\Pr[A^{S, R} \text{ outputs an existential forgery}] \leq \epsilon \quad (2)$$

The redactable signature scheme of Johnson, et al, uses three building blocks: a length-doubling secure PRNG  $G : \{0, 1\}^\ell \rightarrow \{0, 1\}^{2\ell}$ , a hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ , and any standard signature algorithm,  $\text{Sig}_0$ . Let  $G_0(r)$  and  $G_1(r)$  be the first and second halves, respectively, of  $G(r)$ . To compute a redactable signature on a vector  $x$  of length  $n$ , the signer picks a random seed,  $k_\epsilon$ , and executes the following steps.

1. Build a GGM tree of height  $\log n$  from the seed  $k_\epsilon$ . We can label each node of the tree according to the path from the root to the node. Thus, for example,

- a node  $k_w$  has children  $k_{w0} = G_0(k_w)$  and  $k_{w1} = G_1(k_w)$ . Since the tree has height  $\log n$ , it contains one leaf for each entry in  $x$ , and we can interpret the label on each leaf as a binary integer to obtain the index of the corresponding element of  $x$ .
2. Set  $v_w = H(0, k_w, x_w)$  for each entry in  $x$ , and build a Merkle hash tree from the  $v_w$  values via the rule  $v_\eta = H(1, v_{\eta0}, v_{\eta1})$ . The root of this tree will be  $v_\epsilon$ .
  3. Set  $\text{Sig}(x) = (k_\epsilon, \text{Sig}_0(v_\epsilon))$ .

Given a redactable signature  $(k_\epsilon, \text{Sig}_0(v_\epsilon))$  on some vector  $x$ , a verifier can use  $k_\epsilon$  to repeat steps 1 and 2 of the above algorithm to obtain  $v_\epsilon$  and then verify the signature  $\text{Sig}_0(v_\epsilon)$ .

Suppose a redactor deletes a suffix of  $x$  to obtain  $x'$ . The recipient of  $x'$  will not be able to compute the values  $v_w$  that correspond to deleted entries in  $x'$ , and hence will not be able to verify the signature. To overcome this problem, the redactor can reveal the hashes  $v_w$  corresponding to each deleted position in  $x'$ . Notice that the redactor can save space since, whenever he reveals two siblings  $v_{\eta0}$  and  $v_{\eta1}$ , he could simply reveal the witness  $v_\eta$  instead. After recursively coalescing hashes in this way, the redactor only has to reveal the  $O(\log n)$  hashes at the siblings of the nodes along the path from the right-most non-deleted entry in  $x'$  to the root.

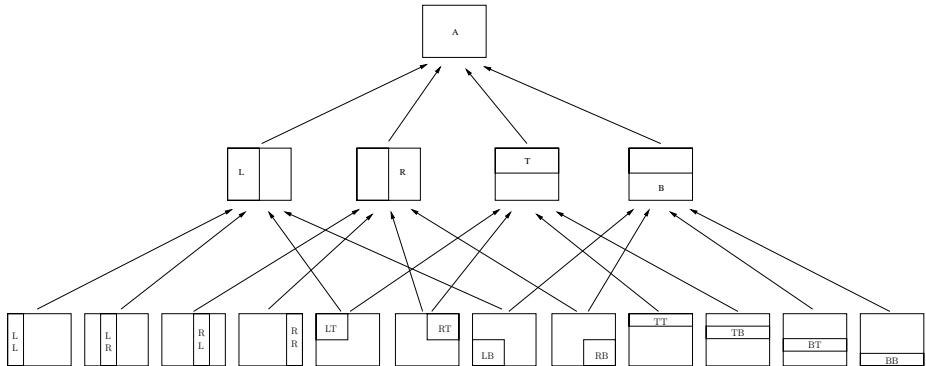
If the entries of  $x$  are easy to guess, though, then an attacker who sees a redacted vector  $x'$  might be able to guess the missing entries and use the leaves of the GGM tree and the revealed hashes to verify his guesses. Thus, the redactor cannot let an attacker learn the GGM nodes corresponding to deleted positions of  $x'$ . To prevent this, the redactor erases  $k_\epsilon$  from the signature and instead includes all the GGM leaves corresponding to non-deleted positions in  $x'$ . Then, as with the hash tree, the redactor can combine revealed siblings  $k_{w0}$  and  $k_{w1}$  and reveal only their parent,  $k_w$ .

Taken together, a redactor can delete the suffix of  $x$  and create a new signature on  $x'$  by revealing  $O(\log n)$  GGM tree values and  $O(\log n)$  hash witnesses. In general, deleting a contiguous region of  $x$  requires revealing  $O(\log n)$  GGM nodes and  $O(\log n)$  witnesses.

Given two signatures  $s_1$  and  $s_2$  on redactions  $x_1$  and  $x_2$ , respectively, derived from a signature  $s$  of the original message  $x$ , one can create a signature  $s'$  on the vector  $x'$  defined by  $x'[i] = x_1[i]$  if  $x_1[i] \neq \perp$  and  $x'[i] = x_2[i]$  otherwise. A position in  $x'$  is deleted only if it is deleted in both  $x_1$  and  $x_2$ . This property seems harmless, since  $s'$  does not convey any new information that is not already apparent from  $s_1$  and  $s_2$ . Our cropping-homomorphic signatures will have the same property.

### 3.1 A Naive 2-Dimensional Construction

The above construction efficiently supports redactions of contiguous sections of a signed vector  $x$  and we would like to extend this property to higher dimensional data. Specifically, we would like a scheme for signing a matrix  $A$  such that we



**Fig. 1.** Two levels of  $D_A$ , a 2-dimensional hash structure

can efficiently “crop” the matrix, i.e. cut out a rectangular submatrix  $A'$ , and produce a new signature for the cropped submatrix.

Naively applying the above scheme yields a workable but inefficient cropable signature scheme. Given an  $H \times W$  matrix  $A$ , we can interpret  $A$  as a vector of length  $HW$  using row-major order and then sign this vector with the redactable scheme above. Using this arrangement, cropping an  $h \times w$  submatrix from  $A$  corresponds to deleting  $h+1$  contiguous regions from  $A$ 's vector representation. Thus the signatures generated by this scheme contain  $\Omega(h \log(W-w))$  witnesses and  $\Omega(h \log w)$  GGM values. If we instantiate this scheme using 256-bit hashes and 128-bit PRNG seeds, then the signature on a cropped subimage of a 16 megapixel digital photograph can be over 2MB in size.

In this paper, we improve on this solution in two ways. First, we present a new hashing structure that reduces the number of hash witnesses in the signature to  $O(\log HW \log hw)$ . We then present two alternatives to the above GGM tree arrangement. The first alternative is provably secure against normal adversaries but only reduces the number of seeds in a signature to  $O(w+h)$ . The second construction is only provably secure against an adversary that makes 1 signing oracle query, but is probably secure given some reasonable assumptions about the entropy of pixels in digital photographs. With the second construction, we only need  $O(\log hw)$  seeds in a cropped signature.

## 4 Merkle Hashing for Multi-dimensional Data

Given a multi-dimensional array  $A$ , we can extend Merkle hashing to this array by defining a set of canonical subregions of the array, ordered by inclusion. These subregions will form a directed acyclic graph, and we can compute a hash for each subregion based on the hashes of its children. We begin by defining the subregions and hashing algorithm, and then we analyze the efficiency and security of this hashing scheme.

Let  $A$  be an  $H \times W$  matrix. In the digital photograph application,  $A$  will be the pixels in a photograph. The matrix  $A$  induces a DAG  $D_A$  as follows. The reader may wish to refer to Figure 1, which shows two levels of  $D_A$ . A node  $r \in D_A$  will be of the form  $r = (Y, X)$  where  $X$  and  $Y$  are the intervals that determine the canonical subregion to which  $r$  corresponds. The DAG has a single root,  $r_A = ([0, H - 1], [0, W - 1])$ . A node  $r = ([a, b], [c, d])$  has up to 4 children:

$$\begin{aligned} r^T &= ([a, \lfloor \frac{a+b}{2} \rfloor], [c, d]) & r^B &= ([\lfloor \frac{a+b}{2} \rfloor + 1, b], [c, d]) \\ r^L &= ([a, b], [c, \lfloor \frac{c+d}{2} \rfloor]) & r^R &= ([a, b], [\lfloor \frac{c+d}{2} \rfloor + 1, d]) \end{aligned} \quad (3)$$

corresponding to  $r$ 's top, bottom, left, and right halves, respectively. Every region has either 0, 2, or 4 children: if  $a = b$  then  $r$  will not have top or bottom children; if  $c = d$  then  $r$  will not have left and right children. Analogous to the notation from redactable signature schemes, nodes of  $D_A$  can be specified via strings from the language  $\{R, L, T, B\}^*$ . In this setting, the correspondence is not one-to-one because  $r^{TR} = r^{RT}$  and  $r^{BL} = r^{LB}$ , although  $r^{TB} \neq r^{BT}$  and  $r^{LR} \neq r^{RL}$ . If  $T_Y$  and  $T_X$  are binary interval trees on  $[a, b]$  and  $[c, d]$  respectively, then  $T_Y$  has  $2H - 1$  nodes,  $T_X$  has  $2H - 1$  nodes, and  $D_A \equiv T_Y \times T_X$ , so  $D_A$  has approximately  $4HW$  nodes.

For each node of  $D_A$ , we compute a hash as follows. Let  $H_0$  be a collision-resistant hash-function. Set

$$h_r = \begin{cases} H_0(0||A[a, b]) & \text{if } r = (\{a\}, \{b\}) \\ H_0(1||h_{r^T}||h_{r^B}) & \text{if } r = ([a, b], \{c\}), a \neq b \\ H_0(1||h_{r^R}||h_{r^L}) & \text{if } r = (\{a\}, [b, c]), b \neq c \\ H_0(1||h_{r^T}||h_{r^R}||h_{r^B}||h_{r^L}) & \text{otherwise} \end{cases} \quad (4)$$

Finally, set  $H(A) = h_{r_A}$ .

*Security.* Most security properties of Merkle hash trees carry over to multi-dimensional Merkle hashing because we can convert a Merkle DAG into a Merkle tree by “exploding” the DAG as follows. Let  $D_A$  be a Merkle DAG on an  $H \times W$  matrix  $A$ . In this case, every leaf of  $D_A$  is at height  $\log HW$  and each node has at most 4 children, so there are at most  $4^{\log HW} = (HW)^2$  paths from the root to leaves. Let  $P$  be the set of paths in  $D_A$ , let  $A[p]$  be the element of  $A$  at the end of a path  $p$ . Each path  $p$  has a corresponding word  $w_p \in \{T, B, L, R\}^*$ , and we can sort the paths according to the lexicographic ordering of their corresponding words. This induces a vector  $x$  of length at most  $(HW)^2$ , indexed by paths in  $P$ , and defined by  $x[p] = A[p]$ . The paths in  $P$  give rise to a tree structure over  $x$ , and performing Merkle hashing over this tree structure will yield the same result as in the original DAG over  $A$ .<sup>2</sup> Thus, for example, if an attacker can find a hash collision using multi-dimensional Merkle hashing, he can immediately convert this into a collision using standard Merkle hashing.

---

<sup>2</sup> Technically, this tree structure is not a Merkle tree since many of the internal nodes have 4 children. This doesn't affect the security of Merkle hashing, and we could eliminate this wrinkle by introducing suitable intermediate nodes in the DAG.

```

procedure 2d-hash ( $A, r, s, C, M, O$ )
  if  $r \in \text{domain}(M)$ 
     $h := M[r]$ 
  else if  $r \in \text{domain}(C)$ 
     $h := C[r]$ 
  else if  $r = (\{a\}, \{c\})$ 
     $h := H_0(0||A[a, c])$ 
  else
     $t := \epsilon$ 
     $x := s$ 
    for  $i = 1, \dots, 4$ 
      if  $r^x$  exists
         $t := t || 2\text{d-hash}(A, r^x, x, C, M, O)$ 
         $x := \text{next}(x)$ 
       $h := H_0(1||t)$ 
    if  $r \notin \text{domain}(C)$ 
       $C[r] := h$ 
    else
      delete  $C[r]$ 
    if  $r \in \text{domain}(O)$ 
       $O[r] := h$ 
    return  $h$ 

procedure witness-set( $R, \hat{r}, \hat{w}, r'$ )
  if  $r' = \emptyset$ 
    return  $\{R\}$ 
  if  $r' = R$ 
    return  $\emptyset$ 
  if  $r'$  spans  $R$  in the  $x$  or  $y$  direction
    Pick  $c, c' \in \text{children}(R)$  such that  $c$  and  $c'$  span  $R$  in the same direction as  $r'$ 
  else if  $\hat{r} = R$ 
    Pick  $c, c' \in R$  children( $R$ ) such that  $c \cup c' = R$ 
  else
    Pick  $\{c, c'\} = \text{children}(R) \setminus \hat{w}$ 
  return  $(\text{children}(R) \setminus \{c, c'\}) \cup$ 
         witness-set( $c, c \cap \hat{r}, \hat{w}, c \cap r'$ )  $\cup$ 
         witness-set( $c', c' \cap \hat{r}, \hat{w}, c' \cap r'$ )

procedure next( $T = R$ )
  next( $R$ ) =  $B$ 
  next( $B$ ) =  $L$ 
  next( $L$ ) =  $T$ 

```

**Fig. 2.** The 2d-hash and witness-set algorithms. Note that, for 2d-hash,  $C$  and  $O$  are passed by reference. The “next” procedure just defines a clockwise ordering of  $T$ ,  $R$ ,  $B$ , and  $L$ .

*Efficiency.* A memoized recursive algorithm, such as the 2d-hash procedure shown in Figure 2, can compute all these hashes in  $O(HW)$  time because there are only  $4HW$  hashes to compute and each hash can be computed from its children in constant time. In this procedure,  $C$  is the memoization cache of previously computed hashes, which we assume is passed by reference. The mappings  $M$  and  $O$  will be used to perform croppings but, for now, assume that they both have empty domains. The following theorem proves that the algorithm is also memory-efficient.

**Theorem 1.** *If  $\text{domain}(M)$  is empty, then, during the execution of 2d-hash on a region of size  $H \times W$ , the cache  $C$  never contains more than  $4 \min(H, W) + 3 \max(H, W)$  elements.*

*Proof.* See companion technical report[8].

The algorithm can further reduce memory usage by not caching regions that span the original input range. These regions only have one parent in the DAG, so they do not need to be cached. Finally, we have implemented this algorithm and verified experimentally that it’s performance is consistent with the theorem.

*Witness Sets.* Suppose that  $A'$  is a submatrix of  $A$  covering subregion  $r' \subseteq r_A$ . Here,  $r'$  can be any subregion of  $r_A$ , not just the regions present in  $D_A$ . We can construct a proof that  $A'$  was part of the data used to compute  $h_{r_A}$  by revealing the hash values at appropriate nodes in  $D_A$ . For example, if  $A'$  is contained

entirely in the bottom half of  $A$ , then we can reveal the hashes  $h_{r^T}$ ,  $h_{r^R}$ , and  $h_{r^L}$ . This then reduces the problem to proving that  $A'$  was part of the data used to compute  $h_{r^B}$ . The witness-set procedure in Figure 2 computes the set of nodes whose hashes must be revealed to prove that the data in some region  $r'$  was used in the hash computation for data in region  $r$ . The following theorem bounds the size of the set of revealed witnesses.

**Theorem 2.** *Suppose region  $r$  is an  $H \times W$  rectangle, region  $r' \subseteq r$  is an  $h \times w$  rectangle, and  $O = \text{witness-set}(r, r')$ . Then  $|O| \leq 12 \log HW \log hw$ .*

*Proof.* See companion technical report[8].

The bound in Theorem 2 is rather conservative, and our experiments in Section 8 show that, on average, this theorem over-estimates the number of witnesses by a factor of 3. Furthermore, we can make witness sets even smaller by introducing intermediate nodes in the hash DAG. For example, if we change the definition of the hash to

$$h_r = \begin{cases} H_0(0||A[a, b]) & \text{if } r = (\{a\}, \{b\}) \\ H_0(1||h_{r^T}||h_{r^B}) & \text{if } r = ([a, b], \{c\}), a \neq b \\ H_0(1||h_{r^R}||h_{r^L}) & \text{if } r = (\{a\}, [b, c]), b \neq c \\ H_0(1||H_1(h_{r^T}||h_{r^R})||H_1(h_{r^B}||h_{r^L})) & \text{otherwise} \end{cases} \quad (5)$$

then witness-set only has to give away one or two witnesses at each step instead of two or three. Overall, this reduces the constants in the above analysis to  $8 \log HW \log hw$ .

Theorem 2 assumes that the submatrix  $A'$  is rectangular, but it is possible to construct a set of witnesses for a subregion of any shape. We have not attempted to analyze the size of the witness sets that would be required.

It may sometimes be desirable to take a witness set for a submatrix  $A'$  of  $A$  and construct another witness set for a submatrix  $A''$  of  $A'$ . In the realm of digital photographs, this would correspond to cropping an already cropped photograph. The next theorem confirms that it is always possible to construct a small witness-set for  $A''$  from  $A'$  and its witness set.

**Theorem 3.** *Let  $A'', A'$ , and  $A$  be matrices covering regions  $r'' \subseteq r' \subseteq r$ , respectively. For every witness set  $O = \text{witness-set}(r, r')$ , there exists a witness set  $O' = \text{witness-set}(r, r'')$  such that all the witnesses in  $O'$  can be computed from  $A'$  and the witnesses specified by  $O$ .*

*Proof.* See companion technical report[8].

Since  $O'$  is selected by the witness-set procedure, it must be small, as established in Theorem 2.

## 5 PRNGs for Croppable Signatures

In this section we describe two methods for generating a matrix of random mask values for use in a multi-dimensional croppable signature scheme. Recall

that the method from Section 3 built a standard 1-dimensional GGM tree and then organized its leaves into a matrix using row-major order. When used in a cropable signature scheme, the signature for an  $h \times w$  submatrix of a signed  $H \times W$  matrix must include  $\Omega(h \log w)$  values from the GGM tree. Our first improvement uses the same technique with a better space-filling curve to achieve  $O(h + w)$  tree nodes in a signature. Since it is merely a re-arrangement of the outputs of the GGM tree, it is clearly just as secure as the original scheme. Our second method sacrifices some security to substantially reduce the number of seed values that must be included in a signature. With our second scheme, signatures need to include only  $O(\log hw)$  seeds.

*Morton-order PRNG.* Let  $r$  be an  $H \times W$  region, and let  $r^T$ ,  $r^B$ ,  $r^L$ , and  $r^R$  be the top, bottom, left, and right halves of  $r$ , as in the previous section. We label nodes in the GGM tree with strings from the language  $\{T, B, L, R\}^*$ . Given a random seed  $k_\epsilon$ , we generate GGM values recursively using the formula

$$G(k_w) = \begin{cases} (k_{wL}, k_{wR}) & \text{if } r^w \text{ is wider than it is tall} \\ (k_{wT}, k_{wB}) & \text{otherwise} \end{cases} \quad (6)$$

Note that if  $r^w$  corresponds to a region of size 1, then  $k_w$  is a leaf of this tree. Each leaf value  $k_w$  is mapped to location  $r^w$  in the final output matrix. Since this algorithm is simply a binary GGM tree, it generates the same values as the original solution in Section 3, but arranges them in the final output matrix using a variant of the Morton-order space-filling curve[18]. The following well-known theorem, originally in the context of quad-trees[25], establishes the  $O(h + w)$  bound promised above.

**Theorem 4.** *Let  $r$  be an  $H \times W$  region, let  $R$  be the set of regions corresponding to nodes in the Morton-order GGM tree defined above, and let  $r' \subseteq r$  be any  $h \times w$  subregion of  $r$ . There exists a set of at most  $4(h + w)$  disjoint regions  $r_1, \dots, r_m \in R$  such that  $r' = \bigcup_{i=1}^m r_i$ .*

Unfortunately,  $4(w + h)$  can be quite large for digital photograph applications. For example, a subimage of a 16MP photograph could require about  $2^{15}$  GGM tree values. Using 16-byte seeds, this would create a signature over  $500KB$  in size.

*Intersecting PRNGs.* Our alternative solution is much more efficient, but sacrifices security. Pick a random seed  $k_\epsilon$ , set  $(x_\epsilon, y_\epsilon) = G(k_\epsilon)$ , and use normal GGM trees to generate  $x_0, \dots, x_{W-1}$  and  $y_0, \dots, y_{H-1}$ . Output  $i, j$  is simply  $x_i || y_j$ . This construction is not a PRNG, but it is now easy to reveal the outputs in any region  $r'$ : reveal the  $x$  outputs spanning  $r'$  horizontally and the  $y$  outputs spanning  $r'$  vertically. All total, this only reveals  $\log w + \log h = \log hw$  tree nodes. The price of this efficiency is that an adversary that obtains the outputs of this generator in two different regions,  $r_1 = ([a_1, b_1], [c_1, d_1])$  and  $r_2 = ([a_2, b_2], [c_2, d_2])$ , can combine the seeds to learn the outputs on two other regions:  $([a_1, b_1], [c_2, d_2])$  and  $([a_2, b_2], [c_1, d_1])$ . Nonetheless, we can use this scheme to build a cropable signature scheme that is secure against an adversary that only makes one query to the signing oracle, as proven in the next section.

## 6 Cropping-Homomorphic Signatures

We can now build two different cropping-homomorphic signature schemes.

*Morton-curve-based signature scheme.* To sign a matrix  $A$  corresponding to region  $r$ , the signer picks a random seed  $k_\epsilon$  and executes the following algorithm:

1. Use a GGM tree and Morton-curve to generate a matrix of pseudo-random values,  $k[i, j]$ .
2. Set  $v[i, j] = H(0, k[i, j], A[i, j])$  and use multi-dimensional Merkle hashing to compute a hash,  $v_\epsilon$  of the matrix  $v$ .
3. Output  $\text{Sig}_M(A) = (r, k_\epsilon, \text{Sig}_0(v_\epsilon))$ .

As explained in previous sections, a cropper can construct a signature on a submatrix  $A'$  of size  $h \times w$  by deleting  $k_\epsilon$  from the signature of  $A$  and including  $4(w + h)$  GGM values and  $O(\log HW + (\log hw)^2)$  hash values. The signature should also specify the location,  $r'$ , of  $A'$  within  $A$ . Thus, the format of a cropped signature is  $(r, r', \{k_w\}, \{v_w\}, \text{Sig}_0(v_\epsilon))$ .

The definition of security for cropping-homomorphic signatures is directly analogous to the definition of security for redactable signatures. The next theorem is analogous to the security theorem for redactable signatures.

**Theorem 5.** *Let  $G : \{0, 1\}^m \rightarrow \{0, 1\}^{2m}$  be a  $(t, \epsilon_G)$ -secure PRNG,  $H$  a  $(t, p_H)$ -collision-resistant hash function, and  $\text{Sig}_0$  a signature scheme  $(t, q, p_S)$ -secure against existential forgeries. Suppose also that  $H(0, k, \cdot)$  is a  $(t, q, \epsilon_H)$ -secure PRF (indexed by  $k$ ). Then the cropping-homomorphic signature scheme  $\text{Sig}_M$  defined above, when used to sign matrices of size at most  $H \times W$ , is  $(t', q, p')$ -secure against existential forgeries, where  $t' \approx t$  and  $p' = p_S + p_H + qHW\epsilon_G + qHW\epsilon_H + qHW2^{-m}$ .*

*Proof.* See companion technical report[8].

*Intersection-based signature scheme.* To reduce the number of PRNG seeds that must be revealed when cropping a matrix, we can replace the Morton-curve construction with the PRNG intersection scheme presented in Section 5, yielding a new signature scheme which we call  $\text{Sig}_I$ . This scheme is otherwise identical to the one above. This change reduces the number of seeds in a cropped signature from  $O(h+w)$  to  $O(\log hw)$ , but reduces security, as the following theorem makes explicit.

**Theorem 6.** *Let  $G$  and  $\text{Sig}_0$  be as in Theorem 5 and assume that  $H$  is a  $(t, p_H)$ -collision-resistant hash function. Suppose also that  $H(0, x||y, \cdot)$  is a  $(t, q, \epsilon_H)$  PRF indexed by  $x$  and a  $(t, q, \epsilon_H)$  PRF indexed by  $y$ . In other words, if the attacker gets to choose one of  $x$  and  $y$  and the other is chosen randomly, the attacker cannot distinguish  $H(0, x||y, \cdot)$  from a random function. Let  $\mathcal{A}$  be an adversary that makes at most one query  $S(i, \cdot)$  for each  $i$ . Then the probability that  $\mathcal{A}$  successfully constructs an existential forgery against  $\text{Sig}_I$  is at most  $p_S + p_H + q(H + W)\epsilon_G + qHW\epsilon_H + qHW2^{-m}$ .*

*Proof.* See companion technical report[8].

## 7 Other Homomorphic Signatures for Photographs

The croppable signature scheme above is already useful in the context of digital photographs, but we can use it as a foundation for building other homomorphic signature schemes.

*Scaling-homomorphic signatures.* Scaling and cropping are connected via the Discrete Cosine Transform (DCT)[26], so we can use this to immediately convert any cropping-homomorphic signature scheme into a scaling-homomorphic scheme. These two operations are related by the equation

$$\text{Scale}_{h \times w}^{H \times W} = \text{Clamp}_0^{255} \circ \sqrt{\frac{hw}{HW}} \circ \text{DCT}^{-1} \circ \text{Crop}_{h \times w} \circ \text{DCT} \quad (7)$$

where  $\text{Scale}_{h \times w}^{H \times W}$  scales an  $H \times W$  image down to an  $h \times w$  image,  $\text{Clamp}$  constrains values to be in the range of pixel values (typically 0 to 255),  $\sqrt{\frac{hw}{HW}}$  is a normalization factor,  $\text{Crop}_{h \times w}$  crops a matrix to only include its upper left  $h \times w$  submatrix, and  $\text{DCT}$  is the Discrete Cosine Transform. Thus, for any cropping-homomorphic signature scheme  $\text{Sig}_C$ ,  $\text{Sig}_S(I) = \text{Sig}_C(\text{DCT}(I))$  is a scaling-homomorphic signature scheme.

The Clamp operation and  $\sqrt{\frac{hw}{HW}}$  normalization introduce a slight wrinkle in the scheme. After scaling  $I$  to  $I'$  using the above algorithm, we can construct a new signature  $s'$  on  $\text{Crop}(\text{DCT}(I))$ , so a signature verifier must be able to compute  $\text{Crop}(\text{DCT}(I))$  in order to verify the signature. Since the Clamp operator and the rounding performed in the normalization by  $\sqrt{\frac{hw}{HW}}$  are both non-invertible, it is not possible to reconstruct  $\text{Crop}(\text{DCT}(I))$  from  $I' = \text{Clamp} \circ \sqrt{\frac{hw}{HW}} \circ \text{DCT}^{-1} \circ \text{Crop} \circ \text{DCT}(I)$ . Thus, after scaling  $I$ , we must store and transmit  $I'$  as  $D' = \text{Crop}(\text{DCT}(I))$ . The final recipient of an image will have to compute  $I' = \text{Clamp} \circ \sqrt{\frac{hw}{HW}} \circ \text{DCT}^{-1}(D')$  before displaying it.

*Scaling- and cropping-homomorphic signatures.* Let  $\text{Sig}_C$  be a 4-dimensional croppable signature scheme. Although we have only presented 2-dimensional constructions in this paper, they all generalize easily to higher-dimensions, so we know that such a signature scheme exists under standard cryptographic assumptions. To sign an  $H \times W$  image  $I$ , divide  $I$  into  $B_1 \times B_2$  blocks, each of size  $\frac{H}{B_1} \times \frac{W}{B_2}$ , creating a 4-dimensional array  $B[i, j, k, \ell]$ , where indices  $i$  and  $j$  select a block and  $k$  and  $\ell$  select a pixel within that block. Compute the DCT of each block separately, creating a new array  $D[i, j] = \text{DCT}(B[i, j])$ . Let  $\text{Sig}_{CS}(I) = \text{Sig}_C(D)$ . As with the scaling-homomorphic scheme, we must store and transmit  $D$  instead of the original image.

Now consider the different cropping operations we can perform on  $D$ . Cropping in the  $i$  or  $j$  dimensions crops entire rows or columns of blocks of  $D$ , which corresponds to cropping entire rows or columns of blocks of  $B$ , which corresponds to cropping the original image  $I$  by a multiple of the block size. Cropping  $D$  in the

$k$  or  $\ell$  dimension scales each block  $B[i, j]$  by the same amount, and scaling each block separately is equivalent to scaling the image by a multiple of the number of blocks. Thus, with some granularity, we can scale and crop the original image while preserving the signature. We recommend a block size of  $\sqrt{H} \times \sqrt{W}$ . For typical digital photographs, this would yield blocks of size about  $32 \times 32$  or  $64 \times 64$ .

Curiously, cropping the image actually *improves* the granularity of subsequent scaling operations, and vice versa. To see why, consider cropping the image down to single block. We can now scale the size of that block with perfect granularity. Conversely, if we scale the image down so that each block contains exactly a single pixel, then we can crop to any desired size.

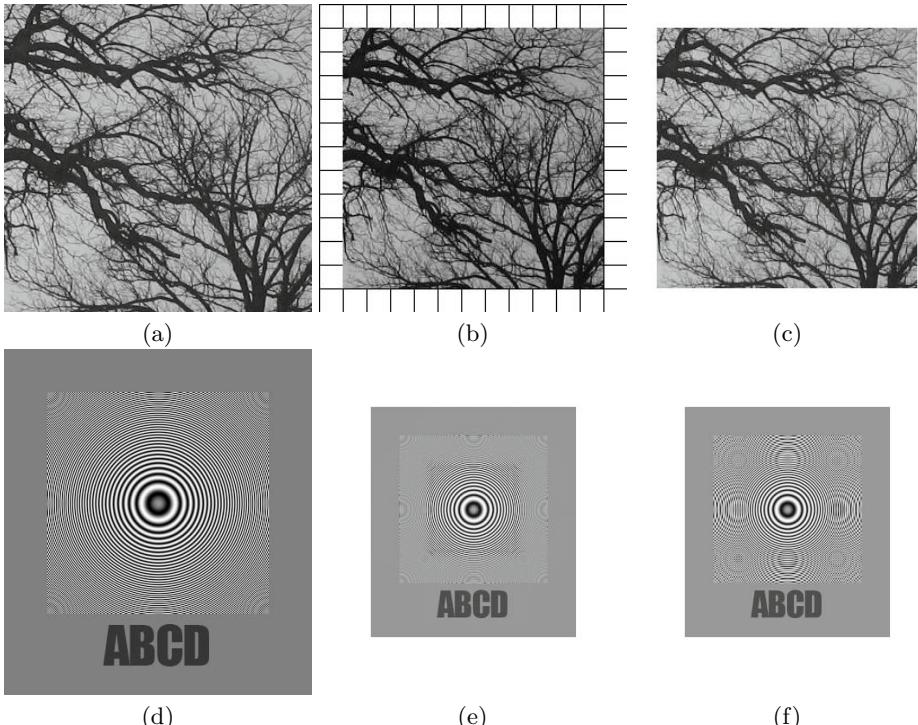
We can simplify this signature scheme by sacrificing some scaling power. Suppose we wish to support only aspect-ratio preserving scalings. In this case, each (square) block of  $D$  can only be cropped to a square sub-block, so we can re-order the coefficients in each block into a one-dimensional array, with the last-to-be-cropped coefficients at the beginning of the array and the first-to-be-cropped coefficients at the end of the array. This reduces  $D$  to a 3-dimensional array, so we can build such a cropping- and scaling-homomorphic signature scheme from a 3-dimensional cropping-homomorphic scheme.

JPEG compresses images by dividing them into  $8 \times 8$  pixel blocks, computing the DCT on each block, and quantizing the DCT coefficients. Thus the scaling and cropping homomorphic signature scheme would mesh well with a generalized version of JPEG that supports arbitrary block sizes.

To verify that the proposed scaling algorithm produces acceptable results, we scaled the image in Figure 3 using our algorithm and with the scaler implemented in The Gimp 2.6.10[31]. We scaled the image from  $352 \times 352$  pixels down to  $231 \times 231$  pixels by dividing it into blocks of size  $32 \times 32$  and then scaling each block to  $21 \times 21$  pixels using the algorithm described in the previous section. Figure 3(a) shows the original image, and Figure 3(b) shows our scaled image, set on a  $21 \times 21$  pixel grid to aid in identifying the boundaries between blocks. The DCT scaled version is visually indistinguishable from the Gimp scaled version in Figure 3(c) and does not exhibit any obvious artifacts at block boundaries, even though the image contains many high-frequency details. Our algorithm is not perfect, though. Figures 3(d-f) show that, in extreme cases, other scaling algorithms may generate substantially better results.

*JPEG-like compression.* JPEG applies a block-by-block DCT transform, as we do in the scalable and cropable scheme above, although JPEG always uses  $8 \times 8$  blocks. The only lossy step performed in JPEG compression is “quantization”, in which the coefficients in each block are divided by a constant to reduce the number of bits required to represent them, although with a corresponding loss of precision. Note that JPEG enables different quantization factors for each coefficient.

We can support a limited form of quantization quite simply by dividing the coefficients in  $D$  into their individual bits, turning  $D$  into a 5-dimensional array  $D[i, j, k, \ell, b]$ , where index  $b$  selects the desired bit of coefficient  $k, \ell$  of block  $i, j$ . Cropping away the  $c$  least-significant bits quantizes all the coefficients by a factor of  $2^c$ . Compressing different coefficients by different quantization factors,

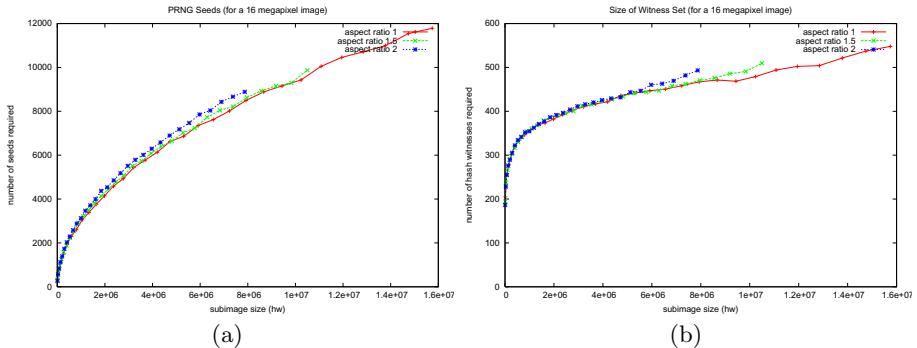


**Fig. 3.** A comparison of our scaling algorithm to the scaling algorithm in The Gimp. (We advise the reader to view these images on a printed version of this manuscript or with a high level of zoom, since we have noticed that many document viewers introduce their own artifacts when scaling these images for presentation on the screen.) (a) An original photograph[4]. (b) The photograph scaled with our algorithm using  $32 \times 32$  blocks. (c) The photograph scaled using The Gimp. (d) An artificial test image[2]. (e) The test image after scaling with our algorithm. (f) The test image scaled with The Gimp.

a feature crucial to good JPEG compression, is possible with the croppable signature schemes of Section 6, but the resulting signatures will be larger. Even with this extension, though, quantization factors must be powers of 2.

## 8 Experimental Results

Figure 4 plots the average seed set size and hash witness set size for a subimage cropped from a  $4096 \times 4096$  digital photograph versus the number of pixels in the subimage. These figures are based on a 2-dimensional hashing scheme that only supports cropping. For a randomly selected  $h \times w$  subimage, the expected seed set size is approximately  $3\sqrt{hw}$ . The size of witness sets is much more efficient than predicted, averaging less than  $4(\log hw)^2$  instead of the predicted



**Fig. 4.** Average (a) witness set and (b) seed set sizes for subimages of different sizes, all from 16MP digital photographs

$12 \log HW \log hw$ . In practice, this means that the size of a signature will be dominated by the size of the seed set. Signature size is also independent of the aspect ratio of the cropped subimage.

Using 256-bit hash values and 128-bit PRNG seeds, the largest signature we expect to see will be around 210KB, which only occurs when the cropped subimage is 15 megapixels. In this case, the JPEG representation of the subimage would be over 2MB, giving a signature overhead of 10%.

Our hashing scheme requires computing many small hashes. On a 2.8GHz Pentium, the OpenSSL implementation of SHA-256 can perform about 275,000 hashes per second. Hashing a 16MP image requires 64 million hash operations, which would take just under 4 minutes. A digital camera implementing our signature scheme could include specialized hardware for performing hashes, and could also perform them offline while otherwise idle.

## 9 Conclusion

We have presented new hashing and PRNG constructions for building efficient signature schemes that are homomorphic with respect to cropping of multi-dimensional data. We then presented several useful applications to authenticating digital photographs and other media, including signature schemes that are simultaneously homomorphic with respect to cropping, scaling, and JPEG-like compression. This research leaves several open questions. Is there a provably secure matrix PRNG construction that admits small seed-sets in cropable signatures? If we had four one-way functions,  $f_T$ ,  $f_B$ ,  $f_R$ , and  $f_L$  such that  $f_T$  and  $f_B$  commute with  $f_R$  and  $f_L$ , but not with each other, then we could construct such a PRNG by letting  $A_w = f_{w_1}(f_{w_2}(\dots(f_{w_n}(r))))$ . Commuting one-way functions may exist, for example, two instances of the RSA function with the same modulus but different exponents will commute, but we have not found a secure, working scheme based on this idea. Is there a generic way to construct signature schemes that are simultaneously homomorphic with respect to two different document

operations? For digital photographs, is there a croppable and scalable signature scheme with perfect granularity? What about other important, generally benign image operations, such as sharpen, brighten, and contrast adjustments?

## References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Chan, G.: Scaling artifacts and resolution,  
<http://www.glenncchan.info/broadcast-monitors/scaling-artifacts/scaling-artifacts.htm>
3. Chen, W., Shi, Y.Q., Xuan, G.: Identifying computer graphics using hsv color model and statistical moments of characteristic functions. In: Proceedings of the 2007 IEEE International Conference on Multimedia (2007)
4. Daily, G.: Sickle's witness tree (November 2008),  
<http://www.gettysburgdaily.com/?p=856>
5. David, A.: Trusted digital camera, <http://andrew.triumf.ca/trustcam> (viewed 2/6/2006)
6. Farid, H.: Detecting digital forgeries using bispectral analysis. Technical report, MIT (1999)
7. Farid, H., Lyu, S.: Higher-order wavelet statistics and their application to digital forensics. In: Proceedings of the 2003 IEEE Workshop on Statistical Analysis in Computer Vision (2003)
8. Johnson, R., Walsh, L., Lamb, M.: Homomorphic Signatures for Digital Photographs. Stony Brook University, TR-SB-CSE-SPLAT-2010-09 (2010)
9. Grosbois, R., Gerbelot, P., Ebrahimi, T.: Authentication and access control in the jpeg 2000 compressed domain. In: Proceedings of the 2001 SPIE Meeting no Applications of Digital Image Processing (2001)
10. Haggag, A., Yahagi, T., Lu, J.: Image authentication and integrity verification using jpsec protection tools. In: Proceedings of the First International Workshop on Image Media Quality and its Applications (2005)
11. Johnson, M., Farid, H.: Exposing digital forgeries by detecting inconsistencies in lighting. In: Proceedings of the 2005 ACM Multimedia and Security Workshop (2005)
12. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: RSA Cryptographer's Track (2002)
13. Kiltz, E., Mityagin, A., Panjwani, S., Raghavan, B.: Append-only signatures. In: Automata, Languages and Programming (2005)
14. Lin, C.-Y., Chang, S.-F.: Generating robust digital signature for image/video authentication. In: Multimedia and Security Workshop at ACM Multimedia, Bristol, UK (September 1998), <http://www.ctr.columbia.edu/~cylin/pub/acmmm98.ps>
15. Lin, C.Y., Chang, S.F.: Generating robust digital signature for image/video authentication. In: Proceedings of Multimedia and Security Workshop at ACM Multimedia (September 1998)
16. Lukas, J., Fridrich, J., Goljan, M.: Detecting digital image forgeries using sensor pattern noise. In: Proceedings of the 2006 SPIE Meeting no Electronic Imaging and Photonics West (2006)
17. Micali, S., Rivest, R.: Transitive signature schemes. In: Proceedings of the 2002 RSA Conference (2002)

18. Morton, G.M.: A computer oriented geodetic data base; and a new technique in file sequencing. Technical report, IBM, Ottawa, Canada (1966)
19. Ng, T.T., Chang, S.F., Hsu, J., Xie, L., Tsui, M.P.: Physics-motivated features for distinguishing photographic images and computer graphics. In: Proceedings of the 2005 ACM Multimedia Conference (2005)
20. Peng, C., Deng, R., Wu, Y., Shao, W.: A flexible and scalable authentication scheme for jpeg2000 codestreams. ACM Multimedia, 433–441 (November 2003)
21. Popescu, A.C., Farid, H.: Statistical Tools for Digital Forensics. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 128–147. Springer, Heidelberg (2004)
22. Popescu, A., Farid, H.: Exposing digital forgeries by detecting traces of re-sampling. IEEE Transactions on Signal Processing 55(2), 758–767 (2005)
23. CNET Reviews. Canon EOS-1Ds Digital SLR (February 2006),  
[http://reviews.cnet.com/Canon\\_EOS1Ds\\_Digital\\_SLR/4514-6501\\_7-20610303.html](http://reviews.cnet.com/Canon_EOS1Ds_Digital_SLR/4514-6501_7-20610303.html) (viewed 2/6/2006)
24. Rivest, R.: Two new signature schemes. Cambridge Seminar Series (2001)
25. Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley (1989)
26. Skodras, A.N., Christopoulos, C.A.: Down-sampling of compressed images in the dct domain. In: Proceedings of the 8th European Signal Processing Conference (September 1998)
27. Steinfeld, R., Bull, L., Zheng, Y.: Content Extraction Signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 205–285. Springer, Heidelberg (2002)
28. Sun, Q., Chang, S.-F.: A secure and robust digital signature scheme for jpeg2000 image authentication. IEEE Transactions on Multimedia 7(3), 480–494 (2005)
29. Sun, Q., Chang, S.F., Kurato, M., Suto, M.: A crypto signature scheme for image authentication over wireless channel. In: Proceedings of the 2004 IEEE International Conference on Multimedia (2004)
30. Sun, Q., Chang, S.F., Kurato, M., Suto, M.: A quantitative semi-fragile jpeg 2000 image authentication system. In: Proceedings of the 2002 International Conference on Image Processing (2002)
31. The Gimp Team. Gimp: The gnu image manipulation program (May 2011),  
<http://www.gimp.org/>
32. Wade, N.: Journal to examine how it reviewed articles. New York Times (January 2006)
33. Wee, S., Apostolopoulos, J.: Secure transcoding with jpsec confidentiality and authentication. Technical report, Hewlett-Packard (2004)
34. Winslow, D.R.: Reuters apologizes over altered lebanon war photos; suspends photographer. National Press Photographers Association (August 2006)
35. Wu, J., Zhu, B.B., Li, S., Lin, F.: A secure image authentication algorithm with pixel-level tamper localization. In: Proceedings of the 2004 IEEE International Conference on Image Processing (2004)
36. Wu, J., Zhu, B.B., Li, S., Lin, F.: New attacks on sari image authentication system. In: Proceedings of the 6th SPIE Security, Steganography, and Watermarking of Multimedia Conference (2004)
37. Yu, H.H.: Scalable multimedia authentication. In: Proceedings of the Fourth Pacific Rim Conference on Multimedia (December 2003)
38. Zhang, Z., Qiu, G., Sun, Q., Lin, X., Ni, Z., Shi, Y.Q.: A unified authentication framework for jpeg 2000. In: Proceedings of the 2004 IEEE International Conference on Multimedia (2004)
39. Zhu, B.B., Swanson, M.D., Li, S.: Encryption and authentication for scalable multimedia: Current state of the art and challenges. In: Proceedings of the 5th SPIE Internet Multimedia Management Systems Conference (2004)

# Revisiting the Computational Practicality of Private Information Retrieval<sup>\*</sup>

Femi Olumofin and Ian Goldberg

Cheriton School of Computer Science  
University of Waterloo Waterloo, ON, Canada N2L 3G1  
`{fgolumof,iang}@cs.uwaterloo.ca`

**Abstract.** Remote servers need search terms from the user to complete retrieval requests. However, keeping the search terms private or confidential without undermining the server’s ability to retrieve the desired information is a problem that private information retrieval (PIR) schemes are designed to address. A study of the computational practicality of PIR by Sion and Carbunar in 2007 concluded that no existing construction is as efficient as *the trivial PIR scheme* — the server transferring its entire database to the client. While often cited as evidence that PIR is impractical, that paper did not examine multi-server information-theoretic PIR schemes or recent single-server lattice-based PIR schemes. In this paper, we report on a performance analysis of a single-server lattice-based scheme by Aguilar-Melchor and Gaborit, as well as two multi-server information-theoretic PIR schemes by Chor et al. and by Goldberg. Using analytical and experimental techniques, we find the end-to-end response times of these schemes to be *one to three orders of magnitude* (10–1000 times) smaller than the trivial scheme for realistic computation power and network bandwidth. Our results extend and clarify the conclusions of Sion and Carbunar for multi-server PIR schemes and single-server PIR schemes that do not rely heavily on number theory.

## 1 Introduction

The retrieval of information from a remote database server typically demands providing the server with clues in the form of data indices, search keywords, or structured queries to assist with the retrieval task. However, keeping retrieval clues private without undermining the server’s ability to retrieve the desired information is a requirement that is common for user-centric privacy-preserving systems. Private information retrieval (PIR) provides a means of retrieval that guarantees access privacy, by preventing the database administrator from being able to learn any information about which particular item was retrieved.

Today’s most developed and deployed privacy-preserving techniques, such as onion routers and mix networks, are limited to anonymizing the identity of users. PIR, on the other hand, by protecting the *contents* of queries, can protect important application domains like patent databases, pharmaceutical databases,

---

\* An extended version of this paper is available [25].

online censuses, real-time stock quotes, location-based services, online behavioral analysis for ad networks, and Internet domain registration [3,14,18].

Chor et al., in defining the notion of PIR, proved that *the trivial PIR scheme* of transferring the entire database to the user and having him retrieve the desired item locally has optimal communication complexity for information-theoretic privacy protection with a single server. [7] However, more efficient information-theoretic solutions with sub-linear communication complexity were shown to exist if multiple, non-colluding servers hold copies of the database. They proposed a number of such multi-server information-theoretic PIR schemes [7], including a simple  $\ell$ -server scheme transferring  $O(\sqrt{n})$  bits, where  $n$  is the size of the database in bits and  $\ell \geq 2$  is the number of servers. Subsequent work has mostly focused on improving PIR's communication complexity bounds [7], while some others [3,13,16] have addressed such problems as using amortization and preprocessing to reduce server-side computational overheads and improving query robustness, amongst others.

Chor and Gilboa [8] were the first to relax the absolute privacy offered by multi-server information-theoretic PIR by using cryptographic primitives. They proposed a family of 2-server computationally private PIR schemes by making intractability assumptions on the existence of pseudorandom generators or one-way functions. Schemes in this family have a worst-case communication complexity of  $O(n^\epsilon)$ , for every  $\epsilon > 0$ . In the same year (1997), Kushilevitz and Ostrovsky [19] proposed the first single-server PIR scheme with a similar communication complexity by assuming quadratic residuosity decisions modulo a composite of unknown factorization are hard. Thus, the best protection offered by any non-trivial single-server PIR scheme is computational privacy, but database replication is not required. Several other single-server PIR schemes followed, each making some intractability assumption [2,6,20].

In 2007, Sion and Carbunar [28] considered the practicality of single-server computational PIR schemes and concluded that PIR would likely remain several orders of magnitude slower than an entire database transfer — the trivial PIR scheme — for past, current, and future commodity general-purpose hardware and networks. They based their result on the cheaper cost of transferring one bit of data compared to the cost of PIR-processing that bit using modular multiplication on such hardware. The PIR scheme of Kushilevitz and Ostrovsky, which was used in their comparison, requires one modular multiplication per database bit. They projected future increases in computing performance and network bandwidth using Moore's Law [21] and Nielsen's Law [23] respectively, and argued that improvements in computing performance would not result in significant improvements in the processing speed of PIR because of the need to use larger key sizes to maintain security. The significance of this work lies in establishing that any computational PIR scheme that requires one or more modular multiplications per database bit cannot be as efficient as the trivial PIR scheme.

However, it is not clear whether the conclusions of Sion and Carbunar [28] also apply to multi-server PIR schemes as well as single-server PIR schemes that do not rely heavily on number theory (i.e., modular multiplications). This is an

important clarification to make because PIR-processing with most multi-server PIR schemes and some single-server PIR schemes [2,29] costs much less than one modular multiplication per database bit. Besides, the projections from [28] assume that all PIR schemes make intractability assumptions that would necessitate the use of larger keys when today’s hardware and networks improve. However, multi-server PIR schemes offering information-theoretic privacy will continue to guarantee security and privacy without requiring key size changes irrespective of these improvements.

In this paper, we revisit the computational practicality of PIR in general by extending and clarifying the results in [28]. First, we provide a detailed performance analysis of a recent single-server PIR scheme by Aguilar-Melchor and Gaborit [1,2], which has attempted to reduce the cost of processing each database bit by using cheaper operations than modular multiplications. Unlike previous schemes that rely heavily on number theory, this particular scheme is based on linear algebra, and in particular, lattices. The authors introduced and based the security of the scheme on the differential hidden lattice problem, which they show is related to NP-complete coding theory problems [31]. They proposed and implemented the protocols, but their analysis was limited to server-side computations by the PIR server [1] on a small experimental database consisting of twelve 3 MB files. It is unclear how well the scheme compares against the trivial PIR scheme for realistic database sizes. Using the PIR scheme of Kushilevitz and Ostrovsky and updated parameters from [28], we first reestablished the result by Sion and Carbunar that this scheme is an order of magnitude more costly than the trivial PIR scheme. We also provide a new result that shows that the single-server PIR scheme in [2] offers an order of magnitude smaller response time compared to the trivial scheme, thus extending the conclusions of Sion and Carbunar about computational PIR schemes.

Second, we explore the case of multi-server information-theoretic PIR, which is yet to be considered by any previous study. Considering multi-server PIR is important because such schemes do not require costly modular arithmetic, and hence will benefit immensely from advances in computing and network trends. We derive upper-bound expressions for query round-trip response times for two multi-server information-theoretic PIR schemes by Chor et al. [7] and by Goldberg [16], which is novel to this paper. Through analytical and experimental techniques we find that the end-to-end response times of multi-server PIR schemes to be two to three orders of magnitude (100–1000 times) smaller than the trivial scheme for realistic computation powers and network bandwidths.

## 1.1 Preliminaries

We begin by outlining a few building blocks, some of which are based on [28]. These include the hardware, network bandwidth between the user and the server, and execution time estimates for modular multiplication.

**Hardware description.** All but one of our experiments were performed on current server hardware with two quad-core 2.50 GHz Intel Xeon E5420 CPUs,

**Table 1.** Bandwidth estimates (in Mbps) for 1995 to 2010. We adapted values up to 2007 from [28] and those after 2007 are based on the Internet speed data for Canada and US from [26].

Network types	1995	1997	1998	1999	2001	2005	2006	2007	2008	2009	2010
End-user( $B_1$ )	.028	.056		.768	1	4	6	6	6	8	9
Ethernet LAN( $B_2$ )	10	100		1000		10000	10000	10000	10000	10000	10000
Commercial( $B_3$ )	.256	.768		1	10	100	1000	1500	1500	1500	1500

32 GB of 667 MHz DDR2 memory, 6144 KB cache per core, an Adaptec 51645 RAID controller with 16 1.5TB SATA disks, and running Ubuntu Linux 9.10. The memory bandwidth is 21.344 GB/s and the disk bandwidth is at least 300 MB/s. We note that these machine characteristics are not unusual for database server hardware; this machine cost less than \$8,000. We ran the GPU implementation of the scheme in [2] on a machine with a Tesla C1060 GPU, 8 GB RAM, 116 MB/s disk bandwidth, and running Ubuntu Linux 9.10.

**Network.** Three types of network setups were considered [28]: average home-user last-mile connection, Ethernet LAN, and commercial high-end inter-site connections. Table 1 shows various network connection speeds (Mbps) since 1995, when PIR was introduced. The values up until 2006 are reused from [28], while we provided the subsequent values based on the capacity of today’s network bandwidths.

**Modular multiplication.** The work in [28] uses Dhrystone MIPS ratings for Pentium 4 CPUs in order to estimate  $t_{mul}$ , the time it takes to compute a modular multiplication — the building block for the PIR scheme of Kushilevitz and Ostrovsky [19]. Such CPUs have long been retired by Intel and are no longer representative of today’s multi-core CPUs. In addition, the Dhrystone benchmark, which found widespread usage at the time it was introduced in 1984, is now outdated. According to Dhrystone benchmark author Reinhold P. Weicker, it can no longer be relied upon as a representative benchmark for modern CPUs and workloads [30].

Instead, we measure the time directly. Using the key size schedule from NIST [22], the current recommended key size for the security of the Kushilevitz and Ostrovsky scheme is 1536 bits. We experimentally measured the value of  $t_{mul}$  on the server hardware described above. After repeated runs of the measurement code and averaging, we obtained  $t_{mul} = 3.08 \pm 0.08 \mu\text{s}$ .

**Projections.** Moore’s Law [21] has an annual growth rate of 60%, which surpasses the 50% growth rate of Nielsen’s Law [23]. While the faster growth rate of computing capabilities does not necessarily favour computational single-server PIR schemes, it does favour multi-server information-theoretic PIR schemes.

## 2 Related Work

The literature has mainly focused on improving the communication complexity of PIR schemes because communication between the user and the server(s) is

considered to be the most expensive resource [4]. Despite achieving this goal, other barriers continue to limit realistic deployment of PIR schemes; the most limiting of these barriers is the computational requirement of PIR schemes. The performance measure of a scheme in terms of its computational complexity has only received attention much more recently. The first of these is the work by Beimel et al. [4] which shows that, given an  $n$ -bit database  $X$  that is organized into  $r$   $b$ -bit blocks, standard PIR schemes cannot avoid a computation cost that is *linear* in the database size because each query for block  $X_i$  must necessarily process all database blocks  $X_j$ ,  $j \in \{1, \dots, r\}$ . They introduced a model of PIR with preprocessing which requires each database to precompute and store some *extra bits* of information, which is polynomial in the number of bits  $n$  of the database, before a PIR scheme is run the first time. Subsequently, the databases can respond to users' queries in a less computationally expensive manner using the extra bits. Asonov et al. [3] similarly explores preprocessing with a secure coprocessor for reducing server-side computation. However, the specialized hardware requirement at the server makes this solution less desirable.

In 2006, panelists from SECURECOMM [10] came together to discuss how to achieve practical private information retrieval. The discussion covers several aspects of transitioning cryptographic primitives from theory to practice and the need for practical PIR implementations and benchmarks on real data. The panelists were optimistic about future PIR deployments and pointed to the need for finding PIR schemes that require cheaper operations or utilize secure hardware.

The paper by Sion and Carbunar [28] compares the bandwidth cost of trivial PIR to the computation and bandwidth cost of a single-server computational PIR scheme [19], which they considered to be the most efficient at that time. The motivation of [28] was to stimulate practical PIR schemes; nevertheless, the result has been cited in the literature to promote the general idea that non-trivial PIR is always more costly than trivial download. Our work extends the work from [28] in important ways. First, their analysis was based on a number-theoretic computational PIR scheme [19], whereas we considered different varieties of computational PIR schemes: a number-theoretic scheme [19] and a lattice-based linear algebra scheme [2]. A consideration of the state of the art PIR schemes on the basis of their underlying mathematical assumptions is important because computational performance is currently the most mitigating factor to the practicality of PIR schemes. Secondly, we extend the analysis of practicality to multi-server PIR schemes which has never been considered by any previous measurement study. Multi-server PIR schemes are especially important because they can offer a stronger privacy guarantee for non-colluding servers, unlike computational PIR schemes that require large keys to protect against future powerful adversaries. Besides, multi-server PIR schemes give better performance and are directly deployable in domains where the databases are naturally distributed, such as Internet domain name registration [24]. Even in domains where the database is not distributed, deployment is possible using servers containing random data [13], which eliminates the need for an organization to replicate its data to foreign servers.

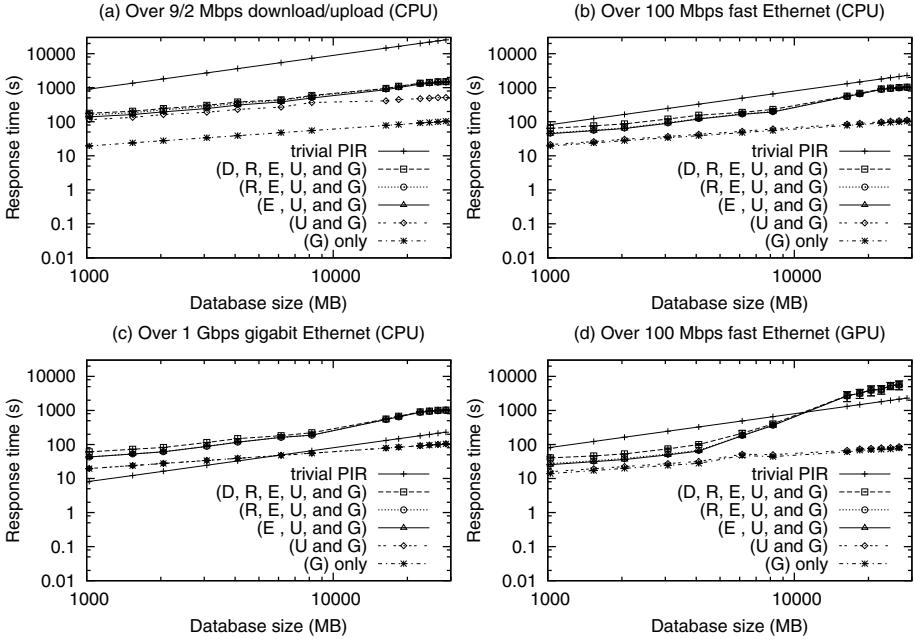
Aguilar-Melchor and Gaborit [2,1] explore linear algebra techniques using lattices to propose an efficient single-server PIR scheme. The security of the scheme is based on the hardness of the differential hidden lattice problem — a problem related to NP-complete coding theory problems [31]. Aguilar-Melchor et al. [1] subsequently used commodity Graphics Processing Units (GPUs), which are highly parallelizable, to achieve a database processing rate of 2 Gb/s, which is about ten times faster than running the same PIR scheme on CPUs. That work makes two main contributions. First, it shows that its scheme exhibits one order of magnitude speedup by using GPUs instead of CPUs to do the bulk of the computation, and claims that other schemes will see the same speedup. Second, it shows that in GPU-based scenarios, linear algebra based single-server PIR schemes can be more efficient than trivial download for most realistic bandwidth situations; this attempts to dispel the conclusions by Sion and Carbunar [28] with respect to the practicality of single-server PIR schemes. However, the evaluation from Aguilar-Melchor et al. [1] consider a small experimental database consisting of twelve 3 MB files and they did not measure the total roundtrip response time for queries; they considered the server-side cost but ignored client-side computation and transfer costs. It is important to consider the total cost because their scheme is not as efficient in terms of communication complexity as other existing schemes, and roundtrip response time depends on both the communication and computational complexities of a scheme. In addition, the measurements for the single-server PIR schemes [12,20] used for their comparison was based on estimates derived from *openssl speed rsa*, which is quite unlike our approach where the comparison is based on analytical expressions for query response times and experimental observations. Besides, they only considered single-server PIR schemes, whereas we also consider multi-server PIR schemes and the state-of-the-art single-server PIR schemes.

In the context of keyword search using PIR, Yoshida et al. [32] considered the practicality of a scheme proposed by Boneh et al. [5]. This public key encryption based keyword search protocol is essentially single-server PIR. Their investigations found the scheme to be costlier than the trivial PIR solution.

### 3 Efficient Single-Server PIR (LPIR-A)

We experimentally evaluated an implementation of the single-server PIR scheme by Aguilar-Melchor et al. [1]. This is the most efficient known single-server PIR scheme, and has available source code both for CPUs and GPUs. We present a note of caution, however, that although this PIR scheme resists known lattice-based attacks, it is still relatively new, and its security is not as well understood as those of the PIR schemes that rely heavily on number theory.

We obtained the source code [17] for this scheme, removed interactivity, changed the default parameters to one that guarantees security in a practical setting (complexity of over  $2^{100}$  operations) [2], and added instrumentation to the CPU and GPU code variants. The data set for our experiment consists of various databases of sizes between 1 GB and 28 GB, each containing random



**Fig. 1.** Logarithmic scale plots for query generation (G), query upload(U), response encoding (E), response download (R), and response decoding (D) times for the single-server PIR scheme [1] and the trivial PIR scheme in different bandwidth scenarios

data. Bugs in the implementation [17] prevented us from testing larger databases for the selected security parameters.

We ran queries to retrieve between 5 and 10 random blocks for each database size.

Figure 1 shows the log-log plots of our results with breakdowns of the time for query generation and upload, response encoding and download, response decoding, as well as the trivial download time for the different sizes of databases we tested. Plots (a), (b), (c), and (d) respectively reflect bandwidth values typical of an Internet connection in the US and Canada, a 100 Mbps fast Ethernet, a 1 Gbps gigabit Ethernet, and a 100 Mbps fast Ethernet on the GPU hardware.

In plot (a), for example, the largest portion of the overall time is that of query upload; this is due to the comparatively low 2 Mbps upload bandwidth typical of a home Internet connection [26]. On the other hand, the time to download the query result (at 9 Mbps) is much smaller. In general, the response time is proportional to  $n$  and the slope of the line is 1, as the computation costs, in particular server-side response encoding, dominate. When the database exceeds the available RAM size, further slowdowns are seen in the results.

The slope of the trivial PIR line is always 1, since the time is simply that of transferring the entire database. For small databases, the trivial PIR scheme is faster, but depending on the bandwidth, there is a crossover point at which

sending less data plus computing on every bit of the database becomes faster than sending the entire database. For the average home connection, for example, we found this to occur at a very small database size (approximately 32 MB). For the 1 Gbps connection, the network is so fast that the entire database can be transferred in less time than it takes for the client to even generate its query, except for databases of 6 GB and larger. Even then, trivial transfer was much faster than the overall cost of this PIR scheme for such fast networks.

We note that plot (a) is the most representative of today's consumer bandwidth situation. Based on the recently available Internet speed database [26], the average bandwidth for the Internet user is improving rather slowly, with average download rates of 6, 7.79, and 9.23 Mbps for Canada and the US for 2008, 2009, and January 1 to May 30 of 2010. The average upload rates for the respective periods are 1.07, 1.69, and 1.94 Mbps. We note that Nielsen's Law specifically addresses the type of users described as normal "high-end" who can afford to pay a premium for high-bandwidth network connections [23]. We contrast these users from "low-end" users [23] that the above bandwidth averages from the Internet speed data [26] include. Hence, the majority of Internet users are low-end users, and their bandwidth is much more limited than that predicted by Nielsen's Law.

In the plots and in the analysis above, we show changing bandwidths and assume that computing power stays the same. However, if we assume that processors improve at a faster rate than Internet bandwidth for high-end users due to Moore's Law and Nielsen's Law, then the crossover point will move down and the PIR scheme will become faster at smaller database sizes. From plot (d), the GPU run gives a better response time, in comparison to plot (b), for memory-bound databases (about 6 GB or less). For disk-bound databases, the response time degenerates due to the lower disk bandwidth of the GPU machine. We ran the same code on the CPU of the GPU hardware; using the GPU, we found about five times speedup in the server-side processing rate for memory-bound databases and no noticeable speedup for disk-bound databases. Our observed speedup is half the speedup reported in [1], but we used much larger databases.

## 4 Multi-server PIR

In this section, we provide detailed performance analyses of two multi-server information-theoretic PIR schemes, from Chor et al. [7] and from Goldberg [16]. We begin with an overview of these schemes and later show how they compare with the single server schemes [2,19] and the trivial PIR scheme. The reason for choosing [7] is its simplicity, being the first PIR protocol invented. The reason for choosing [16] is its comprehensiveness and source code availability which allows for easy experimental analysis. The implementation of [16], known as Percy++ [15], is an open-source project on SourceForge.

In order to maintain the user's privacy, it must be the case that not all (in the case of the Chor et al. protocol) or at most a configurable threshold number (in the case of the Goldberg protocol) of the database servers collude to unmask the user's query. This is sometimes brought forward as a problematic requirement

of these schemes. We note that, as discussed elsewhere [24], there are reasonable scenarios — such as distributed databases like DNS or whois databases, where the copies of the database may be held by competing parties — in which the non-collusion requirement is acceptable. Further, other privacy-enhancing technologies, such as anonymous remailers [9] and Tor [11], also make the assumption that not all of the servers involved are colluding against the user.

#### 4.1 First Scheme (MPIR-C)

We first describe the simple  $O(\sqrt{n})$  protocol by Chor et al. The database  $D$  is treated as an  $r \times b$  matrix of bits, where the  $k^{\text{th}}$  row of  $D$  is the  $k^{\text{th}}$  block of the database. Each of  $\ell$  servers stores a copy of  $D$ . The client, interested in block  $i$  of the database, picks  $\ell$  random bitstrings  $\rho_1, \dots, \rho_\ell$ , each of length  $r$ , such that  $\rho_1 \oplus \dots \oplus \rho_\ell = e_i$ , where  $e_i$  is the string of length  $r$  which is 0 everywhere except at position  $i$ , where it is 1. The client sends  $\rho_j$  to server  $j$  for each  $j$ . Server  $j$  computes  $R_j = \rho_j \cdot D$ , which is the XOR of those blocks  $k$  in the database for which the  $k^{\text{th}}$  bit of  $\rho_j$  is 1, and sends  $R_j$  back to the client. The client computes  $R_1 \oplus \dots \oplus R_\ell = (\rho_1 \oplus \dots \oplus \rho_\ell) \cdot D = e_i \cdot D$ , which is the  $i^{\text{th}}$  block of the database.

Sion and Carbunar [28] used a closed-form expression for the computation and communication cost of the PIR scheme in [19]. While we derive similar expressions for the multi-server schemes we studied, we note that it will only approximate the cost because most modern x86 CPUs support hardware-level parallelism such as superscalar operations; single-cycle operations, such as XORs, are parallelized even within a single core. Hence, such expressions can be used to determine an *upper bound* on what response time to expect. We will later determine the exact response time for this PIR scheme through experiments.

For optimal performance, we set  $r = b = \sqrt{n}$ . Hence, the upper bound for the client and server execution times for this protocol can respectively be computed as  $2(\ell - 1)\frac{\sqrt{n}}{m}t_{\oplus} + 2\ell\sqrt{n}t_t$  and  $\frac{n}{m} \cdot (t_{\oplus} + 2t_{ac}) + n \cdot t_{ov}$ , where  $t_{\oplus}$  and  $t_t$  are respectively the execution times for one XOR operation and the transfer time for one bit of data between the client and the server;  $m$  is the machine word-size (e.g., 64 bits),  $n$  is the database size (in bits),  $\ell$  is the number of servers,  $t_{ov}$  represents the amortized server overhead per bit of the database — this overhead is dominated by disk access costs, but also includes things like the time to execute looping instructions — and  $t_{ac}$  denotes the time for one memory access. Note that the server execution time is the worst-case time because it assumes all the blocks in the database are XORed, whereas we only need to XOR blocks where the  $i^{\text{th}}$  bit of  $\rho_j$  is 1. The expression charges all of the data transfer to the client, since it needs to be serialized there, whereas the server processing is performed in parallel among the  $\ell$  servers.

An upper bound on the query round-trip execution time for this multi-server PIR scheme is then  $T_{MPIR-C} < (2(\ell - 1)\sqrt{n}/m + n/m) \cdot t_{\oplus} + 2\ell\sqrt{n} \cdot t_t + 2n/m \cdot t_{ac} + n \cdot t_{ov}$ . The most dominant term is  $n \cdot (\frac{1}{m}t_{\oplus} + \frac{2}{m}t_{ac} + t_{ov})$ , which will suffice for the entire expression when the value of  $n$  is large.

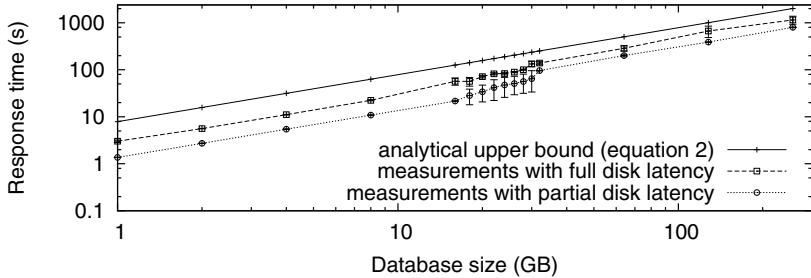
The work in [28] denoted  $t_t = \frac{1}{B}$ , given that  $B$  is the bandwidth (in bps) of the network connection between the client and the server.  $t_{\oplus}$  will be one cycle.

(We indeed measured it to be  $0.40 \pm 0.01$  ns, which is exactly as expected on our 2.50 GHz processor.) Similarly, we measured  $t_{ac}$  to be 1 cycle ( $0.4000 \pm .0003$  ns). Using unrolling to minimize the overhead of loop instructions,  $t_{ov}$  will be dominated by the memory bandwidth if the database fits into memory, or by disk bandwidth otherwise. An upper bound for  $t_{ov}$  on our test machine is therefore 0.006 ns for in-memory databases and 0.417 ns for disk-bound databases, based on the numbers in Section 1.1.

## 4.2 Second Scheme (MPIR-G)

Goldberg's scheme is similar to the Chor et al. scheme in its use of simple XOR operations to accomplish most of its server-side computations. However, it uses Shamir secret sharing [27] to split the user's query vector  $e_i$  into  $\ell$  shares which are then transmitted to the servers. The server database  $D$  is treated as an  $r \times b$  matrix of  $w$ -bit words (i.e., elements of  $GF(2^w)$ ), where again  $r$  is the number of blocks and  $b$  is the number of  $w$ -bit words per block. In addition, the elements of  $e_i$ ,  $\rho_j$ , and  $R_j$  are elements of  $GF(2^w)$ , instead of single bits. These changes are necessary because the protocol addresses query robustness for byzantine servers that may respond incorrectly or not respond at all. For simplicity, in this paper we will only consider honest servers, which respond correctly. For head-to-head comparison with the Chor et al. protocol, we set the privacy level  $t$  (the number of servers which can collude without revealing the client's query) to  $\ell - 1$ . As before, we choose  $r = b$ , but now  $r = b = \sqrt{n/w}$ . We also choose  $w = 8$  to simplify the cost of computations; in  $GF(2^8)$ , additions are XOR operations on bytes and multiplications are lookup operations into a 64 KB table. These are the choices made by the open-source implementation of this protocol [15].

A client encodes a query for database block  $i$  by first uniformly choosing  $\ell$  random distinct non-zero indices  $\alpha_1, \dots, \alpha_\ell$  from  $GF(2^8)$ . Next, the client chooses  $r$  polynomials of degree  $t$ , one for each block in  $D$ . The coefficients of the non-constant terms for polynomial  $f_k$  are random elements of  $GF(2^8)$ , while those for the constant terms should be 1 if  $i = k$  and 0 otherwise. Afterwards, the client hands out to each server  $j$  a vector  $\rho_j$  formed from evaluating all  $r$  polynomials at  $\alpha_j$ ; that is,  $\rho_j = [f_1(\alpha_j), \dots, f_r(\alpha_j)]$ . (Note that each  $f_k(\alpha_j)$  is an element of  $GF(2^8)$  — a single byte.) In a manner similar to the Chor et al. scheme, each server computes a response vector  $R_j = \rho_j \cdot D$ , where each of the  $b$  elements of vector  $R_j$  is also a single byte. The servers send  $R_j$  to the client and the client computes the query result using Lagrange interpolation, which also amounts to simple arithmetic in  $GF(2^8)$ . Using the protocol description in [16] and the source code [15], we counted each type of operation to derive upper bounds for the respective client and server execution times as  $\ell(\ell-1)\sqrt{n/8}(t_{\oplus} + t_{ac}) + 2\ell\sqrt{8nt_t} + 3\ell(\ell+1)(t_{\oplus} + t_{ac})$ , and  $(n/8)(t_{\oplus} + 3t_{ac}) + n \cdot t_{ov}$ , where the terms are as above. Again, note that we charge all of the communication to the client. The upper bound expression for the protocol's round-trip response time is then  $T_{MPIR-G} < ((\sqrt{n/8} + 3)\ell^2 - (\sqrt{n/8} - 3)\ell + n/8)(t_{\oplus} + 3t_{ac}) + 2\ell\sqrt{8n} \cdot t_t + n \cdot t_{ov}$ . Here, the dominant term is  $n \cdot (\frac{1}{8}(t_{\oplus} + 3t_{ac}) + t_{ov})$ .

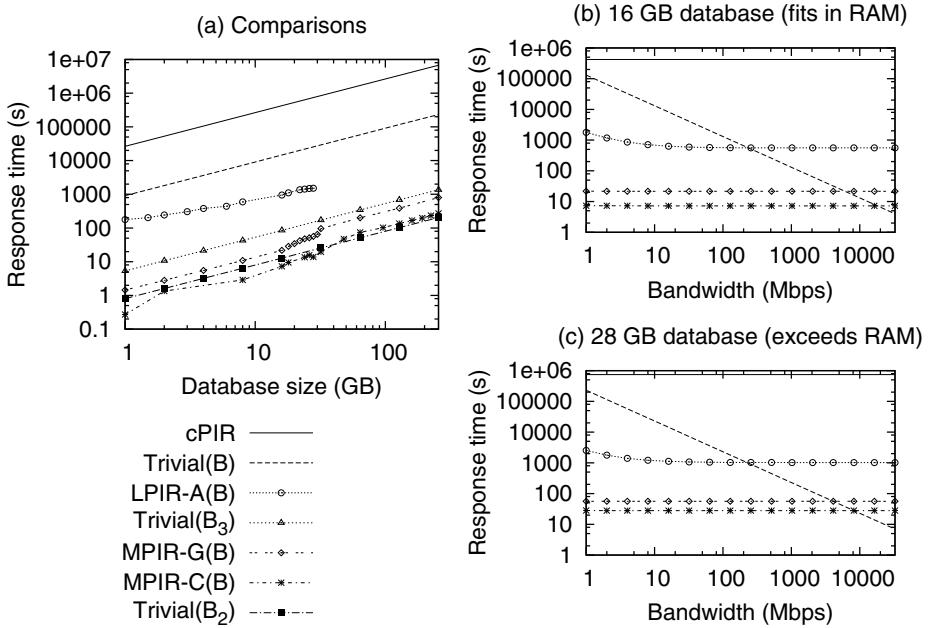


**Fig. 2.** Analytical and experimental measurements of the response time of Goldberg’s multi-server PIR scheme [16] (computations only). The upper line is derived from equation (2), but excluding time for communications. The middle line is the time for the first query, which includes startup overhead and reading the database from disk. The lower line is the time for subsequent queries, which only incur disk latencies once the database exceeds the available RAM size.

### 4.3 Response Time Measurement Experiment

We measure the round-trip response times for the multi-server PIR schemes in this section. We first modified an implementation of MPIR-G (Percy++) [15] to use wider data types to enable support for larger databases. We then measured its performance over five different sets of databases, with databases in each set containing random data and ranging in size from 1 GB to 256 GB.

Next, we fetched 5 to 10 blocks from the server. On the first query, the database needs to be loaded into memory. The server software does this with `mmap()`; the effect is that blocks are read from disk as needed. We expect that the time to satisfy the first query will thus be noticeably longer than for subsequent queries (at least for databases that fit into available memory), and indeed that is what we observe. For databases larger than available memory, we should not see as much of a difference between the first query and subsequent queries. We show in Figure 2 plots of the average response time with standard deviations for these two measurements (i.e., PIR response time for the first query, and for the second and subsequent queries). From the plot, the speed of 1.36 seconds per GB of data is consistent until the databases that are at least 16 GB in size are queried. Between 18 GB and 30 GB, the time per GB grew steadily until 32 GB. The threshold crossed at that range of database sizes is that the database size becomes larger than the available RAM (somewhat smaller than the total RAM size of 32 GB). As can be seen from the plot, the measured values for that range are especially noisy for the lower line. We designed our experiment to take measurements for more databases with size in that range; we surmise that the particulars of Linux’s page-replacement strategy contribute a large variance when the database size is very near the available memory size. For even larger databases, PIR query response times consistently averaged 3.1 seconds per GB of data. This is because every query now bears the overhead of reading from the disk. In realistic deployment scenarios where the database fits into available memory, the overhead of



**Fig. 3.** (a) Comparing the response times of PIR schemes by Kushilevitz and Ostrovsky (cPIR) [19], Aguilar-Melchor [1] (LPIR-A), Chor et al. [7] (MPIR-C), and Goldberg [16] (MPIR-G), as well as the trivial PIR scheme over three current network bandwidths data in Table 1, using different database sizes. The bandwidth used for the non-trivial PIR schemes is  $B$ . (b,c) Plots of response time vs. bandwidth for the PIR schemes as in (a) for database sizes that fit in RAM (16 GB) and exceed RAM (28 GB).

disk reads is irrelevant to individual queries and is easily apportioned as part of the server’s startup cost. Even when the database cannot fit in available memory, the bottleneck of disk read overheads could be somewhat mitigated by overlapping computation and disk reads; we did not implement this optimization because the current performance was sufficient for head-to-head comparison with the trivial solution. Note that in practice, the disk read latency would equally come into play even for trivial PIR.

We made similar measurements for the Chor et al. [7] MPIR-C scheme using an implementation we developed. The implementation differed from [15] by doing XORs in 64-bit words, instead of by bytes. We obtained a speed of 0.5 seconds per GB (sometimes as fast as 0.26 seconds per GB) for small databases that fit in available memory and 1.0 seconds per GB for larger databases.

## 5 Comparing the Trivial and Non-trivial PIR Schemes

We next compare the round-trip response rates for each of the PIR schemes already examined to the response rates of the trivial PIR scheme and the Kushilevitz

vitz and Ostrovsky [19] scheme. We note that for the non-trivial schemes, the amount of data transmitted is tiny compared to the size of the database, so the available bandwidth does not make much difference. To be as generous as possible to the trivial PIR scheme, we measure the non-trivial schemes with the home connection bandwidth  $B$  — 9 Mbps download and 2 Mbps upload. We provide comparisons to the trivial PIR scheme with bandwidths of  $B$ ,  $B_2$  — 10 Gbps Ethernet, and  $B_3$  — 1.5 Gbps inter-site connections (see Table 1).

Figure 3(a) shows the log-log plot of the response times for the multi-server and lattice-based PIR schemes against the earlier results from [28], which include the trivial scheme and the Kushilevitz and Ostrovsky scheme [19]. As in [28], we give maximal benefit to the scheme in [19] by ignoring all costs except those of modular multiplication for that scheme, using the value for  $t_{mul}$  given in Section 1.1. We point out that the values for the trivial scheme and the Kushilevitz and Ostrovsky scheme are computed lower bounds, while those for the LPIR-A, MPIR-G, and MPIR-C schemes are experimentally measured. The number of servers for the multi-server schemes is  $\ell = 2$ .

We can see from the plot that, as reported in [28], the trivial PIR scheme vastly outperforms the computational PIR scheme of Kushilevitz and Ostrovsky, even at the typical home bandwidth. However, at that bandwidth, the lattice-based scheme of Aguilar-Melchor et al. is over 10 times faster than the trivial scheme. Further, both multi-server schemes are faster than the trivial scheme, even at the  $B_3$  (1.5 Gbps) speeds; the MPIR-G scheme is about 4 times faster for databases that fit in RAM, and the MPIR-C scheme is over 10 times faster. For large databases, they are 1.7 and 5 times faster, respectively. Only at  $B_2$  Ethernet speeds of 10 Gbps does the trivial scheme beat the multi-server schemes, and even then, in-memory databases win for MPIR-C. The apparent advantage of the trivial scheme even at these very high bandwidths may, even so, be illusory, as we did not include the time to read the database from memory or disk in the trivial scheme’s lower-bound cost, but we did for the LPIR and MPIR schemes.

One might try rescuing the trivial PIR scheme by observing that, having downloaded the data *once*, the client can perform *many* queries on it at minimal extra cost. This may indeed be true in some scenarios. However, if client storage is limited (such as on smartphones), or if the data is updated frequently, or if the database server wishes to more closely control the number of queries to the database — a pay-per-download music store, for example — the trivial scheme loses this advantage, and possibly even the ability to be used at all.

To better see at what bandwidth the trivial scheme begins to outperform the others, we plot the response times vs. bandwidth for all five schemes in Figure 3(b,c). We include one plot for a database of 16 GB, which fits in RAM (a), and one for 28 GB, which does not (b). We see that the trivial scheme only outperforms LPIR-A at speeds above about 100 Mbps, and it outperforms the MPIR schemes only at speeds above 4 Gbps for large databases and above 8 Gbps for small databases. In addition, due to the faster growth rate of computing power as compared to network bandwidth, multi-server PIR schemes will become even faster over time relative to the trivial scheme, and that will increase the bandwidth crossover points for all database sizes.

## 6 Conclusions

We reexamined the computational practicality of PIR following the earlier work by Sion and Carbunar [28]. Some interpret [28] as saying that no PIR scheme can be more efficient than the trivial PIR scheme of transmitting the entire database. While this claim holds for the number-theoretic single-database PIR scheme in [19] because of its reliance on expensive modular multiplications, it does not hold for all PIR schemes. We performed an analysis of the recently proposed lattice-based PIR scheme by Aguilar-Melchor and Gaborit [2] to determine its comparative benefit over the trivial PIR scheme, and found this scheme to be an order of magnitude more efficient than trivial PIR for situations that are most representative of today's average consumer Internet bandwidth. Next, we considered two multi-server PIR schemes, using both analytical and experimental techniques. We found multi-server PIR to be a *further* one to two orders of magnitude more efficient. We conclude that many real-world situations that require privacy protection can obtain some insight from our work in deciding whether to use existing PIR schemes or the trivial download solution, based on their computing and networking constraints.

**Acknowledgments.** We would like to thank Gregory Zaverucha and the anonymous reviewers for their helpful comments for improving this paper. We also gratefully acknowledge NSERC and MITACS for funding this research.

## References

1. Aguilar Melchor, C., Crespin, B., Gaborit, P., Jolivet, V., Rousseau, P.: High-Speed Private Information Retrieval Computation on GPU. In: SECURWARE 2008, pp. 263–272 (2008)
2. Aguilar-Melchor, C., Gaborit, P.: A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol. In: WEWORC 2007 (July 2007)
3. Asonov, D. (ed.): Querying Databases Privately. LNCS, vol. 3128. Springer, Heidelberg (2004)
4. Beimel, A., Ishai, Y., Malkin, T.: Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing. *J. Cryptol.* 17(2), 125–151 (2004)
5. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith III, W.E.: Public Key Encryption that Allows PIR Queries. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 50–67. Springer, Heidelberg (2007)
6. Cachin, C., Micali, S., Stadler, M.A.: Computationally Private Information Retrieval with Polylogarithmic Communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
7. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: FOCS 1995: Proceedings of the 36th Annual Symposium on the Foundations of Computer Science, pp. 41–50 (October 1995)
8. Chor, B., Gilboa, N.: Computationally Private Information Retrieval (extended abstract). In: STOC 1997, pp. 304–313 (1997)
9. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a Type III Anonymous Remailer Protocol. In: IEEE S&P, pp. 2–15 (May 2003)

10. Di Crescenzo, G., et al.: Towards Practical Private Information Retrieval. Achieving Practical Private Information Retrieval Panel. In: SecureComm 2006 (August 2006), <http://www.cs.sunysb.edu/sion/research/PIR.Panel.Securecomm.2006/giovanni.pdf>
11. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
12. Gentry, C., Ramzan, Z.: Single-Database Private Information Retrieval with Constant Communication Rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005)
13. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting Data Privacy in Private Information Retrieval Schemes. In: STOC 1998, pp. 151–160 (1998)
14. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private Queries in Location Based Services: Anonymizers are not Necessary. In: SIGMOD 2008, pp. 121–132 (2008)
15. Goldberg, I.: Percy++ project on SourceForge, <http://percy.sourceforge.net/>
16. Goldberg, I.: Improving the Robustness of Private Information Retrieval. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 131–148 (2007)
17. GPGPU Team: High-speed PIR computation on GPU on Assembla, [http://www.assembla.com/spaces/pir\\_gpgpu/](http://www.assembla.com/spaces/pir_gpgpu/)
18. Juels, A.: Targeted Advertising... And Privacy Too. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 408–424. Springer, Heidelberg (2001)
19. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: FOCS 1997, p. 364 (1997)
20. Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)
21. Moore, G.E.: Cramming More Components Onto Integrated Circuits. Electronics Magazine 38(8) (1965)
22. National Institute of Standards and Technology: Key Management Guideline (2007), <http://csrc.nist.gov/groups/ST/toolkit/index.html>
23. Nielsen, J.: Nielsen's Law of Internet Bandwidth (April 1988), <http://www.useit.com/alertbox/980405.html>
24. Olumofin, F., Goldberg, I.: Privacy-Preserving Queries over Relational Databases. In: Atallah, M.J., Hopper, N.J. (eds.) PETs 2010. LNCS, vol. 6205, pp. 75–92. Springer, Heidelberg (2010)
25. Olumofin, F., Goldberg, I.: Revisiting the Computational Practicality of Private Information Retrieval. Tech. rep., CACR 2010-17, University of Waterloo (2010), <http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-17.pdf>
26. Ookla: Canada and US Source Data, <http://www.netindex.com/source-data/>
27. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
28. Sion, R., Carbunar, B.: On the Computational Practicality of Private Information Retrieval. In: NDSS 2007 (2007)
29. Trostle, J., Parrish, A.: Efficient Computationally Private Information Retrieval From Anonymity or Trapdoor Groups. ePrint 2007/392 (2007)
30. Weiss, A.R.: Dhrystone Benchmark: History, Analysis, Scores and Recommendations. EEMBC White Paper (2002)
31. Wieschebrink, C.: Two NP-complete Problems in Coding Theory with an Application in Code Based Cryptography. In: ISIT 2006, pp. 1733–1737 (July 2006)
32. Yoshida, R., Shigetomi, Y.C., Imai, R., The Practicality, H.: of the Keyword Search Using PIR. In: ISITA 2008, pp. 1–6 (December 2008)

# Optimal One Round Almost Perfectly Secure Message Transmission (Short Paper)

Mohammed Ashraful Alam Tuhin and Reihaneh Safavi-Naini

Department of Computer Science, University of Calgary  
`{maatuhin,rei}@ucalgary.ca`

**Abstract.** In this paper we consider 1-round almost perfectly secure message transmission protocols with optimal transmission rate, and give constructions for two levels of connectivity,  $n = 2t + k$  and  $n = (2 + c)t, c > \frac{1}{t}$ . Here  $n$  and  $t$  are the total number of wires and the number of corrupted wires, respectively, and  $k$  and  $c$  are constants. The first protocol has a modular construction that with appropriate instantiations of each module results in optimal protocols for  $n = 2t + k$ . The second protocol is the first construction for optimal protocol when  $n = (2 + c)t$ .

## 1 Introduction

The **Secure Message Transmission (SMT)** problem was introduced in [5] to address the problem of secure communication between two nodes in an incomplete network. In the SMT problem, the sender  $\mathcal{S}$  and the receiver  $\mathcal{R}$  do not share a key but are connected by  $n$  ‘wires’ where *at most*  $t$  of which are controlled by an adversary. Wires are abstractions of node-disjoint paths between  $\mathcal{S}$  and  $\mathcal{R}$ . Security means  $\mathcal{R}$  will receive the message sent by  $\mathcal{S}$  in a *private* and *reliable* way. The initial motivation for this model has been to reduce connectivity requirements in secure multi-party protocols [3]. In recent years SMT protocols have found other applications including key distribution in sensor networks [4].

**Motivation of our work.** It was shown [5] that one round protocols with perfect privacy and perfect reliability requires  $n \geq 3t + 1$ .  $(\varepsilon, \delta)$ -SMT ( $0 \leq \varepsilon, \delta \leq 1$ ) was defined in [8] where the loss of privacy and reliability is bounded by  $\varepsilon$  and  $\delta$ , respectively.  $(\varepsilon, \delta)$ -SMT protocols exist for  $n \geq 2t + 1$ . A perfectly private ( $\varepsilon = 0$ ) and  $\delta$ -reliable secure message transmission, denoted by  $(0, \delta)$ -SMT, is called an **Almost Perfectly Secure Message Transmission (APSMT, for short)**.

In this paper we consider the 1-round APSMT problem for different levels of network connectivity, starting from the minimum requirement of  $n = 2t + 1$  up to the case where  $n = 2t + k$ , and also when  $n = (2 + c)t$ , i.e., a constant fraction of wires are corrupted. These are motivated by real life applications of these protocols [4,12].

**Our Results.** We consider two types of connectivity. In the first case,  $n = 2t + k$ , where  $k \geq 1$  is a constant. In the second case,  $n = (2 + c)t$ , where  $c$  is a constant satisfying  $c > \frac{1}{t}$  and so, a *constant fraction* of wires are corrupted.

**1.** We first present a modular construction for 1-round APSMT protocol for  $n = 2t + k$  which consists of two modules. The first module  $(n, t, \delta)\text{-Send}$  is a protocol that is used to deliver with  $\delta$ -reliability, an information matrix of size  $(n - 2t) \times n$  (random elements chosen from a finite field, to the receiver such that the adversary can learn *at most* a sub-matrix of size  $(n - 2t) \times t$ . At the end of the protocol, sender and receiver share a sub-matrix of size  $(n - 2t) \times (n - t)$  which is completely unknown to the adversary. However, the sender and the receiver cannot determine the sub-matrix.

The second module is a privacy amplification (PA) protocol that extracts  $(n - t)$  elements that are completely unknown to the adversary, from a shared vector of size  $n$  which has at most  $t$  elements known by the adversary. We propose a new construction for the first module. For the second module, one can use the protocol described in [10]. We however adapt another existing PA technique which results in a computationally more efficient protocol. Using this approach, we show a construction for  $n = 2t + k$  that has linear (in  $n$ ) transmission rate that matches the lower bound on the transmission rate for 1-round APSMT protocol for this connectivity and so is optimal.

**2.** Next, we present a 1-round APSMT protocol for  $n = (2 + c)t$ , where  $c$  is a constant satisfying  $c > \frac{1}{t}$ . This protocol has *constant* transmission rate and is *optimal*. The protocol uses the two modules  $(n, t, \delta)\text{-Send}$  and  $PA(n, n - t)$  used in the first protocol. We also adapt an existing protocol as the third module, to send the ciphertexts (secrets encrypted with some one-time pads) with *constant* transmission rate.

Modular construction of SMT protocols introduced in this work allows construction of new SMT protocols from more basic primitives and is a promising approach for reusing existing primitives to construct new protocols.

**Related Work.** The lower bound on transmission rate for 1-round APSMT is  $\Omega(\frac{n}{n-2t})$  [10]. Protocols whose transmission rate asymptotically matches this bound are called *rate-optimal* (or *optimal*, for short). This means that for  $n = 2t + k$  and  $n = (2 + c)t$ , optimal protocols should have transmission rates  $O(n)$  and  $O(1)$ , respectively.

An optimal and efficient 1-round APSMT protocol for  $n = 2t + 1$  is given in [10]. All the other known 1-round APSMT protocols are either *not optimal* [7,6] or *computationally inefficient* [9]. There are also efficient APSMT protocols for  $n = 3t + 1$  [1,10], but the one in [1] is *not optimal*. Prior to this work, there is no known general construction for connectivity  $n = (2 + c)t$ , where  $c > \frac{1}{t}$ .

*Organization.* Section 2 gives definitions and notations. In Section 3 we present the 1-round APSMT protocol for  $n = 2t + k$  together with security and efficiency analysis and comparison with related work. In Section 4, we give our 1-round APSMT protocol for  $n = (2 + c)t$  and analyze its security and efficiency. In Section 5 we conclude our work.

## 2 Background

**Communication Model.** We consider a *synchronous, incomplete* network. The sender  $\mathcal{S}$  and the receiver  $\mathcal{R}$  are connected by  $n$  vertex-disjoint paths, also known as wires or channels. Both  $\mathcal{S}$  and  $\mathcal{R}$  are honest. The goal is for  $\mathcal{S}$  to send a message  $m$  to  $\mathcal{R}$  such that  $\mathcal{R}$  receives it correctly and privately. The wires are undirected and two-way. The protocol can have one or more rounds. In a round, a message is sent by either  $\mathcal{S}$  or  $\mathcal{R}$  to the other party over the wires. Messages are delivered to the recipient of the round before the next round starts.

**Adversary Model.** The adversary  $\mathcal{A}$  has *unlimited* computational power and corrupts a subset of nodes in the network. A path (wire) that includes a corrupted node is controlled by  $\mathcal{A}$ . Corrupted nodes can arbitrarily *eavesdrop, modify or block* messages sent over the corrupted wires.  $\mathcal{A}$  uses all the information obtained from the corrupted wires to choose and corrupt a new wire up to the threshold  $t$ .  $\mathcal{S}$  and  $\mathcal{R}$  do not know which wires are corrupted.

**Notation.**  $\mathcal{M}$  is the message space from which messages are chosen according to a probability distribution  $\Pr$ . Let  $M_{\mathcal{S}}$  be the message randomly selected by  $\mathcal{S}$ . We assume  $\mathcal{M}$  and  $\Pr$  are known in advance to all parties including the adversary. Let  $R_{\mathcal{A}}$  be the random coins used by  $\mathcal{A}$  to choose  $t$  out of total  $n$  wires to corrupt.

In an execution of an SMT protocol  $\Pi$ ,  $\mathcal{S}$  draws  $M_{\mathcal{S}}$  from  $\mathcal{M}$  using the distribution  $\Pr$  and aims to send it to  $\mathcal{R}$  privately and reliably. We assume that at the end of the protocol,  $\mathcal{R}$  outputs a message  $M_{\mathcal{R}} \in \mathcal{M}$  or ‘NULL’.

Let  $V_{\mathcal{A}}(M_{\mathcal{S}}, r_{\mathcal{A}})$  denotes the view of the adversary  $\mathcal{A}$  when  $\mathcal{S}$  has chosen  $M_{\mathcal{S}}$  and  $R_{\mathcal{A}} = r_{\mathcal{A}}$ . The view  $V_{\mathcal{A}}(M_{\mathcal{S}}, R_{\mathcal{A}})$  is a random variable that depends on the random coins of  $\mathcal{S}$  and  $\mathcal{R}$  and the choice of  $M_{\mathcal{S}}$ . We use  $V_{\mathcal{A}}(M_{\mathcal{S}} = m, r_{\mathcal{A}} = r)$  or  $V_{\mathcal{A}}(m, r)$ , for short, when  $r_{\mathcal{A}} = r$  and  $M_{\mathcal{S}} = m$ .

The *statistical distance* of two random variables  $X, Y$  over a set  $\mathcal{U}$  is defined as

$$\Delta(X, Y) = \frac{1}{2} \sum_{u \in \mathcal{U}} |\Pr[X = u] - \Pr[Y = u]|$$

**Definition 1.** [8] An SMT protocol is called an  $(\varepsilon, \delta)$ -Secure Message Transmission ( $(\varepsilon, \delta)$ -SMT) protocol if the following two conditions are satisfied:

- **Privacy:** For every two messages  $m_0, m_1 \in \mathcal{M}$  and every  $r \in \{0, 1\}^*$ ,

$$\Delta(V_{\mathcal{A}}(m_0, r), V_{\mathcal{A}}(m_1, r)) \leq \varepsilon,$$

where the probability is over the randomness of  $\mathcal{S}$  and  $\mathcal{R}$ .

- **Reliability:**  $\mathcal{R}$  receives the message  $M_{\mathcal{S}}$  with probability  $\geq 1 - \delta$ . That is,

$$\Pr[M_{\mathcal{R}} \neq M_{\mathcal{S}}] \leq \delta,$$

where the probability is over the randomness of all the players and the choice of  $M_{\mathcal{S}}$ .

When  $\varepsilon = 0$ , the protocol is said to achieve *perfect privacy* and when  $\delta = 0$ , the protocol is said to achieve *perfect reliability*.

The number of ***rounds*** of a protocol is the number of interactions between  $\mathcal{S}$  and  $\mathcal{R}$ . We consider synchronous network where time is divided into clock ticks and in each clock tick the sender or the receiver sends a message and the message is received by the other party before the next clock tick.

***Communication complexity*** is the total number of bits transmitted between  $\mathcal{S}$  and  $\mathcal{R}$  for communicating the message. Communication efficiency is often measured in terms of *transmission rate*, which is the ratio of the communication complexity to the length of the message  $m$ . That is,

$$\text{Transmission Rate} = \frac{\text{total number of bits transmitted}(\ell)}{\text{size of the secrets}(|m|)}$$

The message is one or more elements of an alphabet.

***Computation complexity*** is the amount of computation performed by  $\mathcal{S}$  and  $\mathcal{R}$  throughout the protocol. A protocol with *exponential* (in  $n$ ) computation is considered *inefficient*. Efficient protocols need *polynomial* (in  $n$ ) computation. The relationship between  $n$  and  $t$  required for the existence of an SMT protocol is referred to as the ***connectivity requirements*** of the SMT protocol.

**Bounds.** It was shown in [8] that APSMT is possible if and only if  $n \geq 2t + 1$ . Dolev *et al.* showed  $(1, 0)$ -SMT (PRMT) protocols are possible if  $n \geq 2t + 1$  [5].

Patra *et al.* showed that the lower bound on the transmission rate of 1-round APSMT protocol is given by  $\Omega(\frac{n}{n-2t})$  [10]. When  $n = 2t + k, k \geq 1$ , the lower bound on transmission rate becomes  $\Omega(n)$  and when  $n = (2 + c)t$ , where  $c > \frac{1}{t}$  is a constant, it becomes *constant*. SMT protocols that asymptotically achieve the above bounds, for respective connectivities, are called *optimal* with respect to the corresponding bound.

**1-round PRMT Protocol for  $n = (2 + c)t$ .** We now present a 1-round PRMT protocol  $\Pi_1$  for  $n = (2 + c)t, c > \frac{1}{t}$ . The main idea of this protocol is to use codewords of Reed-Solomon codes to send  $ct$  messages (each consisting of one field element) with perfect reliability by sending  $n$  field elements. The sender constructs a polynomial  $f(x)$  of degree at most  $(ct - 1)$  such that the  $ct$  coefficients of  $f(x)$  are the messages to be sent perfectly reliably. The sender then sends evaluations of  $f(x)$  on distinct points, each associated with a wire, through the corresponding wire. The minimum distance of the code is  $(2 + c)t - ct + 1 = 2t + 1$ , and so the receiver can correct  $t$  possible errors, reconstruct the polynomial, and recover the  $ct$  sent messages. This protocol can be seen as an adaptation of the protocol REL-SEND of [11].

### 3 1-Round Optimal APSMT Protocol for $n = 2t + k$

The construction consists of two modules. The first module is called  $(n, t, \delta)$ -Send. The second one is a non-interactive *privacy amplification* (PA) protocol,  $PA(n, n - t)$ .

### 3.1 $(n, t, \delta)$ -Send

This module constructs an input matrix  $R$  of size  $n \times n$ ,  $R = (r_{11}, \dots, r_{1n}, \dots, r_{n,1}, \dots, r_{n,n})$ , consisting of  $(n - 2t)n$  randomly chosen elements together with  $2tn$  elements that are computed from them, and delivers the matrix to the receiver as  $R'$  such that (i)  $\Pr(R = R') \geq 1 - \delta$ ,  $\delta < \frac{1}{2}$ , and (ii) at most a sub-matrix of size  $(n - 2t) \times t$  of  $R$  will be known to the adversary  $\mathcal{A}$ . Therefore, a sub-matrix of size  $(n - 2t) \times (n - t)$  will remain unknown to  $\mathcal{A}$ .

Our proposed  $(n, t, \delta)$ -Send module is shown in Fig. 1.

**Transmission:** Consider a sequence of  $(n - 2t)n$  random elements  $R = (r_{11}, \dots, r_{1n}, \dots, r_{n-2t,1}, \dots, r_{n-2t,n})$  as a matrix of size  $(n - 2t) \times n$ . The sender performs two steps as follows:

Step1 For each  $i, 1 \leq i \leq n - t$ :

Constructs a random polynomial  $p_i(x)$  of degree  $\leq (n - 1)$  such that  $p_i(x) = \sum_{j=0}^{n-1} a_{ij}x^j$ . Here  $a_{ij}$ 's are random elements from  $\mathbb{F}$ .

– For each  $i, 1 \leq i \leq n$ :

Forms a poly.  $q_i(x)$  of degree  $\leq (n - t - 1)$  such that the  $j^{th}$  coefficients of  $q_i(x)$  is  $a_{j,i-1}$ ,  $1 \leq j \leq n - t$ .

– Suppose  $r_{ij} = q_j(i), 1 \leq i \leq n - 2t, 1 \leq j \leq n$ .

– For each  $i, n - t + 1 \leq i \leq n$ :

Constructs a polynomial  $p_i(x) = \sum_{j=0}^{n-1} q_j(i)x^j$ .

Step 2 Randomly selects  $n^2$  field elements  $s_{ij}, 1 \leq i, j \leq n$  and constructs pairs  $(s_{ij}, p_i(s_{ij})), 1 \leq i, j \leq n$ .

– Sends  $p_i(x)$  through wire  $i$  and  $(s_{ij}, p_i(s_{ij}))$  through wire  $j$ , for  $1 \leq i, j \leq n$ .

**Recovery:** The receiver does the following. For each  $i, 1 \leq i \leq n$ :

Step 1 Receive  $p'_i(x)$  over wire  $i$ , and  $(s'_{ij}, v_{i,s_{ij}})$  through wire  $j$ , for  $1 \leq j \leq n$ .

Suppose  $a'_{ij}$  are the coefficients of the received polynomials.

– Compute  $k = |\{j : p'_i(s'_{ij}) \neq v_{i,s_{ij}}\}|$ .

– If  $k \geq t + 1$ , then decide wire  $i$  as corrupted and adds  $i$  to a list  $FAULTY$ .

Step 2 Suppose  $i_1, i_2, \dots, i_{n'}, n' \geq n - t$  are the indices of the wires  $\notin FAULTY$ . For each  $j, 1 \leq j \leq n$ , do the following:

\* Form a poly.  $q'_j(x)$  of degree  $\leq n - t - 1$  using  $a'_{i_1j}, \dots, a'_{i_{n-t}j}$  and verify whether  $q'_j(i_\ell) \neq p'_{i_\ell}(i_\ell)$ ,  $\ell > n - t$ . If there exists one such  $\ell$ , then output 'NULL' and terminate the protocol.

– Reconstruct the first  $(n - t)$  polynomials by considering any  $(n - t)$  polynomials carried by wires not in the list  $FAULTY$ .

– Recover the  $(n - 2t) \times n$  random elements in the same way as the sender.

**Fig. 1.**  $(n, t, \delta)$ -Send

**Theorem 1.** Module  $(n, t, \delta)$ -Send sends  $(n - 2t) \times n$  random elements so that all the  $(n - 2t) \times n$  will be received with probability  $1 - \delta$ , and the adversary can learn at most  $(n - 2t) \times t$  elements, while  $(n - 2t) \times (n - t)$  elements are completely unknown to the adversary. The total required communication is  $O(n^2 \log |\mathbb{F}|)$ .

**Proof. Perfect Privacy.** There are  $(n-t)$  independently generated random polynomials. The adversary sees *at most* any  $t$  polynomials and  $t$  points of any other polynomials. Since polynomials are of degree  $n-1$  then all  $n$  coefficients are independent and so in total  $(n-2t) \times (n-t)$  elements remain unknown to the adversary.

**$\delta$ -reliability.** Omitted due to lack of space.

**Efficiency.** The sender sends  $n$  polynomials, each of degree at most  $(n-1)$  through the  $n$  wires. This incurs a communication of  $n^2 \log |\mathbb{F}|$ . He also sends each of the  $n$  pair of values (evaluation points) through each wire, for all the polynomials. This needs a communication of  $2n^2 \log |\mathbb{F}|$ . Therefore, the total communication of this protocol is  $O(n^2 \log |\mathbb{F}|)$ . ■

### 3.2 Non-interactive Privacy Amplification for SMT

Privacy amplification allows the sender and the receiver to non-interactively generate  $a$  random elements which will be completely unknown to the adversary, from  $b > a$  random elements, where the adversary knows *at most*  $(b-a)$  elements. We use the protocol in [2] given in Figure 2.

---

$PA(b, b-a); a < b$ : input  $(x_1, \dots, x_b) \in \mathbb{F}^b$ ; output:  $(X_1, X_2, \dots, X_a) \in \mathbb{F}^a$

1. Forms a polynomial  $f(x)$  of degree  $\leq (b-1)$  such that  $f(i) = x_i, 1 \leq i \leq b$ .
  2. Outputs  $(f(b+1), \dots, f(b+a))$ .
- 

**Fig. 2.** The non-interactive Privacy Amplification Technique  $PA(b, b-a)$

### 3.3 Description of the Protocol

For simplicity we will describe the protocol for  $k=1$ , but can be easily used for larger  $k$ . Our 1-round APSMT protocol  $\Pi_2$  for  $n=2t+1$  is given in Fig. 3. The receiver in this protocol will never output incorrect message(s). He will either output the correct message(s) or output ‘NULL’.

---

The sender  $\mathcal{S}$  wishes to send  $n-t = (t+1)$  secrets  $m_0, m_1, \dots, m_t \in \mathbb{F}^{t+1}$  to the receiver  $\mathcal{R}$ . Since  $n=2t+1$ , here  $(n-2t)n=n$ .

Step 1. The sender  $\mathcal{S}$  does the following:

1. Calls  $(n, t, \delta)$ -Send to send  $n$  random elements  $r_i, 1 \leq i \leq n$  to the receiver.
2. Calls  $PA(n, n-t)$  with  $(r_1, r_2, \dots, r_n)$  as input and obtains  $(R_1, R_2, \dots, R_{t+1})$ .
3. Forms  $t+1$  ciphertexts as  $c_i = m_i \oplus R_i$ , and broadcasts  $c_i, 1 \leq i \leq t+1$ .

Step 2. The receiver does the following.

1. Receives the  $n$  random elements  $r_1, r_2, \dots, r_n$ .
  2. Calls  $PA(n, n-t)$  with  $(r_1, r_2, \dots, r_n)$  as input and obtains  $(R_1, R_2, \dots, R_{t+1})$
  3. Recovers the  $t+1$  messages as  $m_i = c_i \oplus R_i, 1 \leq i \leq t+1$ .
- 

**Fig. 3.** The 1-round APSMT protocol  $\Pi_2$  for  $n=2t+1$

**Security and Efficiency Analysis.** The protocol uses  $(n, t, \delta)$ -Send and  $PA(n, n-t)$ , followed by a step in which  $(t+1)$  ciphertexts are broadcasted. The broadcast can be seen as a third module with perfect reliability. The broadcast is also perfectly secure for messages because of the one-time-pad encryption used for messages. Reliability of the overall protocol directly follows from the reliability of  $(n, t, \delta)$ -Send and the final broadcast. Perfect security of the protocol follows from the perfect security of the one-time-pads that are generated through the application of  $PA(n, n-t)$  and Theorem 1.

The transmission rate of  $\Pi_2$  is  $O(n)$ . This is true because  $(n, t, \delta)$ -Send has communication cost of  $O(n^2 \log |\mathbb{F}|)$ . The protocol  $\Pi_2$  also broadcasts  $(t+1)$  ciphertexts with a communication cost of  $n(t+1) \log |\mathbb{F}|$ . Therefore, the total communication of this protocol is  $O(n^2 \log |\mathbb{F}|)$ . The protocol sends  $(t+1)$  messages of total size  $(t+1) \log |\mathbb{F}| = O(n \log |\mathbb{F}|)$  and so the transmission rate is  $O(n)$  which is optimal for a 1-round  $(0, \delta)$ -SMT protocol for  $n = 2t+1$ .

**Comparison.** The comparison with related work is outlined in Table 1.

**Table 1.** Comparison with 1-round APSMT protocols for  $n = 2t+1$  (here Comp. refers to computation complexity and  $q$  is the field size)

Author	Comp.	$\delta$	Optimality
Kurosawa and Suzuki [9]	Exp.	$\leq ((\binom{n}{t+1} - \binom{n-t}{t+1})\lambda^1$	Yes
Desmedt and Wang [7]	Poly.	$\leq \frac{n}{q}$	No
Patra <i>et al.</i> [10]	Poly.	$\leq \frac{n^3}{q}$	Yes
Desmedt <i>et al.</i> [6]	Poly.	$\leq \frac{t(t+1)}{q}$	No
This work	Poly.	$\leq \frac{n^2}{q}$	Yes

## 4 1-Round Optimal APSMT Protocol for $n = (2+c)t$

We present a 1-round APSMT protocol for  $n = (2+c)t$ , where  $c$  is a constant satisfying  $c > \frac{1}{t}$ . The protocol has *optimal* transmission rate. The protocol is designed by extending the protocol  $\Pi_2$  and using the 1-round PRMT protocol  $\Pi_1$  for  $n = (2+c)t, c > \frac{1}{t}$  showed in Section 2.

**Description of the protocol.** The protocol is given in Fig. 4. The receiver will either output the correct message(s) or output ‘NULL’.

**Perfect Privacy and  $\delta$ -Reliability.** Perfect privacy of  $\Pi_3$  follows from the perfect privacy of the first two modules and independent executions of each

<sup>1</sup> Here  $\lambda$  is the probability that the cheater win in a secret sharing scheme with a cheater.

---

The sender wishes to send  $(n - t)(n - 2t)$  secrets  $m_{11}, \dots, m_{1,n-t}, m_{21}, \dots, m_{2,n-t}, \dots, m_{n-2t,1}, \dots, m_{n-2t,n-t} \in \mathbb{F}^{(n-2t)(n-t)}$  to the receiver  $\mathcal{R}$ .

Step 1. The sender  $\mathcal{S}$  does the following:

1. Calls  $(n, t, \delta)$ -Send (communicate  $(n - 2t)n$  random elements  $r_{ij}, 1 \leq i \leq n - 2t, 1 \leq j \leq n$ ).
2. Calls  $PA(n, n - t)$ ,  $(n - 2t)$  times, for  $1 \leq i \leq n - 2t$  (use  $(r_{ij}, 1 \leq j \leq n)$  as input to obtain  $(n - t)$  random-elements  $(R_{i1}, \dots, R_{i,n-t})$ ).
3. Generates  $(n - 2t)(n - t)$  ciphertexts,  $c_{ij} = m_{ij} \oplus R_{ij}, 1 \leq i \leq n - 2t, 1 \leq j \leq n - t$  and send them by calling  $\Pi_1$ , in parallel,  $(n - t)$  times.

Step 2. The receiver does the following.

1. Receives the  $(n - 2t)n$  random elements  $r_{ij}, 1 \leq i \leq n - 2t, 1 \leq j \leq n$ .
  2. Calls  $PA(n, n - t)$ ,  $(n - 2t)$  times with  $(r_{ij}, 1 \leq j \leq n)$  as input to get  $(n - t)$  random-elements  $(R_{i1}, \dots, R_{i,n-t})$ , for  $1 \leq i \leq n - 2t$ .
  3. Receives the  $(n - 2t)(n - t)$  ciphertexts perfectly reliably and recover the  $(n - 2t)(n - t)$  secrets using  $(R_{11}, \dots, R_{1,n-t}, \dots, R_{n-2t,1}, \dots, R_{n-2t,n-t})$  as  $m_{ij} = c_{ij} \oplus R_{ij}, 1 \leq i \leq n - 2t, 1 \leq j \leq n - t$ .
- 

**Fig. 4.** The 1-round APSMT protocol  $\Pi_3$  for  $n = (2 + c)t$

invocation of the module  $PA(n, n - t)$ . The reliability of  $\Pi_3$  follows directly from the reliability of  $(n, t, \delta)$ -Send and that of  $\Pi_1$ . Therefore,  $\Pi_3$  is unreliable with probability at most  $\frac{n^2}{|\mathbb{F}|}$ .

**Efficiency.**  $(n, t, \delta)$ -Send needs a communication of  $O(n^2 \log |\mathbb{F}|)$ . The communication for using  $\Pi_1$  is  $n(n - 2t) \log |\mathbb{F}| = O(n^2 \log |\mathbb{F}|)$  bits. Therefore, the total communication of the protocol  $\Pi_3$  is  $O(n^2 \log |\mathbb{F}|)$ . The protocol sends  $(n - 2t)(n - t) = ct(t + ct)$  messages of total size  $ct(t + ct) \log |\mathbb{F}| = O(n^2 \log |\mathbb{F}|)$ , resulting in  $O(1)$  transmission rate and is thus *optimal*.

## 5 Conclusion and Open Problems

We gave two 1-round *optimal* APSMT protocols for connectivities  $n = 2t + k$  and  $n = (2 + c)t, c > \frac{1}{t}$ . The first protocol has the highest reliability compared to existing optimal 1-round APSMT protocols. The second protocol is the first for this kind of connectivity. Constructing optimal 1-round APSMT protocols with higher reliability remains an interesting open problem.

**Acknowledgments.** Financial support for this research was provided in part by Alberta Innovates Technology Future in the Province of Alberta, as well as NSERC (Natural Sciences and Engineering Research Council) in Canada. The authors wish to thank Yvo Desmedt and Qiushi Yang for their comments on an earlier version of the paper.

## References

1. Araki, T.: Almost Secure 1-Round Message Transmission Scheme with Polynomial-Time Message Decryption. In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 2–13. Springer, Heidelberg (2008)
2. Agarwal, S., Cramer, R., de Haan, R.: Asymptotically Optimal Two-round Perfectly Secure Message Transmission. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 394–408. Springer, Heidelberg (2006)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation (extended abstract). In: Proc. of STOC, pp. 1–10 (1988)
4. Chan, H., Perrig, A., Song, D.: Random Key Predistribution for Sensor Networks. In: Proc. of IEEE Conference on Security and Privacy (2003)
5. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly Secure Message Transmission. *Journal of the ACM* 40(1), 17–47 (1993)
6. Desmedt, Y., Erootkritou, S., Safavi-Naini, R.: Simple and Communication Complexity Efficient Almost Secure and Perfectly Secure Message Transmission Schemes. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 166–183. Springer, Heidelberg (2010)
7. Desmedt, Y.G., Wang, Y.: Perfectly Secure Message Transmission Revisited. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 502–517. Springer, Heidelberg (2002)
8. Franklin, M.K., Wright, R.N.: Secure Communication in Minimal Connectivity Models. *Journal of Cryptology* 13(1), 9–30 (2000)
9. Kurosawa, K., Suzuki, K.: Almost Secure (1-Round,  $n$ -Channel) Message Transmission Scheme. In: Desmedt, Y. (ed.) ICITS 2007. LNCS, vol. 4883, pp. 99–112. Springer, Heidelberg (2009)
10. Patra, A., Choudhary, A., Srinathan, K., Rangan, C.: Unconditionally Reliable and Secure Message Transmission in Undirected Synchronous Networks: Possibility, Feasibility and Optimality. *IJACT* 2(2), 159–197 (2010)
11. Srinathan, K., Narayanan, A., Pandu Rangan, C.: Optimal Perfectly Secure Message Transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004)
12. Wang, Y.: Robust Key Establishment in Sensor Networks. *SIGMOD Record* 33(1), 14–19 (2004)

# A New Approach towards Coercion-Resistant Remote E-Voting in Linear Time

Oliver Spycher<sup>1,2</sup>, Reto Koenig<sup>1,2</sup>, Rolf Haenni<sup>2</sup>, and Michael Schläpfer<sup>3</sup>

<sup>1</sup> University of Fribourg, CH-1700 Fribourg, Switzerland

{oliver.spycher,reto.koenig}@unifr.ch

<sup>2</sup> Bern University of Applied Sciences, CH-2501 Biel, Switzerland

{oliver.spycher,reto.koenig,rolf.haenni}@bfh.ch

<sup>3</sup> ETH Zurich, CH-8092 Zurich, Switzerland

michschl@inf.ethz.ch

**Abstract.** Remote electronic voting has attracted increasing attention in cryptographic research. A promising protocol presented by Juels et al. is currently widely discussed. Although it offers a remarkably high degree of coercion-resistance under reasonable assumptions, it can not be employed in practice due to its poor efficiency. The improvements that have been proposed either require stronger trust assumptions or turned out to be insecure. In this paper, we present an enhancement of the protocol, which runs in linear time without changing the underlying trust assumptions.

## 1 Introduction

Many governments are aiming at introducing modern technology into their voting processes. Particularly, remote e-voting systems are meant to make voting easier, faster, and more attractive. As appealing as that may seem, introducing physical distance between the voter and the ballot-box comes with a price. Since voters can no longer witness their ballot reach its destination with their own eyes, they need to be provided with another means of assurance. At first sight, this seems to be a simple problem, easily solvable by publishing the set of collected ciphertext votes to let voters verify that their votes have been cast as intended. However, care needs to be taken. Generally, such an approach will allow voters to prove violent coercers or generous vote buyers how they voted. Since voter coercion and vote buying (short: coercion) are highly scalable in an electronic network environment, they need to be prevented. Unfortunately, it seems very difficult to prove voters that their vote is cast as intended (*individual verifiability*), without allowing them to prove others how they voted (*receipt-freeness*).

The protocol underlying this paper was published in 2005 by Juels, Catalano, and Jakobsson [8], often referred to as *the JCJ protocol*. Even today, it seems to be the only known protocol for remote e-voting that offers individual verifiability and receipt-freeness simultaneously under somewhat acceptable trust assumptions. Apart from disabling voters from proving how they voted, the protocol

even ensures immunity against coercers who try to force voters into handing out their credentials (simulation attack) or not casting their votes at all (forced abstention attack). Protocols that avoid all conceivable attacks of coercion are attributed *coercion-resistant*. The JCJ protocol offers a remarkably high degree of coercion-resistance.

Since JCJ imposes unrealistic computational requirements on the tallying authorities, it can not be employed in a real-world context. Nevertheless, the protocol is widely discussed and taken as a starting point for further improvements [2–4, 11, 12]. The ultimate goal of these proposals is to reduce the quadratic running time of the JCJ tallying procedure. We propose our modification of the JCJ protocol to allow tallying in linear time. Section 2, describes JCJ in more detail and points out its security properties and trust assumptions. Section 3 presents our modification of the protocol and shows why the security properties of JCJ are preserved without having to strengthen any trust assumptions. Section 4 concludes the paper and exposes some open questions.

## 2 The JCJ Protocol

To achieve receipt-freeness, other protocols need to assume an *untappable channel* [10] between authorities and voters at every voting event. Requiring voters to visit the authorities' offices at each occasion clearly compromises the spirit of remote e-voting. JCJ is distinguished by assuming an untappable channel only during the distribution of the voters' credentials. Since JCJ allows credentials to be re-used in many subsequent voting events, they can be distributed easily when citizens appear in person at the administration offices to register as new community members.

### 2.1 Description of the Protocol

In the following paragraphs, we present each phase of the JCJ protocol. Due to space constraints, we settle for a semi-formal style of exposition. In particular, we do not thoroughly explain well-known cryptographic techniques. Furthermore, we assume the application of publicly verifiable group threshold mechanisms whenever registering or tallying authorities perform joint computations, even if the text might suggest a single entity. All ciphertexts are ElGamal encryptions over a pre-established multiplicative cyclic group  $(\mathcal{G}_q, \cdot, 1)$  of order  $q$ , for which the decisional Diffie–Hellman problem (DDHP) is assumed to be hard.

**Registration.** The *registrars* jointly establish the random credential  $\sigma \in \mathcal{G}_q$  and pass it to voter  $V$  through an untappable channel. Additionally, they append a randomized encryption  $S = \text{Enc}_\varepsilon(\sigma, \alpha_S)$  of  $\sigma$  to  $V$ 's entry in the voter roll, which is modeled as a public bulletin board. Value  $\alpha_S$  denotes the encryption's randomness, and  $\varepsilon$  stands for the tallying authorities' common public key. Assuming a majority of trustworthy registrars, in the end only  $V$  will know  $\sigma$  and no one will know  $\alpha_S$ .

**Vote Casting.** Voter  $V$  identifies her choice  $c$  from the available set of valid choices (or candidates)  $\mathcal{C}$ . To cast the vote, she posts the encryptions  $A = \text{Enc}_\varepsilon(\sigma, \alpha_A)$  and  $B = \text{Enc}_\varepsilon(c, \alpha_B)$  to the public bulletin board, through an anonymous channel. The pair  $(A, B)$  must be accompanied by two non-interactive zero-knowledge proofs (NIZKP), one to prove knowledge of  $\sigma$  and one to prove  $c \in \mathcal{C}$ . Requiring the first proof prevents attackers from casting unauthorized votes by re-encrypting entries from the voter roll (recall that  $\alpha_S$  is not known to anyone). Since each authorized vote on the voting board will be decrypted during the tallying phase, the second proof is needed to prevent coercers from forcing voters to select  $c \notin \mathcal{C}$  according to some prescribed pattern, thus obtaining a receipt [6]. To circumvent coercion, the voter can deceive the coercer by posting a fake vote to the voting board. To do so,  $V$  simply claims some  $\sigma' \in \mathcal{G}_q$  to be her real credential and uses it to compute  $A$ . She computes  $B$  according to the coercer's preference and reveals the plaintexts of  $A$  and  $B$  to justify compliance. Alternatively,  $V$  can even let the coercer compute  $A$  and  $B$  and cast the vote using  $\sigma'$ .

**Tallying.** At the end of the vote casting phase, the voting board contains  $N$  posted votes, of which not all must be counted. First, the *talliers* verify all proofs that were cast along with the votes. If a proof does not hold for a vote  $(A, B)$ , it is marked accordingly and excluded from further processing. Then the talliers need to filter out votes that were cast multiple times with a proper credential and votes that were cast with a fake credential. For both tasks, the authors of JCJ propose the application of a *plaintext equivalence test* (PET) [7]. Given two ElGamal encryptions  $X = \text{Enc}_\varepsilon(x, \alpha_X)$  and  $Y = \text{Enc}_\varepsilon(y, \alpha_Y)$ , the algorithm  $\text{PET}(X, Y)$  returns *true* for  $x = y$  and *false* for  $x \neq y$ , without revealing any information on  $x$  or  $y$ .<sup>1</sup>

**Removing Duplicates.** Exclude from further processing all  $(A_i, B_i)$ , for which the voting board contains  $(A_j, B_j)$ ,  $i \neq j$ , such that  $\text{PET}(A_i, A_j)$  returns *true*. Given that the voting board contains the votes in the order as cast, a “last-vote-counts” (“first-vote-counts”) policy is implemented by starting the search with big (small) values  $j$ . This exhaustive search over the entire voting board of size  $N$  runs in  $\mathcal{O}(N^2)$  time.

**Removing Invalid Votes.** Invalid votes could easily be excluded from the tally by applying  $\text{PET}(S_i, A_j)$  in an exhaustive search over all values  $S_i$  of the voter roll and all values  $A_j$  of the voting board, similarly to the previous step. However, that would allow the voters to prove the coercer how they voted. To prevent that, the voter roll and the voting board are mixed and re-encrypted using a verifiable re-encryption mixnet, resulting in values  $\widehat{S}_i$  and  $(\widehat{A}_j, \widehat{B}_j)$ , respectively. Now talliers compute  $\text{PET}(\widehat{S}_i, \widehat{A}_j)$  for all pairs  $\widehat{S}_i$  and  $\widehat{A}_j$ . If the algorithm returns *true* for some index  $i$ ,  $\widehat{B}_j$  is decrypted and counted in the tally. This procedure runs in  $\mathcal{O}(N \cdot n)$  time, where  $n$  denotes the size of the voter roll.

---

<sup>1</sup> A common way of performing PET in a homomorphic encryption scheme is to check whether the decryption of  $(X/Y)^z$  equals 1 for some random value  $z \in \mathbb{Z}_q$ .

If the voting board is flooded with a large number of fake votes,  $N$  may be orders of magnitudes larger than  $n$ , which implies that the JCJ tallying procedure has an  $\mathcal{O}(N^2)$  worst-case running time (quadratic with respect to the number of votes). This makes the scheme not only vulnerable to denial-of-service attacks, but also practically infeasible in large-scale settings. The authors of Civitas, a running prototype implementation based on JCJ, have shown this in [5].

## 2.2 Security Properties and Assumptions

We briefly want to point out, to which degree JCJ satisfies the key requirements *privacy* and *accuracy*, and why JCJ provides a high level of coercion-resistance. Privacy is motivated by the notion of the secrecy of the vote. It is satisfied if no vote can be linked to the voter from whom it originates. Accuracy captures the notion that all (and only) legitimate votes are tallied as cast.

**Privacy.** With respect to privacy, JCJ relies on the security of the anonymous channel and the trustworthiness of the tallying and mixing authorities. Since a majority of tallying authorities could collude to jointly decrypt entries of the voter roll and the voting board, they could easily break privacy. Similarly, the mixing authorities could violate privacy by jointly establishing a link from the decrypted votes back to the voter roll. In both cases, the violation of privacy could be hidden by the conspiring parties.

**Accuracy.** By observing the voting board, voters verify that their vote has been cast as intended. Changing or removing votes from the tally would be detected by the public. Adding illegitimate votes requires the knowledge of a credential  $\sigma$  that complies with a value  $S$  in the voter roll. Since all values  $S$  are related to a voter enlisted in the voter roll, adding an illegitimate value could be noticed by voters that are about to register. Attacks of that kind are thus not scalable. As pointed out in the previous paragraph, a colluding majority of authorities could secretly decrypt  $S$  to obtain  $V$ 's valid credential  $\sigma$ . However, if they use  $\sigma$  for casting votes, they could be exposed by  $V$  when the corresponding PET algorithm returns *true* at removing duplicates during the tallying procedure.

**Coercion-Resistance.** Assuming that the coercer cannot communicate with the registrars, voters can always lie about their credentials  $\sigma$ . They are thus protected against coercers that want to push them into voting in a prescribed way, voting at random, or handing out their credentials. If the coercer wants  $V$  to abstain from voting,  $V$  can still cast a vote, given at least one moment of privacy. As pointed out before, we allow a minority of authorities to be untrusted. Disallowing communication between the coercer and all registrars would strengthen that assumption. However, allowing communication would enable the coercer to force the voter into handing out the proper credential: The coercer could claim knowledge about the secret share that a colluding registrar has provided to  $V$ , without saying which one. To be safe,  $V$  needs to hand out all secrets truthfully. We therefore need to assume that the voter knows at least one registrar not colluding with the coercer. Note that this is not implied by assuming *any* majority of trustworthy registrars. Thus,  $V$  can lie to the coercer about that secret.

### 3 Coercion-Resistance in Linear Time

We pointed out that the steps to remove duplicate and illegitimate votes are inefficient in the original JCJ protocol. This issue is widely discussed and has been addressed in the literature. Before presenting our enhancement of the scheme, we introduce two highly promising known approaches, that also aim at improving efficiency at tallying.

**Scheme by Smith and Weber [11–13].** Instead of applying  $\text{PET}(A_i, A_j)$  on all pairs of distinct ciphertexts for removing duplicates,  $1 \leq i, j \leq N$ , both Smith and Weber in essence suggest computing and decrypting  $A_1^z = \text{Enc}_\varepsilon(\sigma_1^z), \dots, A_N^z = \text{Enc}_\varepsilon(\sigma_N^z)$ , where  $z \in \mathbb{Z}_q$  is a random value shared among the talliers. The resulting *blinded* values  $\sigma_i^z$  are stored in a hash table for collision detection in linear time. Clearly,  $\sigma_i = \sigma_j$ , iff  $\sigma_i^z = \sigma_j^z$ . Both authors propose using the same procedure for eliminating illegitimate votes. In that case, however, based on the fact that the same exponent  $z$  is used across all ciphertexts, the coercer gets an attack strategy to identify whether a vote with known  $\sigma$  is counted [2, 5, 9]. Note that this attack does not apply at removing duplicates.

**Scheme by Araujo et al. [1–3].** To solve the efficiency problem of the JCJ scheme, the authors suggest an approach based on group signatures. At registration, voters obtain their credential. Unlike JCJ, no public values are related to voter roll entries. Their credentials enable the voters to deduce invalid credentials and mislead coercers. If the provided proofs hold, duplicates on the voting board are publicly identifiable by the equality of two values that are cast along with the vote. After mixing the relevant values on the voting board, the tallying authorities use their private keys to identify the legitimate votes. Notably, all information on their legitimacy is sent along with the vote itself, but can only be assessed by a sufficiently large group of talliers. Fully avoiding matches between cast values and voter roll entries summarizes the essence of this elegant approach to avoid the inefficient comparison procedure.

An inherent weakness of this approach is the fact that a majority of colluding registrars could compute valid (but illegitimate) credentials unnoticed. As described earlier, adding illegitimate votes to the tally in JCJ requires the knowledge of a credential  $\sigma$  that complies with an entry  $S$  in the voter roll, i.e., such attacks could easily be detected. This is not the case in Araujo et al.’s scheme. Nevertheless, we believe that the approach holds much potential.

#### 3.1 Description of the Enhanced Protocol

Our enhancement strongly relates to the original JCJ, so the modifications are easily summarized. For removing duplicates, we propose using the linear-time scheme proposed by Smith and Weber. For identifying the legitimate votes, we suggest preserving the use of the voter roll. The key to efficiency lies in requiring voters to indicate which voter roll entry their vote ( $A, B$ ) relates to. Talliers then apply PET only on respective re-encryptions of  $A$  and  $S$ , where  $S$  is the public value copied from the indicated voter roll entry. Authorizing legitimate

votes thus becomes linear over the total number of cast votes. In the following paragraphs, we present the protocol in further detail. Later we justify why the security properties of JCJ are preserved under unchanged trust assumptions.

**Registration.** The registration step is conducted according to JCJ. Additionally, we assume that a distinct public number  $i$  is assigned to each voter. For simplicity, we take  $i$  to be the index of  $V$ 's entry in the voter roll.

**Vote Casting.** To cast a vote,  $V$  performs the same steps as in JCJ. Additionally to posting values  $A$  and  $B$  along with corresponding proofs,  $V$  posts the value  $C = \text{Enc}_\varepsilon(i, \alpha_C)$ , accompanied by a non-interactive zero-knowledge proof to prove knowledge of  $i$ . The tallying authorities will later use  $i$  to locate  $S_i$  on the voter roll and efficiently detect legitimate votes. Note that the voting board must also accept wrong values  $C \neq \text{Enc}_\varepsilon(i, \alpha_C)$ .

**Tallying.** After excluding votes with invalid proofs, the talliers add a random number  $X_i$  of additional fake votes for each voter (see discussion below). After removing duplicates by applying Smith's and Weber's scheme on values  $A_i$ , the resulting adjusted list is passed as input to a first re-encryption mixnet, which outputs tuples  $(\hat{A}_j, \hat{B}_j, \hat{C}_j)$ . Next, the talliers jointly decrypt  $\hat{C}_j$  into  $i$  and establish a list of tuples  $(\hat{A}_j, \hat{B}_j, S_i)$ . Votes for which the decryption renders an invalid index  $i \notin \{1, \dots, n\}$  are excluded from further processing. The remaining tuples  $(\hat{A}_j, \hat{B}_j, S_i)$  are then passed to a second re-encryption mixnet, which outputs tuples  $(\tilde{A}_j, \tilde{B}_j, \tilde{S}_i)$ . Now the talliers perform PET( $\tilde{A}_j, \tilde{S}_i$ ) for each tuple. If the algorithm returns *true*,  $\tilde{B}_j$  is decrypted and counted.

The generation of additional fake votes is important to conceal the existence of a real vote after employing the first mixnet, where the encrypted voter roll indices  $i$  are decrypted. The presence of fake votes at that point enables voters to repudiate the fact of having posted a valid vote. The fake votes must be generated and published by trustworthy authorities, such that the exact number of fake votes for voter  $V$  is not revealed. We will later argue that it is sufficient if the registrar, who enjoys  $V$ 's trust, posts a random number  $X \geq 1$  of fake votes designated to  $V$ . Clearly, if  $X$  is independent of  $N$  for all voters, then our tallying procedure runs in  $\mathcal{O}(N)$  time (provided that each of the two mixnets runs in linear time).

### 3.2 Security Properties and Assumptions

In JCJ, coercion-resistance is based on the voter's ability to lie about  $\sigma$  and secretly cast the real vote in a private moment. Note that by witnessing the voter casting a vote, the coercer will need to assume that the voter did not reveal the proper credential. Thus, that one moment of privacy is required in any mode of coercion, not only in the event of a forced abstention attack. The voter may then claim not having cast *any* vote, except possibly the one instructed by the coercer, posted with a fake credential. We will now argue why this argument yields coercion-resistance in our scheme as well.

During the vote casting phase, the voting board reveals no more information to the coercer than in JCJ. During the tallying phase, however, the coercer learns how many votes are related to  $V$ 's entry in the voter roll. Let  $x$  denote that number. The coercer's strategy is to decide, whether  $x$  is distributed according to the same random distribution as for the other voters or if it is greater by one.

**Taking the Risk.** To decide whether  $x$  originates from  $X$  or  $X + 1$  based on one sample (or even a few in case of repeated coercion in subsequent voting events) seems hardly feasible. This conjecture is additionally supported by the fact that other voters may also attribute fake votes to  $V$ . We believe that  $V$  is likely to take the small risk of being caught, in case  $V$  is not exposed to the risk of being punished. If the coercer needs to assume that voters will generally not fear getting caught, any coercion attack seems obsolete. Therefore, we are confident that this notion suffices for solving the vote buying problem, which is the only concern in many countries regarding the notion of coercion.

**Understanding the Risk.** The more voter  $V$  fears consequences in the event of getting caught, the more important it becomes to quantify the risk.  $V$  will agree to co-operate with the coercer unless  $V$  is actually convinced that the risk of getting caught is vanishingly low. That it is infeasible to decide whether  $x$  originates from  $X$  or  $X + 1$  given a distribution function  $F_X$  with a high standard deviation, even over reasonably small values, is a hypothesis we will quantify in our future work. For the time being, we relate the problem to an analogy in JCJ.

We thus give an idea of how the distribution function  $F_X$  for determining the random value  $X$  needs to be defined, in order to make our scheme as secure as JCJ. Recall that in JCJ,  $V$  needs a time-slice  $\Delta t$  of privacy for casting the real vote. Note that the coercer may monitor the voting board at the beginning and at the end of that time-slice. Let  $x'$  denote the number of votes posted during that time. The coercer's strategy is to decide whether  $x'$  is distributed according to the same random distribution as the other values  $x'_i$  of the remaining  $m - 1$  time-slices of equal length (if  $T$  denotes the total length of the voting period, we have  $m = \frac{T}{\Delta t}$ ). If JCJ is coercion-resistant for defined  $n$  and  $m$ , an average of  $x' = \frac{N}{m}$  votes is sufficient to disguise  $V$ 's additional vote (or  $x' = \frac{n}{m}$  assuming all voters participate and post one vote only).  $V$  obviously enjoys the same protection in our scheme, if the registrar's random function  $F_X$  produces an average of  $x = \frac{n}{m}$  fake votes. The same distribution function  $F_X$  can then be employed in the case of a greater number of voters  $n$ .

## 4 Conclusion and Future Work

We have shown a new protocol that solves the efficiency problem in coercion-resistant remote e-voting, without changing the trust assumptions and the security features of JCJ. We pointed out that coercion-resistance of both JCJ and our scheme assumes a private moment for voters to secretly cast their vote. This is only sufficient if the coercer can not deduce from the voting board whether voters took advantage of their privacy. We have related this problem to distinguishing whether  $x$  is distributed as  $X$  or  $X + 1$  for reasonably small values and

a high standard deviation of  $F_X$ , where there is one sample per voting event. In our future work, we will show formally that this problem is infeasible to solve and thus justify JCJ and our scheme to be sufficiently secure against coercion attacks under the known assumptions.

**Acknowledgments.** Research supported by the *Hasler Foundation* (project No. 09037), the *Mittelbauförderung* of the Bern University of Applied Sciences, and the Swiss Federal Chancellery.

## References

1. Araújo, R.: On Remote and Voter-Verifiable Voting. Ph.D. thesis, Department of Computer Science, Darmstadt University of Technology, Darmstadt, Germany (2008)
2. Araújo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. In: Chaum, D., Kutylowski, M., Rivest, R., Ryan, P. (eds.) FEE 2007, Frontiers of Electronic Voting, Dagstuhl, Germany, pp. 330–342 (2007)
3. Araújo, R., Ben Rajeb, N., Robbana, R., Traoré, J., Youssfi, S.: Towards Practical and Secure Coercion-Resistant Electronic Elections. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 278–297. Springer, Heidelberg (2010)
4. Clark, J., Hengartner, U.: Selections: Internet voting with over-the-shoulder coercion-resistance. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 47–61. Springer, Heidelberg (2011)
5. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: 29th IEEE Symposium on Security and Privacy, SP 2008, pp. 354–368 (2008)
6. Di Cosmo, R.: On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack. Hyper Articles en Ligne hal-00142440(2) (2007)
7. Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation Via Cipher-texts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
8. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Atluri, V., De Capitani di Vimercati, S., Dingledine, R. (eds.) 4th ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, USA, pp. 61–70 (2005)
9. Pfitzmann, B.: Breaking an Efficient Anonymous Channel. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995)
10. 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)
11. Smith, W.D.: New cryptographic voting scheme with best-known theoretical properties. In: FEE 2005, Workshop on Frontiers in Electronic Elections, Milan (2005)
12. Weber, G., Araujo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: ARES 2007, 2nd International Conference on Availability, Reliability and Security, Vienna, Austria, pp. 908–916 (2007)
13. Weber, S.: Coercion-Resistant Cryptographic Voting: Implementing Free and Secret Electronic Elections. VDM Verlag, Saarbrücken (2008)

# An Attack on PUF-Based Session Key Exchange and a Hardware-Based Countermeasure: Erasable PUFs

Ulrich Rührmair<sup>1,\*</sup>, Christian Jaeger<sup>2</sup>, and Michael Algasinger<sup>2</sup>

<sup>1</sup> Computer Science Department

<sup>2</sup> Walter Schottky Institut

Technische Universität München

85748 Garching

ruehrmair@in.tum.de,

{christian.jaeger,michael.algasinger}@wsi.tum.de

<http://www.pcp.in.tum.de>

**Abstract.** We observe a security issue in protocols for session key exchange that are based on Strong Physical Unclonable Functions (PUFs). The problem is illustrated by cryptanalyzing a recent scheme of Tuyls and Skoric [1], which has been proposed for use in a bank card scenario. Under realistic assumptions, for example that the adversary Eve can eavesdrop the communication between the players and gains physical access to the PUF twice, she can derive previous session keys in this scheme. The observed problem seems to require the introduction of a new PUF variant, so-called “*Erasable PUFs*”. Having defined this new primitive, we execute some first steps towards its practical implementation, and argue that Erasable PUFs could be implemented securely via ALILE-based crossbar structures.

## 1 Introduction

**Motivation and Background.** Electronic devices have pervaded our everyday life, making them a well-accessible target for adversaries. Classical cryptography offers several measures against the resulting security and privacy problems, but they all rest on the concept of a secret binary key: They presuppose that the electronic devices can contain a piece of information that is, and remains, unknown to an adversary. However, this requirement can be difficult to uphold in practice: Physical attacks such as invasive, semi-invasive, or side-channel attacks, as well as software attacks like API-attacks and viruses, can lead to key exposure and security breaks.

The described situation was one motivation that led to the development of *Physical Unclonable Functions (PUFs)*. A PUF is a (partly) disordered physical system  $S$  that can be challenged with so-called external stimuli or challenges  $C_i$ , upon which it reacts with corresponding responses  $R_i$ . Contrary to standard digital systems, a PUF’s responses shall depend on the nanoscale structural disorder present in it. It is assumed that this disorder cannot be cloned or reproduced exactly, not even by the PUF’s original manufacturer, and that it is unique to each PUF. This means that any PUF  $S$  implements

---

\* Corresponding author.

an individual function  $F_S$  mapping challenges  $C_i$  to responses  $R_i$ . The tuples  $(C_i, R_i)$  are thereby often called the *challenge-response pairs (CRPs)* of the PUF.

Due to its complex internal structure, a PUF can avoid some of the shortcomings associated with digital keys. It is usually harder to read out, predict, or derive its responses than to obtain the values of digital keys stored in non-volatile memory. This fact has been exploited for various PUF-based security protocols. Prominent examples include schemes for identification [2], key exchange [1], or digital rights management purposes [3] [5]. Another advantage of (Strong) PUFs is that they can lead to protocols whose security does not depend on the usual, unproven number theoretic assumptions (such as the factoring or discrete logarithm problem), but rests on independent hypotheses.

**Strong PUFs and Weak PUFs.** Two important subtypes of PUFs, which must explicitly be distinguished in this paper, are *Strong PUFs*<sup>1</sup> and *Weak PUFs*<sup>2</sup>. This distinction has been made first in [4] [3], and has been elaborated on further in [6] [7] [8].

*Strong PUFs* are PUFs with a very large number of possible challenges. The adversarial ability to apply challenges to them and to read out their responses from them is usually not restricted. Their central security features are: (i) It must be impossible to physically clone a Strong PUF, i.e. to fabricate a second system which has the same challenge-response-behavior as the original PUF. This restriction must hold even for the original manufacturer of the PUF. (ii) Due to the very large number of possible challenges and the PUF's finite read-out rate, a complete measurement of all challenge-response pairs (CRPs) within a limited time frame (such as several days or even weeks) must be impossible. (iii) It must be difficult to numerically predict the response  $R_i$  of a Strong PUF to a randomly selected challenge  $C_i$ , even if many other challenge-response pairs are known.

A complete formal specification of Strong PUFs is laborious and besides the scope of this paper, but can be found in [7]. Examples of candidates for Strong PUFs are complex optical scatterers [2] or special, delay-based integrated circuits [9] [10] [11] (albeit several of the latter have been broken up to a certain size in recent machine learning attacks [6]). Also analog circuits have been proposed recently [12].

*Weak PUFs* may have very few challenges — in the extreme case just one, fixed challenge. Their response(s)  $R_i$  are used to derive a standard secret key, which is subsequently processed by the embedding system in the usual fashion, e.g. as a secret input for some cryptoscheme. Contrary to Strong PUFs, the responses of a Weak PUF are never meant to be given directly to the outside world.

Weak PUFs essentially are a special form of non-volatile key storage. Their advantage is that they may be harder to read out invasively than non-volatile memory like EEPROM. Typical examples of Weak PUFs are the SRAM PUF [3], Butterfly PUF [5] and Coating PUF [13].

---

<sup>1</sup> Strong PUFs have also been referred to as Physical Random Functions [9] [10], or (almost equivalently) as Physical One-Way Functions [2] in the literature.

<sup>2</sup> Weak PUFs have also been referred to as Physically Obfuscated Keys (POKs) [4]. Note that the predicate “Weak” is not meant to state that these PUFs are “bad” in any sense, we just follow the terminology introduced in [3].

*Applications of Strong PUFs.* We are mostly concerned with Strong PUFs and variants thereof in this paper, whence we focus on them from now on. The archetypical application of Strong PUFs is the identification of entities over insecure networks. It has already been suggested in the first PUF publication [2] by the example of a bank card scenario, and works along the following lines. Each customer’s bank card contains an individual Strong PUF. Before issuing the card, the bank measures several of the PUF’s CRPs, and stores them secretly on its server. When the customer inserts his card into a terminal, the terminal contacts the bank. The bank chooses at random several challenges  $C_i$  from its secret CRP list, and sends them to the terminal. The terminal obtains the corresponding responses  $R_i$  from the PUF, and returns them to the bank. If they match the values in the CRP list, the bank considers the card as genuine. The scheme has the upsides of circumventing the need for secret keys or secret information on the vulnerable bank cards, and of avoiding the usual, unproven complexity theoretic assumptions of classical identification protocols.

A second, central application of Strong PUFs, which also has already been suggested in [2] (page 2029), and which has been worked out in greater detail in [1], is the distribution of a secret key between different parties, for example the terminal and the bank. We are mainly concerned with this second application in this paper.

**Our Contributions.** Our first contribution is to observe a problem in the repeated use of PUF-based session key exchange protocols. We illustrate this problem by the example of a recent protocol by Tuyls and Skoric [1], which has originally been suggested for use in a bank card scenario. We show how to cryptanalyze this protocol under the presumptions that an adversary can eavesdrop the communication between the terminal and the bank, that he has got access to the PUF more than once, and that no secret digital information can be stored on the card. These presumptions seem very natural, even more so in the original application scenario of bank cards or credit cards (see section 2). The problem which our attack exploits is that the CRP-information used to derive a key remains present in the PUF after the completion of the key exchange protocol.

Second, we reason that the described problem cannot be solved via protocol or software measures, and also not on the basis of current PUF architectures. Resolution seems to require the introduction of a new PUF variant, so-called Erasable PUFs. They are a special type of Strong PUF, with the additional feature that the value of single responses can be erased or altered without affecting the value of all other responses. We specify this new primitive, and show how it can be used to fix the above security issues.

Third, we suggest one possible implementation strategy for Erasable PUFs: Large, monolithic crossbar arrays of diodes with random current-voltage characteristics. It has already been demonstrated in earlier work that such crossbar arrays can act as secure Strong PUFs [14] [15] [16]. We now show that the information stored in the diodes of the crossbar can be erased individually: By applying dedicated voltage pulses to selected crossbar wires, the current-voltage curve of any single diode can be altered individually, and without affecting the other diodes in the array. We present measurement data from single ALILE-diodes fabricated in our group that supports the feasibility of the described approach.

**Related Work.** There is no related work concerning the cryptanalysis of the Strong PUF-based session key exchange protocol by Tuyls and Skoric. In general, the cryptanalysis of PUF-based protocols appears to be a relatively recent field. Previous PUF attacks mainly focused on breaking the security properties of PUFs themselves (for example by modeling Strong PUFs via machine learning techniques [6]), but not on analyzing PUF protocols.

With respect to Erasable PUFs, there is obviously a large body of work on Strong PUFs and Weak PUFs, but none of them explicitly considered the property of erasing individual CRPs without affecting other CRPs. The category of PUFs which comes closest to Erasable PUFs are Reconfigurable PUFs (r-PUFs) [17], but the previously proposed optical, scattering-based implementation of r-PUFs has the property that inevitably *all* CRPs are altered by the reconfiguration operation. No erasure or alteration on a single CRP level is enabled. See also section 4 for a further discussion.

**Organization of the Paper.** In Section 2, we illustrate a security problem occurring in PUF-based key establishment protocols. Section 4 discusses the implementation of Erasable PUFs via crossbar structures. Section 4 describes a few obstacles in the practical realization of Erasable PUFs. Section 5 gives some background on the recent concept of a Crossbar PUF. Section 6 describes how information can be erased from Crossbar PUFs, implementing Erasable PUFs. We conclude the paper in Section 7.

## 2 A Problem with PUF-Based Session Key Establishment

### 2.1 The Protocol of Tuyls and Skoric

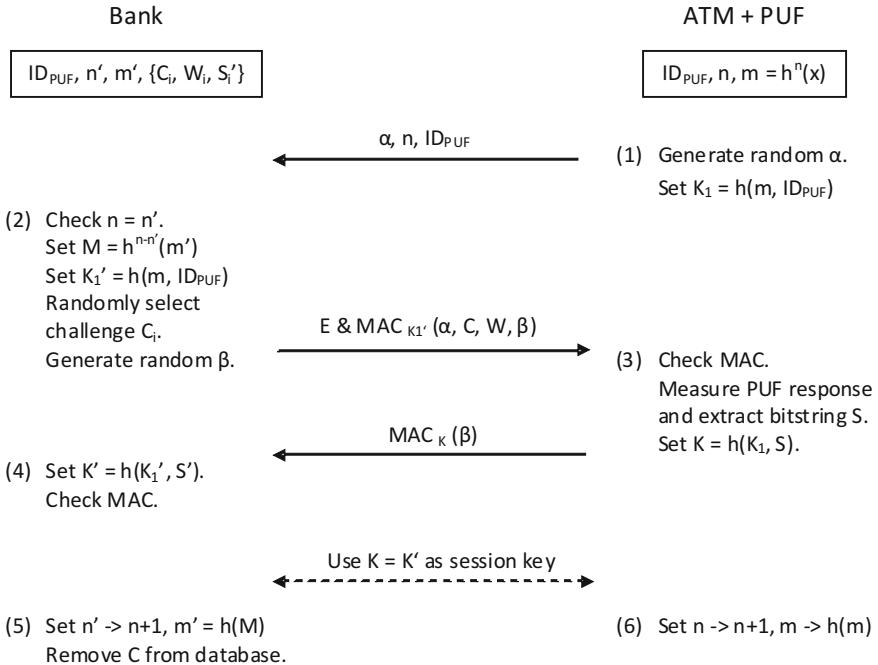
A specific Strong PUF-based protocol for combined identification and session key establishment has been suggested recently in [1]. It is illustrated in Fig. 1. The protocol is run between a Bank on the one hand and an Automated Teller Machine (ATM) plus a security token carrying the Strong PUF on the other hand. It presumes that all involved parties have knowledge of a public one-way hash function  $h$ , and of a publicly known error correction scheme, which is used to derive secrets  $S$  from a given noisy PUF-response  $R$  and helper data  $W$ .

The protocol presupposes a set-up phase, in which the bank has got direct access to the Strong PUF. The bank first of all establishes a (large) secret list of the form  $\{C_i, W_i, S'_i\}$ . Thereby the  $C_i$  are randomly chosen challenges,  $W_i$  denotes helper data that is generated by the bank from the corresponding (noisy) responses  $R_i$  of the PUF, and  $S'_i$  refers to secret information that is derived from the noisy response by use of the helper data. Furthermore, the bank chooses a secret numerical value  $x$  at random, and writes  $h(x)$  onto the card. After that, the card is released to the field.

Each subsequent execution of the protocol is run between the bank and the ATM/PUF. At the beginning of the protocol, the token stores the number  $n$  of previous protocol executions, the value  $m = h^n(x)$ , and an identification number of the Strong PUF, denoted as  $ID_{PUF}$ .

The Bank initially holds a list of the form  $\{C_i, W_i, S'_i\}$  that is stored together with  $ID_{PUF}$  in the Bank's database. The value  $n'$  says how often the Bank has been engaged in a session key exchange protocol with the PUF, and  $m' = h^{n'}(x)$ . The rest of the

protocol is described in Fig. 1, which is essentially taken from [1]. At the end of the protocol, the Bank and the ATM/PUF have established a joint session key  $K$ .



**Fig. 1.** A protocol for combined identification and session key exchange based on Strong PUFs, which has been suggested by Tuyls and Skoric in [1].

## 2.2 Problems Arising from Repeated Access to the PUF

We will now present an attack on the repeated use of the above protocol, which allows Eve to derive previous session keys.

The attack makes the following assumptions: (A) Eve can eavesdrop the communication between the bank and the ATM/PUF. (B) No secret digital numbers (e.g., hash values, secret keys) can be stored safely in a non-volatile fashion on the security token. (C) Eve gains access to the security token at least twice, and can measure selected CRPs from the Strong PUF on the token. All of these assumptions are relatively well motivated: If a secure channel would be at hand, which cannot be eavesdropped by Eve, then no complicated session key exchange protocol is necessary. The secret keys could simply be exchanged by sending them over this channel. Likewise, if we were to assume that secret digital keys (or other secret digital numbers) could be stored safely on the token, then the use of PUFs is unnecessary: The token could execute all necessary communication securely via classical, secret key based cryptography. Finally, assumption (C) is straightforward: For example in a bank card scenario, where an adversary might operate with faked terminals/readers that are under his control, and where the

card is inserted multiple times into these terminals/readers. Again, if we do not allow an adversary to obtain physical access to the card, then the use of PUFs is unnecessary in the first place.

Eve's attack works in three successive phases executed at times  $T_1$ ,  $T_2$  and  $T_3$ .<sup>3</sup> In the first phase at time  $T_1$ , we presume that Eve has got access to the token according to assumption (C). By assumption (B), she can read the current values of  $n$  and  $m$  at time  $T_1$  from the token, denoted by  $n(T_1)$  and  $m(T_1)$ .

In the second attack phase at time  $T_2$ , we assume that Eve eavesdrops a session key establishment protocol between the bank and the ATM/PUF. This is possible according to assumption (A). From the first message sent in the protocol, which we denote by  $\alpha(T_2)$ ,  $n(T_2)$ ,  $ID_{PUF}$ , Eve learns the current counter value  $n(T_2)$ . Since Eve already knows  $n(T_1)$  and  $m(T_1)$  from phase 1, she can deduce the current state  $m(T_2) = h^{n(T_2)}(x) = h^{n(T_2)-n(T_1)}(m(T_1))$ . This allows her to derive the value of the preliminary key  $K_1$  at time  $T_2$  by setting  $K_1(T_2) = h(m(T_2), ID_{PUF})$ . Now, when the bank sends the protocol message  $E\&MAC_{K'_1(T_2)}(\alpha(T_2), C(T_2), W(T_2), \beta(T_2))$ , Eve can remove the encryption, because she knows  $K_1(T_2) = K'_1(T_2)$ . She learns  $C(T_2)$  and the helper data  $W(T_2)$ . This closes Eve's contribution in the second attack phase. In the further course of the protocol (and without Eve's involvement), the ATM/PUF measures the PUF and extracts a secret bitstring  $S(T_2)$  from its responses. Finally, the ATM/PUF sets the session key to be  $K(T_2) = h(K_1(T_2), S(T_2))$ .

In the third attack phase at time  $T_3$ , we assume that Eve has got access to the security token and the Strong PUF, and that she can measure CRPs of the Strong PUF. This is in accordance with assumption (C). Eve uses this ability to measure the PUF's responses  $R(T_2)$  that correspond to the challenge(s)  $C(T_2)$ . Note that the Strong PUF's responses are time invariant and are not actively altered by any protocol participant. Hence Eve can determine  $R(T_2)$ , even though the time has progressed to  $T_3$  at this point. Eve also knows  $W(T_2)$ , whence she can derive  $S(T_2)$  from the responses  $R(T_2)$ . This enables her to compute  $K(T_2) = h(K_1(T_2), S(T_2))$ , since she knows  $K_1(T_2)$  already. In other words, Eve obtains the session key  $K(T_2)$  that was derived and used at time  $T_2$ , breaking the protocol's security.

### 2.3 Consequences for CRP Refreshment and Identification

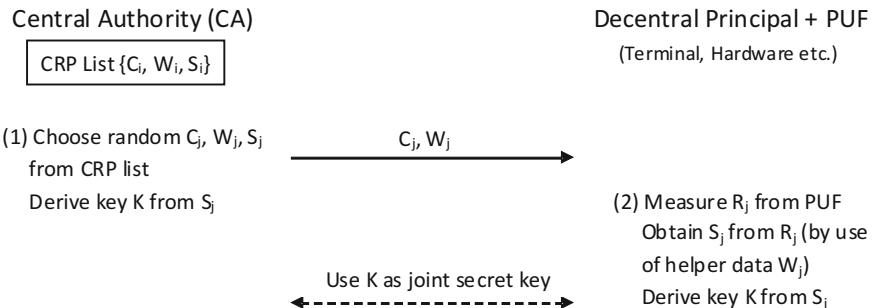
It has been suggested in [1] that a session key  $K$  established via the protocol of Fig. 1 could be used to achieve CRP refreshment between the ATM and the Bank. To that end, the ATM would, in regular intervals, execute the following steps: (i) Measure new data of the form  $\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$  (where  $T_j$  can be an arbitrary point in time). (ii) Exchange a session key  $K(T_j)$  via the protocol of Fig. 1. (iii) Send the encrypted message  $E\&MAC_{K(T_j)}\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$  to the Bank. (iv) The Bank decrypts this message, and adds  $\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$  to its CRP list. This process is termed CRP refreshment. This method allows shorter CRP lists and saves storage requirements on the bank.

<sup>3</sup> In the description of our attack, we will need to consider the value of various protocol parameters, such as  $n$ ,  $m$ , or  $K_1$ , at different points in time. To avoid confusion, we use the notation  $n(T)$ ,  $m(T)$ ,  $K_1(T)$  (or similar expressions) to denote the values of  $n$ ,  $m$  or  $K_1$  at time  $T$ .

But in the attack scenario described in section 2.2, i.e. under the provisions (A) to (C), Eve can break this scheme. First, she can apply the attack described in section 2.2 to obtain  $K(T_j)$ . She can then decrypt the message  $E\&MAC_{K(T_j)}\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$ , and hence learns the values  $\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$  that were intended for CRP refreshment. This enables her to impersonate the PUF in subsequent identification protocols that are built on these CRP values. For example, it allows her to build faked bank cards.

## 2.4 Generality and Difficulty of the Problem

The problem we observed in the previous sections does not only apply to the protocol of Fig. 1. It could be argued that any PUF-based protocol for key establishment between a central authority and decentral principals (terminals, hardware, etc.) involves, explicitly or implicitly, the basic procedure that is shown in Fig. 2.



**Fig. 2.** The “raw”, basic building block for PUF-based key exchange. In practice, it can and will usually be accompanied by other measures, such as message authentication or authentication of the physically transferred PUF.

Any protocol of this form is prone to the type of attack described in section 2.2. Considering the protocol of Fig. 2 sheds light on the heart of the problem: Eve can break the protocol by firstly eavesdropping the  $C_j, W_j$ . Subsequent one-time access to the PUF allows her to measure the corresponding  $R_j$  and to derive the corresponding  $S_j$ . This enables her to obtain  $K$ . We will not give a full formal proof of this statement, but believe that adapted variants of this simple attack can be mounted on any Strong PUF-based session key exchange. One example for such an adapted attack on a much more complicated protocol was given in Sec. 2.2. The key issue in all cases seems that the response information used for the derivation of  $K$  is still extractable from the Strong PUF at later points in time.

It would be hence necessary to “erase” the responses  $R_j$  from the Strong PUF after they have been used for key derivation. Note that in such an “erasure” operation, all other responses  $R_i$  (with  $i \neq j$ ) must remain unchanged: If they were altered, the list  $\{C_i, W_i, S_i\}$  stored at the central authority would no longer be valid. It could neither be used for further key establishment protocols of the above type, nor for the typical PUF-based identification schemes (see Sec. 1).

### 3 Erasable PUFs

We will now make some first steps work towards a hardware-based solution of the above security problem, introducing a new variant of Strong PUFs: So-called Erasable PUFs. For reasons of clarity and unambiguity, we slightly deviate from the established notation for PUFs in the following specification, and denote the response of a PUF  $S$  to a challenge  $C$  by  $R_C^S$ .

**Specification 1 (ERASABLE PUFs)** *A physical system  $S$  is called an ERASABLE PUF if it is a Strong PUF with the following additional properties:*

- *There is a special, physical erasure operation  $\mathbf{ER}(\cdot)$ . It takes as input a challenge  $C_0$  of  $S$ . It turns  $S$  into a system  $S'$  with the following properties:*
  - *$S'$  has got the same set of possible challenges as  $S$ .*
  - *For all challenges  $C \neq C_0$ , it holds that  $R_C^{S'} = R_C^S$ .*
  - *Given  $S'$  and  $C_0$ , it is impossible to determine  $R_{C_0}^{S'}$  with a probability that is substantially better than random guessing.*

Note that Specification 1 is not meant to be a full-fledged formal definition, but shall mainly express the properties of Erasable PUFs in a compact, semi-formal manner. Its style follows [7].

Given the discussion of the previous sections, it is now relatively straightforward to fix the security issues of the protocols of Fig. 1 and 2.

1. PROTOCOL OF FIG. 1: Use an Erasable PUF in the protocol, and add the erasure operation  $\mathbf{ER}(C)$  at the end of step (3).
2. PROTOCOL OF FIG. 2: Use an Erasable PUF in the protocol, and add the erasure operations  $\mathbf{ER}(C_j)$  to the end of step (2).

These steps disable the attacks that have been presented in the previous sections: When Eve has got access to the PUF at a later point in time, she can no more determine the PUF responses used for previous key derivation, as the responses have been erased from the system.

### 4 Obstacles in the Implementation of Erasable PUFs

The implementation of Erasable PUFs on the basis of established PUF architectures turns out to be intricate; we will summarize the occurring difficulties in this section. One reason for the appearing problems is that Erasable PUFs must combine the following properties:

- (i) They must be Strong PUFs, i.e. they must have very many possible challenges, and must be able to withstand full read-out for long time periods, i.e. weeks or months.
- (ii) They must allow the erasure or alteration of single responses, without affecting other responses.

These properties rule out Weak PUFs and their current implementation candidates [3] [5] [13] from the start, since they simply do not fulfill condition (i) above, i.e. they are no Strong PUFs.

An alternative approach would be to modify Strong PUF architectures in order to obtain Erasable PUFs. The erasure operation could, for example, alter some internal components of a Strong PUF. But unfortunately, all popular candidates for Strong PUFs [2] [9] [10] [11] [12] create their responses in a complex interplay of many or even all internal components. Altering one single component will not only change a single response, but will affect many other responses, too. Their responses cannot be altered individually, i.e. with single CRP granularity.

Another, straightforward idea would be to attach an access control module to a Strong PUF. The module could store a list of “forbidden” challenges and prevent the application of these challenges to the Strong PUF. But this approach is costly in practice: It requires non-volatile memory, which must store potentially large amounts of challenges. Furthermore, it cannot reach ultimate security levels: The control module might be circumvented or cut off by a well-equipped attacker, and the content of the memory (i.e. the forbidden challenges) might be manipulated.

The existing concept that presumably comes closest to Erasable PUF are Reconfigurable PUFs (r-PUFs), which were introduced in [17]. By definition, each r-PUF possesses a reconfiguration operation, in which all CRPs of the r-PUF can be changed. However, the currently suggested optical implementation of r-PUFs has the property that all responses are altered by the reconfiguration operation, disabling it as an Erasable PUF. For electrical implementations of r-PUF based on phase-change materials, which are only briefly mentioned aside in [17], it is yet unclear whether they would be Strong PUFs at all, i.e. whether they could be designed to withstand full read-out in short time.

Eventually, there is one recent Strong PUF candidate that seems appropriate to implement Erasable PUFs: So-called Crossbar-based PUFs. They have originally been introduced in [14] [15] [16], and will be treated in the next section.

## 5 Strong PUFs Based on Crossbar Structures

Recent work [14] [15] [16] investigated the realization of a special type of Strong PUF (so-called “SHIC PUFs”<sup>4</sup>). These are Strong PUFs with the additional following properties:

- (i) The PUF possesses maximal information content and density, with all CRPs being mutually (i.e. pairwise) information-theoretically independent.
- (ii) The PUF can only be read out at slow rates.

The motivation behind investigating this type of Strong PUFs was to protect PUFs against any modeling attacks. Such attacks use known CRPs in order to extrapolate the PUF’s behavior on new CRPs, and constitute a serious challenge for the security of Strong PUFs [6]. SHIC PUFs are automatically invulnerable against such modeling attempts, since all of their CRPs are information-theoretically independent: Knowing a subset of CRPs hence does not allow conclusions about other CRPs.

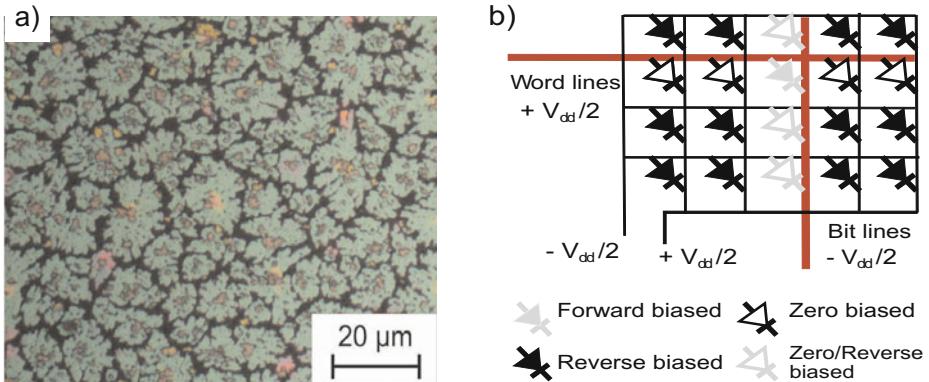
<sup>4</sup> SHIC abbreviates the term “Super-High Information Content”, and is pronounced as “chique”.

Concrete target parameters for the construction of SHIC PUFs discussed in [14] [15] [16] were an information content of up to  $10^{10}$  bits and read-out speeds of  $10^2$  to  $10^3$  bits per second. As argued in [15], such relatively slow read-out speeds are no problem in many typical applications of Strong PUFs, such as bank card identification, key exchange, or also oblivious transfer [18]. On the upside, the combination of slow read out and high information content can potentially immunize the PUF against full read-out for up to month or years of uninterrupted, unnoticed adversarial access [15]. For comparison, several known Strong PUF architectures with a MHz read-out rate can be modeled (and hence broken) via a number of CRPs that can be read out in a few seconds [6].

It has been shown in [14] [15] [16] that SHIC PUFs can be realized by large, monolithic crossbar architectures. At each crosspoint of the crossbar, a diode with a random current-voltage characteristic is present. The necessary random variation in the diodes is generated by a random crystallization technique known as ALILE process. We will review the necessary basics of this approach in this section; much further detail can be found in [14] [15] [16].

*ALILE Crystallization.* In order to construct a Strong PUF with the above properties, one first requires a solid-state fabrication process that generates a maximal amount of entropy in the PUF. The authors of [14] [15] [16] turned to crystallization processes to this end, since the crystallization step amplifies minuscule variations in the starting conditions (such as atomic-scale roughness) to larger, stable variations in the system (for example the shape, size and position of the crystallites). Among many possible crystallization processes, they eventually selected the so-called aluminum-induced layer exchange (ALILE) process [20] [21], since it is a simple crystallization process that involves few production steps and inexpensive starting materials. It results in polycrystalline films with p-type conduction [22], and creates a highly disordered and random structure comprising of crystallized silicon grains (Si) and aluminum (Al). Fig. 3 a depicts the top view onto a crystallized system, illustrating the occurring randomness. By changing the process parameters, the size and density of the grains can be tuned as desired.

*Diodes and Crossbar Read-Out.* In order to read out the information contained in the system, a circuit architecture known as crossbars can be employed. It consists of two sets of parallel wires, one of them applied on the top, the other one at the bottom of the structure. Both sets are arranged orthogonally to each other. The basic schematics are illustrated in Fig. 3 b. Due to the p-n-type cross section of the entire system (the film of p-type conduction is generated on an n-type wafer to this end), each virtual crossing of the crossbar acts like a p-n-diode, with rectification rates of up to  $10^7$  [16]. Its  $I(V)$  curve can be read out by applying a voltage at two chosen crossbar wires (bit and word lines, in analogy to a memory), as illustrated in Fig. 3 b [15]. Due to the random nature of the ALILE crystallization process, the diodes show current-voltage curves which are very irregular and individual in shape. The individual curves differ in their currents by up to four decimal orders of magnitude, but are still stable against aging and multiple measurement [14] [16]. As shown in [14], at least three bits of information can be extracted reliably from each crossing.



**Fig. 3.** (a) Randomly shaped and located Si crystallites (top view, showing the extension in  $x$ - $y$ -directions). (b) Schematic illustration of the crossbar architecture and the diodes at the crossings. Also read-out process, i.e. the selection of a bit line and a word line in order to address and read out a single diode, is illustrated.

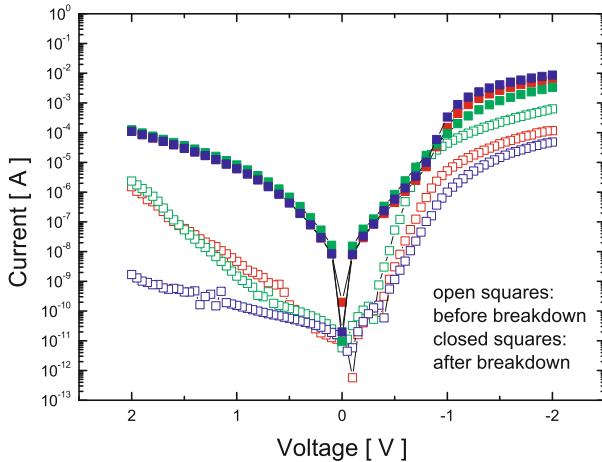
*Information Content and Inherently Slow Read-Out Speed.* Using crossbar architectures has two advantages. First, they can reach ultimate information densities due to their very simple architecture of parallel wires. The information density and content targeted in [15] were  $10^{10}$  bits per  $\text{cm}^2$ . Secondly, they can be designed with an inherently limited read-out speed. To achieve this, the Crossbar PUF is built in one large, monolithic block, not from separate blocks as modern semiconductor memories, and is made from wires that have only finite current-carrying capacity. Simulations conducted in [15] showed that in such large monolithic blocks, several milliseconds must elapse before the sense current/voltage stabilizes. This results in read-out speeds of around 100 bits/sec. Any faster read-out attempts would overload and destroy the wires, leaving the remaining structure unusable [15].

## 6 Erasing Information from Crossbar Structures

We now investigate if – and how – information can be erased from Crossbar PUFs. Since the information is contained in the diodes' current-voltage characteristics, any erasure operation must target the diodes, changing their  $I(V)$ -curves irreversibly. We could not build a device with  $10^{10}$  crossings within the scope of this paper, but argue on the basis of measurement curves obtained from stand-alone fabricated in our group. The fact that the behavior of these single diodes scales very well to the operation of large, monolithic blocks of diodes has been proven in all detail in earlier work [15].

The “erasure operation” works as follows. A specific diode in the crossbar array is chosen by selecting the corresponding bit and word lines of the crossbar structure, similar to the read-out procedure for the crossbars. Then a short voltage pulse of 4 V to 5 V is applied in reverse direction to the diode. This induces a breakdown in the ALILE diode, which destroys the individual information present in the  $I(V)$  curve, and makes all curves after erasure “standardized” and very similar in shape.

This effect has been observed by us in *all measured diodes*; three illustrative examples for  $I(V)$ -curves before and after breakdown are shown in Fig. 4. While the large variations in the original curves range over four orders of magnitude, there is little individuality left after breakdown. The curves after breakdown also differ strikingly from the original curves. Considering the development of the relative positions of the curves over the full voltage range shows that not even the relative positioning of the curves is preserved.



**Fig. 4.** The curves of three exemplary diodes (red, blue and green) before and after breakdown

The fact that the new curves are uncorrelated to the old ones is a consequence of the physical effect behind the breakdown of the diodes. Our explanation of this mechanism is the presence of a thin natural oxide film between the p- and n-layers, effectively resulting in a p-i-n-structure. Such an additional i-layer would strongly reduce the tunneling current in reverse direction (as observed by us), which otherwise had to be expected to be high due to the large hole carrier concentration in the ALILE layers (up to  $10^{19} \text{ cm}^{-3}$ ) [16]. The assumption of an intermediate oxide layer is further supported by the fact that diodes which were exposed to hydrofluoric acid (HF) vapor prior to the deposition of the ALILE layers *did not show* comparable rectification rates; the HF vapor is known to remove Si-oxide, leading to a destruction of the possible p-i-n -structure [23]. The described voltage pulse in reverse direction then simply burns and removes this i-layer.

This physical mechanism behind the erasure supports the security of our construction, for the following reasons: First, the destruction of the thin, irregular oxide film cannot be reversed physically by Eve. Second, after the oxide layer has been removed, independent and secondary features of the structure dominate the  $I(V)$  curve (whereby their effect on the randomness of the curve is by far not as strong as the original configuration, see Fig. 4). From knowing the new curves after breakdown, it is therefore impossible to conclude backwards on the shape of the original  $I(V)$  curves before breakdown.

Finally, please note that the operational voltage for measurement of the diodes in the crossbar structure lies between -2V and +2V. The erasure operation hence is just a

factor of around 2 away from the standard operation of the crossbar. This is compatible with the use of wires with finite current-carrying capacity, which was indispensable to enforce the slow read-out rate of the crossbar (see Section 5, page 200, and [15]).

## 7 Summary

We made the following contributions in this paper. First, we observed a security problem in a recently published session key exchange protocol by Tuyls and Skoric [1], which is based on Strong Physical Unclonable Functions (PUFs). We cryptanalyzed the protocol under the relatively mild presumptions that the adversary gains access to the PUF twice, that she can eavesdrop the communication between the involved parties, and that no secret information can be stored on the card. As discussed earlier, these presumptions are well-motivated, for example in the bank card scenario in which the protocol had been proposed originally. Our attack has severe consequences for the security of the proposed bank card application. The noted security problem seems to be general, applying to any comparable session key exchange based on Strong PUFs.

Second, we introduced a new PUF variant, so-called Erasable PUFs, in order to resolve the described security issue. These are special Strong PUFs, with the additional property that the information stored in single responses of theirs can be irreversibly erased without changing any other response values. As we argued, currently known PUF architectures are unsuited to this end: They either are no Strong PUFs in the first place. Or, they have many interplaying components, which prevents that a single response can be changed without affecting the other responses. The latter problem holds for all delay-based PUFs, but also for the current, optical implementations of Reconfigurable PUFs.

We therefore, thirdly, investigated new architectures for implementing Erasable PUFs. We suggested the use of crossbar structures with randomly crystallized ALILE-diodes. It was known from recent work [14] [15] [16] that such “Crossbar PUFs” can act as Strong PUFs with very high information content and densities and inherently slow read-out speed. We now discussed how the information stored in the ALILE-diodes of the crossbar can be erased individually. Our erasure process works by applying a relatively small threshold current to selected bit and word lines of the crossbar. This induces a “breakdown” in the diode, as it burns intermediate oxide layers. The process is irreversible, and transforms the individual  $I(V)$  curve of any diode into an uncorrelated, new one. The threshold current is low enough to be compatible with the finite current carrying capacity of the crossbar wires and the read-out mechanism of the crossbar array. We supported our proposal by measurements on single, stand alone ALILE-diodes fabricated in our group. It had been shown in extensive simulations in previous work [15] that the behavior of such diodes scales to large diode arrays.

**Acknowledgements.** This work was conducted in the course of the Physical Cryptography Project at the TU München, with support by the Institute for Advanced Study (IAS) and International Graduate School of Science and Engineering (IGSSE) at the TU München. We would like to thank Paolo Lugli, Martin Stutzmann, György Csaba, Ahmed Mahmoud and Michael Scholz for useful discussions.

## References

1. Tuyls, P., Skoric, B.: Strong Authentication with Physical Unclonable Functions. In: Petkovic, M., Jonker, W. (eds.) *Security, Privacy and Trust in Modern Data Management*. Springer, Heidelberg (2007)
2. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical One-Way Functions. *Science* 297, 2026–2030 (2002)
3. Guajardo, J., Kumar, S.S., Schrijen, G.-J., 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)
4. Gassend, B.: Physical Random Functions, MSc Thesis, MIT (2003)
5. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The Butterfly PUF: Protecting IP on every FPGA. In: *HOST 2008*, pp. 67–70 (2008)
6. Rührmair, U., Sehnke, F., Sölder, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling Attacks on Physical Unclonable Functions. In: *ACM Conference on Computer and Communications Security*, Chicago (ILL), USA (2010)
7. Rührmair, U., Busch, H., Katzenbeisser, S.: Strong PUFs: Models, Constructions and Security Proofs. In: Sadeghi, A.-R., Tuyls, P. (eds.) *Towards Hardware Intrinsic Security: Foundation and Practice*, Springer, Heidelberg (2010) (to appear)
8. Rührmair, U., Sölder, J., Sehnke, F.: On the Foundations of Physical Unclonable Functions. *Cryptology e-Print Archive* (June 2009)
9. Gassend, B., Lim, D., Clarke, D., van Dijk, M., Devadas, S.: Identification and authentication of integrated circuits. *Concurrency and Computation: Practice & Experience* 16(11), 1077–1098 (2004)
10. Lee, J.-W., Lim, D., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits with identification and authentication applications. In: *Proceedings of the IEEE VLSI Circuits Symposium* (June 2004)
11. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight Secure PUFs. In: *IC-CAD 2008*, pp. 607–673 (2008)
12. Csaba, G., Ju, X., Ma, Z., Chen, Q., Porod, W., Schmidhuber, J., Schlichtmann, U., Lugli, P., Rührmair, U.: Application of Mismatched Cellular Nonlinear Networks for Physical Cryptography. In: *IEEE CNNA - 12th International Workshop on Cellular Nonlinear Networks and their Applications* (2010)
13. Tuyls, P., Schrijen, G.-J., Škorić, 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)
14. Rührmair, U., Jaeger, C., Hilgers, C., Algasinger, M., Csaba, G., Stutzmann, M.: Security Applications of Diodes with Unique Current-Voltage Characteristics. In: Sion, R. (ed.) *FC 2010*. LNCS, vol. 6052, pp. 328–335. Springer, Heidelberg (2010)
15. Rührmair, U., Jaeger, C., Bator, M., Stutzmann, M., Lugli, P., Csaba, G.: Cryptographic Applications of High-Capacity Crossbar Memories. *IEEE Transactions on Nanotechnology* 99, 1 (2010)
16. Jaeger, C., Algasinger, M., Rührmair, U., Csaba, G., Stutzmann, M.: Random pn-junctions for physical cryptography. *Applied Physics Letters* 96, 172103 (2010)
17. Kursawe, K., Sadeghi, A.-R., Schellekens, D., Skoric, B., Tuyls, P.: Reconfigurable Physical Unclonable Functions – Enabling Technology for Tamper-Resistant Storage. In: *HOST 2009*, pp. 22–29 (2009)
18. Rührmair, U.: Oblivious Transfer based on Physical Unclonable Functions (Extended Abstract). In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) *TRUST 2010*. LNCS, vol. 6101, pp. 430–440. Springer, Heidelberg (2010)

19. Suh, G.E., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: DAC 2007, pp. 9–14 (2007)
20. 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)
21. 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)
22. 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)
23. Carter, R.J., Nemanich, R.J.: HF vapour cleaning of oxide on c-Si. Properties of Crystalline Silicon. EMIS Datareviews Series No 20, University of Virginia, USA (1999)
24. Majni, G., Ottaviani, G.: Growth kinetics of (111)Si through an Al layer by solid phase epitaxy. *Journal of Crystal Growth* 46, 119 (1979)
25. Lim, D.: Extracting Secret Keys from Integrated Circuits. MSc Thesis, MIT (2004)
26. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing Techniques for Hardware Security. In: IEEE International Test Conference (2008)

# Peeling Away Layers of an RFID Security System

Henryk Plötz<sup>1</sup> and Karsten Nohl<sup>2</sup>

<sup>1</sup> Humboldt-Universität zu Berlin

<sup>2</sup> Security Research Labs, Berlin

**Abstract.** The Legic Prime system uses proprietary RFIDs to secure building access and micropayment applications. The employed algorithms rely on obscurity and consequently did not withstand scrutiny.

This paper details how the algorithms were found from opening silicon chips as well as interacting with tags and readers. The security of the tags is based on several secret check-sums but no secret keys are employed that could lead to inherent security on the cards. Cards can be read, written to and spoofed using an emulator. Beyond these card weaknesses, we find that Legic's trust delegation model can be abused to create master tokens for all Legic installations.

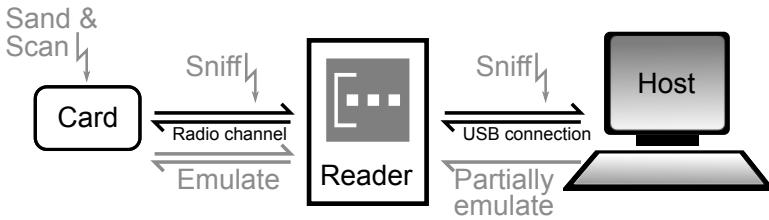
## 1 Introduction

The “Legic Prime” RFID card is used for access control to buildings throughout Europe including critical infrastructure such as military installations, governmental departments, power plants, hospitals and airports. Despite its use in high security installations, access cards can be cloned from a distance or newly created using a spoofed master token.

The Legic Prime cards use proprietary protocols and employ simple check-sums, which have not previously been revealed. This paper discusses how the proprietary protocol and crypto functions were found using a combination of silicon reverse engineering and black box analysis. Since the cards do not employ cryptographic encryption or authentication, knowledge of the proprietary protocol alone allows for cards to be read, written to, and spoofed.

Legic's Prime technology is unique among RFID access technologies in several respects: The Prime chip is the oldest RFID card to use the 13.56 MHz band, it is the most mysterious for none of its protocol is documented, access to Legic hardware is closely guarded, and Legic cards have long been the only ones supporting trust delegation to model organizational hierarchies. We find that due to its lack of public documentation, weaknesses in the cards have gone unnoticed for two decades. The weaknesses allow for Prime cards to be fully cloned with simple equipment and for the trust delegation to be circumvented, which enables an attacker to create Legic tokens for all installations where the technology is used.

Besides for access control, Prime cards are also used as micropayment tokens in cafeterias, resorts and public transport. Money stored on the cards can be



**Fig. 1.** Legic Prime system overview. Our analysis techniques are marked in light grey.

stolen from a distance. Legic's Prime technology must be considered insecure for its intended applications and should be replaced with cards employing peer-reviewed cryptography in open protocols.

The paper makes three main contributions by showing that:

1. Reverse engineering is possible with simple tools even for undocumented systems with multiple layers of obfuscation.
2. Legic Prime is insecure since attacks exists to clone tags and to spoof chains of trust.
3. While the trust delegation model in Legic is insecure, the same concept could be implemented securely using hash trees.

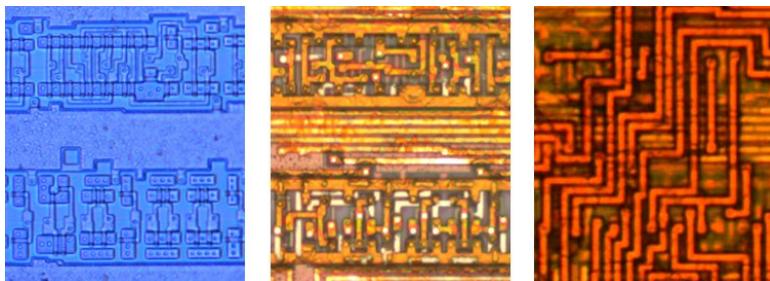
A general overview of the system that we worked with and our analysis techniques are given in Fig. 1. We partly analyzed the USB protocol and used that knowledge as a stepping stone to experimenting with the radio protocol. Concurrently silicon analysis took place on the chip embedded in the card.

The following section illustrates our two complementary approaches to reverse engineering RFID systems. Section 3 documents the Legic Prime card layout and protocol and points out several weaknesses. The concept of a trust hierarchy is introduced in Section 4 along with a discussion of why Legic's implementation is insecure while secure implementations are not hard to built.

## 2 RFID Reverse Engineering

Embedded computing systems such as RFID tokens often use proprietary protocols and cryptographic functions. The details of the algorithms are typically kept secret, partly out of fear that a system compromise will be easier once its operation principles are known. We found that the Legic Prime security system does not provide any inherent security beyond this secrecy.

A necessary first step in the security assessment of a proprietary system is reverse engineering of its functionality, which is achieved using one of two methods: a) Reverse engineering circuits from their silicon implementation as the more cumbersome method that is almost guaranteed to disclose the secret functions; b) Black-box analysis that can often be executed faster but requires prior information about the analysed system or lucky guesses. We employed both approaches in analyzing Legic Prime.



**Fig. 2.** Layers of a silicon chip: transistor layer, logic layer and one of the interconnect metal layers

## 2.1 Silicon Reverse Engineering

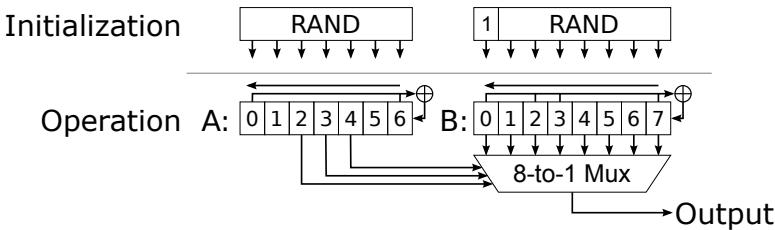
Disclosing secret algorithms from an RFID chip through reconstructing circuits has been demonstrated before when the Crypto-1 cipher was extracted from a Mifare Classic chip [5]. This project produced a suite of image analysis tools that work on images of the silicon chips. These images of the different chip layers are produced by polishing down the chip – a micrometer at a time – and photographing with an optical microscope. The tools then automatically detect recurring patterns in images such as those shown left in Figure 2. These patterns represent logic functions that are used as building blocks for algorithms similar to instructions in an assembly language.

The tools further support semi-automatically tracing of wires between the logic gates that disclose the chip circuit. In the case of Mifare Classic, around five hundred gates (out of a larger chip) and the connections among them were documented to disclose the encryption function [5]. In case of Legic Prime, the entire tag only consists of roughly five hundred gates, less than one hundred of which form a key stream generator. This key stream generator and part of the protocol initialization (described later in the paper) are detailed in Fig. 3. Reverse engineering the circuit of the entire digital part of the Legic Prime chip took two weeks<sup>1</sup>.

The same reverse engineering techniques extend to any silicon chip including smart cards and trusted platform modules. The effort scales with the size of a chip; fully reverse engineering a microprocessor with millions of gates, for example, seems impractical with the current tools. However, security functions such as the on-chip encryption of smart cards are often small separated entities that can be imaged and disclosed independent from the rest of the chip. Currently the best protections against silicon reversing are randomly routed chips that mix security and other functionality, and chips of small feature sizes that require equipment more expensive than optical microscopes for imaging.

---

<sup>1</sup> The tools for silicon analysis (**Degate**) and the whole Legic Prime circuit can be downloaded at <http://degate.org/>



**Fig. 3.** The Legic Prime key stream generator and initialization function

## 2.2 Black Box Analysis

Complementary to silicon reverse engineering, tests are conducted on the running system to add missing information. These two steps are roughly equivalent to disassembling and debugging in the software domain.

Given a functioning system – consisting of host software, a reader and cards – the goal of black box analysis is to learn system details by a) observing the behavior of the system on its communication channels in a passive manner, then b) trying to emulate parts of the system based on the observations from the first step, and finally c) varying the emulation by deviating from already observed behavior in order to observe new behavior. This approach was previously used for reversing the Texas Instruments DST40 cipher and protocol [4].

Our experimental setup consisted of a host PC running original Legic provided software and custom scripts. We wrote the scripts based on observations on the USB interface to replace functionality of the original software without necessarily abiding to its constraints.

As hardware we used a genuine Legic reader, a Proxmark3 RFID tool [3], and a logic analyzer. The Proxmark device was fulfilling multiple purposes: In the simplest case it is a short range RFID sniffer that outputs a demodulated carrier envelope signal on a debug test point to which we connected the logic analyzer. This setup allows for the traces of the communication between the original reader and original cards to be recorded with the stock Proxmark firmware. We could then look at the recorded waveforms on the PC for a rough visual, qualitative inspection and use custom scripts for further analysis of the data with the goal of decoding frames.

An initial survey showed that Legic had already published parts of its lower level protocols when applying for standardization as ISO 14443 Annex F [1]. While the application was rejected the material is still available online [2]. Using this information we were able to separate our trace into frames from reader and card and decode each frame to a bitstream. Later we were also able to write Proxmark3 firmware with the ability to receive and send frames, both in reader and card emulation mode. Without the ISO application document many of the modulation and encoding parameters would have had to be guessed. However, the parameter choices for the Legic Prime protocols are straight-forward (see Section 3) and could have been guessed with at most a few tries (long vs. short modulation from the reader, modulation vs. no modulation from the card).

**Replay.** Legic Prime allows for transactions to be replayed without necessarily understanding the protocol details. This is made possible by the lack of a random number from the card side, which happily accepts any past transaction. More importantly, from an attack perspective, transactions can also be replayed to a reader since the random numbers involved are weak. Replaying a transaction has an average success probability of up to 10% and the replay can usually be tried several times (e.g. at a door reader) until accepted (door opens).

**USB protocol.** To generate lots of similar RF traces for comparison we first partly reverse engineered the USB protocol used between the Legic host software and the Legic reader. The reader is connected to the host using an FTDI USB-to-serial converter (with a custom vendor and product id). We used usbsnoop for Windows to record all exchanges between host and reader. We then used custom scripts to extract the serial communications stream. The general frame format is similar to other Legic readers for which we found documentation on the internet<sup>2</sup>: each frame has a length byte, a body and a Longitudinal Redundancy Check (LRC, simple XOR over all the bytes including the length byte). The length byte does not count itself. When analyzing the USB communication we found one command that always preceded all other commands and seemed to return the UID of the card in the field: 02 B0 B2. We designated this command “GET UID” and wrote custom software to repeatedly (and rapidly) send this command to the reader on its USB channel.

**RF protocol: UID command.** A single GET UID command will try to enumerate cards of all supported protocols in the field: LEGIC RF, ISO 14443-A and ISO 15693. Using our sniffing setup we looked at the LEGIC RF portion of this sequence to understand the general layout of the protocol. We have not investigated the remaining two protocols, which are used by Legic’s Advant tags.

Even for the same card, different GET UID sequences looked very different. Each sequence starts with a 7-bit frame from the reader, which seems to be mostly random, but always has the first bit set to 1. We designated this frame “RAND” since it looked like the initialisation vector of a stream cipher: For all transactions with the same RAND, the sequence of reader commands is identical, while the sequences of card responses are identical only for identical UIDs.

By comparing multiple traces and looking for the first and last modulation observed from the card we found the following general structure in the protocol: 7 bits from reader, 6 bits from card, 6 bits from reader, then five repetitions of 9 bits from reader, 12 bits from card. This structure was observed for a 256byte card (MIM256); a MIM1024 card receives 11 instead of 9 bits from the reader. We named the 7-6-6 part the ‘setup phase’ and the remainder the ‘main phase’. The contents of the setup phase only depend on the RAND frame. The remainder is always identical (within a card type; a MIM1024 card has one bit flipped in the 6-bit card response). So far the protocol looked like an authenticating stream cipher, with weak initialization vector from the reader. The random numbers

---

<sup>2</sup> [http://www.rfid-webshop.com/shop/product\\_info.php?info/p318-LEGIC-Plug---Play-module.html/](http://www.rfid-webshop.com/shop/product_info.php?info/p318-LEGIC-Plug---Play-module.html/)

from the readers are not only short but also statistically biased: 0x55 appears in roughly 1 out of 10 tries. Here, and in the following discussion, frames are represented as single integers transmitted in LSBit-first order.

Since the GET UID sequence must contain the UID of the card we XORED two traces from cards with different UID (but same RAND) to learn about the order of transmission. We found that the first card response in the main phase contained the first byte of the UID in the lower 8 bits, and something else (probably a checksum) in the higher 4 bits. The second card response contained the second byte of the UID, and so on<sup>3</sup>. Since there are only four bytes to the UID, but five data transmissions in the main phase for GET UID, we assumed that the fifth transmission would be some kind of checksum, most likely a CRC-8, which we called the storage CRC.

In order to further test our hypotheses and learn more about the system we implemented a card and a reader emulator which could replay previously recorded frames. In the first attempts we replayed the sniffed frames verbatim. This was made possible by the absence of any random number from the card – a reader emulator can completely play back a recorded transaction – and a weak random number from the reader – a card emulator can completely replay certain transactions with  $\sim 10\%$  probability. Completely replaying frames worked flawlessly (as long as the timing of the original trace was followed precisely), so next we experimented with changing single bits in the main phase of the replayed traces, without touching the setup phase.

Flipping a single bit in the card responses in the main phase would make the reader abort the session, clearly indicating the presence of a checksum. Since we already knew that the data is transmitted in the lower 8 bits, the checksum must be in the high 4 bits, most likely some form of CRC-4, which we called the transport CRC. Since there are only 16 possibilities we opted for a quick brute force approach: Flip one bit in the data section, then try all 16 flip variants on the CRC section to find the variant where the reader would not abort the session. Using this approach we found a table with 8 entries of 4 bits each that would allow us to correctly fix the CRC for an arbitrary change of the data section, given a trace with correct transport CRCs. This approach works since CRCs are linear:  $\text{CRC}(a \oplus b) = \text{CRC}(a) \oplus \text{CRC}(b)$ , under some preconditions (see [7]).

With this table we were able to send arbitrarily modified card responses (based on our initial guess to what the responses would mean) that were accepted by the reader. The reader would still not accept the complete UID, because of the checksum in the fifth byte. We attacked this byte in a similar manner, yielding a table of 32 entries with 8 bits each, allowing to spoof an arbitrary UID in the GET UID sequence. This validated our assumptions on the meaning of all the parts of the card responses and further provides known keystream for all the UID data bytes.

---

<sup>3</sup> Different versions of the official host software display the UID in different formats. Older versions display the UID in order first byte, fourth byte, third byte, second byte. This seems to relate to a structure in the UID: The first byte is a manufacturer code, and the remaining three bytes are treated as a LSByte-first integer. We use the transmission and storage order in this paper.

From the CRC tables we could derive the used CRC polynomial: Since the CRC is over known data with only a single bit set, the differences between the different entries in the table differ only by the amount of shifts in the CRC calculation. Whenever a 1-bit is shifted out, the CRC polynom is XORed onto the state. By looking for these two properties in our tables we found the transport CRC polynomial to be 0xc and the storage CRC polynomial to be 0x63 (but with a reversed shift direction).

**RF protocol: reading memory.** Based on the simplicity of the protocol so far observed we formed the following hypothesis about the reader commands: The GET UID sequence is not really requesting the UID (e.g. such as anticollision ISO 14443) but simply reading the first 5 bytes of memory. Each reader command in the GET UID sequence is a “read byte  $x$ ” command. Since the command is 9 bits, and 8 bits are necessary to address all 256 bytes on a MIM256 card, that leaves 1 bit for the command code, namely “read” in this case.

We verified this assumption by replaying modified frames with correct timing: changing the first bit of the reader command would make the card never respond, while changing any other bit or bit combination would always lead to a response. This confirmed three things:

- The first bit is the command code and only a correct command code will lead to a response. The other command, presumably “write” must have another frame format.
- The remaining bits (8 for MIM256, 10 for MIM1024) are the address.
- The entire memory space can be read, with no restrictions (there is always a response, no matter the address).

Using the recorded first command of the GET UID sequence (which we now know is “read byte 0”), known keystream from its response and by changing the address in that command, we were able to completely dump the contents of any card; including unused and deleted segments.

**RF protocol: key stream obfuscation.** Next we sought to understand a phenomenon that we encountered early on in the implementation phase of the reader emulator: Not exactly following the timing of the recorded traces would sometimes make the card not respond. We concluded that the cipherstream generator must be continuously running, after the end of the setup phase, and replaying a recorded frame with some offset against the recorded timing would lead to a completely different frame on the card side, after decoding. In the instances where the card would not respond this changed frame was invalid. Since we know that the card will always respond to a command with a ‘read’ command code we could assume that the non-responding cases were those where the command code bit was received different from the ‘read’ command code.

Following up on the assumption of the continuously running cipherstream generator we leveraged existing known keystream to find new known keystream. First we determined the clock of the generator to be around  $99.1\ \mu s$ , which approximately matches the bit duration in card originated frames. We did this by a simple sweep over the possible delays before the first command in the

main phase, and then sending a fixed command (all 1). Since the card responds if, and only if, the first bit of the received and decoded command is a correct read command code we could determine the current output of the keystream generator at the start of the command frame: Using our known keystream we found that the ‘read’ command code is 1, so when we got a card response for some delay value we knew that the cipherstream at that point started with 0.

By repeating this experiment for many different delay values (while always powering the card down between two trials) we got a time series that clearly showed the cipherstream output (with transitions only every  $99.1\mu s$ ). Using this method it is possible to use the card as an oracle to generate cipherstream for any setup phase. From reverse engineering the silicon chip we knew that the stream generator has only 15 bits of state, so the stream repeats after  $2^{15} - 1 = 32767$  bits. Reconstructing the complete stream from the card responses would take approximately 14 hours (due to the wait times incurred by powering down the card). This key stream can then be used to fully emulate a card or a reader without knowledge of the key stream generator, which we gained from the silicon chip.

By this point it is also clear that there is no key input to the stream generator since the recorded stream is portable between any card and any reader. We will no longer refer to it as an encryption (which it is not due to a lacking key) but only obfuscation function. Other radio protocols such as Bluetooth have similar mechanisms to enhance physical radio properties, called whitening.

### 3 Logic Prime Protocol

The rejected ISO 14443 annex F describes the lower layer radio protocol of Logic RF: Reader to card is 100% ASK with pulse-pause-modulation (1-bit is  $100\mu s$ , 0-bit is  $60\mu s$ , pause is  $20\mu s$ ), card to reader is load modulation on a  $f_c/64$  ( $\approx 212\text{kHz}$ ) subcarrier with bit duration  $t_{\text{bit}} = 100\mu s$  (subcarrier active means 1-bit). The annex does not specify the framing of card originated frames, merely stating that it is “defined by the synchronization of the communication”. From our observations we found this to mean that the card frame starts at a fixed time after the reader frame (this time was measured to be  $\sim 330\mu s$ , which approximately equals  $3 t_{\text{bit}}$ ) and that the reader must know in advance how many bits the card will send, since there is no explicit frame end indication. Most notably this also means that not sending a frame (e.g. due to no card present, or card removed) is indistinguishable from sending a frame of only 0-bits.

The protocol consists of two phases: setup phase and main phase. The setup phase starts with the reader sending an *initialization frame* RAND of 7 bits (in LSB-first order) with the lowest bit set to 1. At this point the obfuscation stream generator is started by setting  $\text{LFSR}_A := \text{RAND}$  and  $\text{LFSR}_B := (\text{RAND} \ll 1) | 1$  and all further communications are XORed with the current generator output. When no frame is being transmitted the generator generates one new bit every  $t_{\text{bit}}$ . When a frame is transmitted the generator is clocked with the data bit clock (this especially applies to reader originated frames which can have bit durations that are smaller than  $t_{\text{bit}}$ ). We found the generator

initialization by trying different obvious variants of assigning RAND to the generator registers and comparing the output to the known obfuscation stream output from the previous section. Knowledge of the generator and initialization made it possible to completely deobfuscate all recorded traces of communication between the official Legic software, reader and card and observe all the remaining protocol specifics.

The card responds to the RAND frame, after a wait time of  $3 t_{\text{bit}}$ , with an obfuscated *type frame* of 6 bits. This frame is either `0xd` for MIM22, `0x1d` for MIM256 or `0x3d` for MIM1024. The reader must wait at least one  $t_{\text{bit}}$  before sending its obfuscated *acknowledgment frame* of 6 bits. This is `0x19` for MIM22 and `0x39` for MIM256 and MIM1024. After this frame is sent the setup phase is complete and the main phase starts.

In the main phase the reader can send commands at any point in time. Each command has an address field of either 5, 8 or 10 bits (for MIM22, MIM256, or MIM1024 respectively). The following discussion only covers the 8-bit-address case, which is the most common card variant.

There are two types of commands: Read and Write. A read command consists of one bit command code 1, followed by the address. After a waiting time of  $3 t_{\text{bit}}$  the card will respond with 12 bits: The first 8 bits are the data byte from the transponder memory at the given address (LSBit first), the next 4 bits are the transport CRC-4. The transport CRC-4 is calculated with polynomial `0xc`, initial value `0x5`, and is calculated over the command code, address and data byte.

A write command consists of the command code 0, the address, 8 bit data and 4 bit transport CRC-4. The transport CRC is calculated as above (just that the command code is now 0). If the write is successful, the card will respond with an ACK: a single unobfuscated 1-bit. The time until the ACK can vary, but will be approx. 3.5 ms.

### 3.1 Card Layout

The memory space of a Legic Prime transponder is separated into three distinct physical zones: the UID (with its CRC), which is read-only, the decremental field (DCF), which, taken as a little endian integer, can only be decremented, and the remainder of the card, which can be freely written to. The entire memory space can always be read from. There are two variants for the logical organization of the card's payload data: unsegmented media (with master token being a special case), which contain exactly one segment, and segmented media, which can contain multiple segments. The protection features (on a reader firmware level) for both kinds of segments are essentially identical, and the headers are very similar. For this reason the rest of the paper will only consider the segment headers on segmented media, which now make up most of the market, and will cover master token as a special case of unsegmented media.

The general layout of a segmented Legic Prime medium is shown in Fig. 4. The remainder of the card that is not shown in the figure, starting at byte 22, contains the payload segments.

UID <sub>0</sub>	UID <sub>1</sub>	UID <sub>2</sub>	UID <sub>3</sub>	CRC <sub>UID</sub>	DCF <sub>lo</sub>	DCF <sub>hi</sub>	9F
FF	00	00	00	11	BCK <sub>0</sub>	BCK <sub>1</sub>	BCK <sub>2</sub>
BCK <sub>3</sub>	BCK <sub>4</sub>	BCK <sub>5</sub>	CRC <sub>BCK</sub>	00	00		
payload							
⋮							

**Fig. 4.** Legic Prime card layout of segmented medium, containing a unique identifier (UID), decremental field (DCF), and a segment header backup (BCK) with its own CRC.

The payload area is obfuscated with the CRC of the UID: All bytes, beginning with byte 22, are XORed with CRC<sub>UID</sub> (address 4). Within the payload area the different segments are stored consecutively and each segment starts with a five byte segment header. This header consists of

- byte 0: lower byte of segment length, segment length includes the 5 bytes for the segment header
- byte 1, bits 0-3: high nibble of segment length
- byte 1, bit 6: segment valid flag, if this flag is not set, the segment has been deleted
- byte 1, bit 7: last segment flag, if this bit is set, no more segments are following
- byte 2: WRP, “write protection”
- byte 3, bits 4 through 6: WRC, “write control”
- byte 3, bit 7: RD, “read disabled”
- byte 4: CRC over the segment header

The different protection features are implemented in the firmware of all official Legic readers. To this end the data portion of a segment usually starts with the stamp of that segment, with the length of the stamp contained in the WRC field. A reader will compare this stamp to an internal database of stamps that it is authorized to operate on and then behave accordingly: write access is only allowed if the reader is authorized for that stamp. If the RD flag is set and the reader is not authorized for the stamp, then it will not allow any read access to the segment (including to the stamp). A reader will never allow write access to the bytes protected by the WRP field. A reader emulator can ignore all of these rules.

When writing a segment header, the official readers follow a special backup procedure to ensure that the segment structure cannot be corrupted by prematurely removing the card. Before changing a segment header which is included in the existing segment chain (e.g. all segment headers up to and including the first header that has the ‘last’ flag set), the complete header, including the CRC, is copied to BCK<sub>1</sub> through BCK<sub>5</sub>, BCK<sub>CRC</sub> is set to the CRC over BCK<sub>0</sub> through BCK<sub>5</sub> (note that BCK<sub>0</sub> has not been written yet). Then BCK<sub>0</sub> is written: bit 0 through 6 contain the number of the segment header (with 1 being

the first segment header, this limits a card to maximal 127 segments) and bit 7 is a ‘dirty’ flag, which is set. Only after the backup area has been written will the actual segment header be changed. As soon as the new header is completely written, the backup area is invalidated by clearing the ‘dirty’ flag. Should the write to the segment header be interrupted, a reader will notice the flag next time when the card is presented and restore the the original segment header and then clear the flag. This procedure guarantees that any single change to any header is always handled atomically.

### 3.2 Weaknesses

Most protection functions are implemented in the firmware of the official Legic readers. The only hard protections in the card are the read-only state of bytes 0 through 4 (UID) and the decrement-only logic of bytes 5 and 6. Everything else on the card is freely read- and writable with a custom reader that ignores the protection flags. For all applications that do not explicitly check the UID it is possible to directly copy data from one card to another, as long as one fixes the payload obfuscation and CRCs (which both depend on the UID). Moreover there is no keyed authentication involved anywhere, so a clean dump and full emulation of cards are possible to trick even application that do check for the UID.

Since no keys are involved it is then also possible to spoof new, non-existing cards, including master tokens. Spoofed master-tokens can simplify attacks since with them it is not necessary to reverse engineer the complete payload format of an application: one can simply use an existing, official reader for that application and make it believe that it is authorized to work on the cards to be attacked.

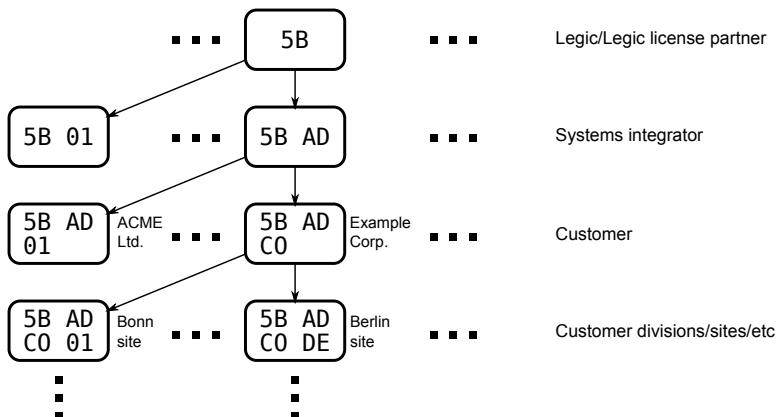
Legic Prime poses an unusually large skimming risk, since, unlike most other card types, there is no read protection. A reader can read any card that is in its range. Because Prime was developed with read range in mind and uses very low power due to its simplicity, skimming ranges above what is common with ISO 14443 should be possible. The manufacturer variously states up to 70 cm read range for their official readers.

We also observed that the reader would usually only do the absolutely minimum changes necessary to modify the linked segment structure. For example when deleting a segment it will only clear the ‘valid’ bit in the segment header and not clear the segment payload. Also the backup area is never cleared after use. This means that a custom reader can gather much more data from a card than an official Legic reader: Most ‘deleted’ segments will still be intact and through the backup area there is a trace that shows which segment header was changed last.

## 4 Legic Trust Delegation

### 4.1 Card Hierarchy Concept

Legic systems are designed as a replacement for mechanical locking systems and implement not only their functional properties but also the organizational



**Fig. 5.** Legic card trust delegation: The length of a card's stamp encodes its level in the hierarchy

concepts of locking systems. The cards are organized in a hierarchy that represents the distribution path from Legic to the customer as well as the permission hierarchies within the customer's organization.

Cards that are higher in this virtual hierarchy can produce all cards below them. For example, a distributor can generate master tokens for all of its customers, even after the system was fully deployed; and Legic itself can generate tokens for all distributors. By delegation of trust, Legic can clone any card in existence or create new cards for any system since all cards are part of the same trust tree. While it is arguable whether a single company should have this level of control over its customers' systems, the trust tree implementation of Legic Prime allows for more concerning attacks: Anybody can move to the highest level of the tree thereby gaining control over all Legic Prime systems.

## 4.2 Legic Prime Implementation

Legic Prime distinguishes at least three types of master token:

- General Authorization Media (GAM), to create further master tokens
- Identification Authorization Media (IAM), to create segments on cards
- System Authorization Media (SAM), to transfer read/write authorizations to readers

Each node in the trust delegation hierarchy has an identifier which is called the stamp (sometimes also “genetic code”). When creating a child node at least one byte is appended to the stamp, so nodes that are farther down in the hierarchy have longer stamps, see Fig. 5 for an example hierarchy<sup>4</sup>. Authorization decisions are made by a prefix match: A reader that is authorized to read segments with stamp **5B AD** is also authorized to read segments with stamp **5B AD CO**.

<sup>4</sup> The stamp **5B AD CO DE** was chosen as a fictional example. Any resemblance to any real-world stamps is purely coincidental.

Creation of sub-tokens is controlled by another bit: Organization Level Enable (OLE). Only when this control bit is set is a master token authorized to create sub-tokens. A GAM (with OLE=1) can create any non-GAM master token with the same stamp as its own, or any master token with a stamp that is longer than its own and has the same prefix. IAM and SAM (both with OLE=1) can create IAMs and SAMs, respectively, with a stamp that is longer than their own and has the same prefix.

A master token is a special case of an unsegmented Legic medium which uses 22 bytes of data (exactly fits a MIM22 card, but can be written to larger media). The token type is encoded in the DCF, which has two main consequences: a master token cannot at the same time be a normal segmented medium and there are certain restrictions when writing master tokens to original Legic cards. For easier experimentation we performed the analysis of the master token structure with our card emulator which was implemented on the Proxmark and which allowed free change of the DCF. We first emulated the verbatim contents of a real master token and then performed incremental changes to the data, followed by a read using the official Legic software, in order to determine the meaning of the different fields.

For master tokens the header is interpreted as follows:

- $\text{DCF}_{\text{lo}}$  (byte 5), bit 7: OLE, organization level enable flag
- $\text{DCF}_{\text{lo}}$ , bits 0 through 6: token type: 0x00-0x2f IAM, 0x30-0x6f IAM, 0x70-0x7f GAM
- $\text{DCF}_{\text{hi}}$  (byte 6): must be 0xfc - (stamp length), indicates level in the hierarchy
- byte 7, bits 0 through 3: WRP, contains the stamp size
- byte 7, bits 4 through 6: WRC, on SAM cards contains the number of stamp bytes that will be stored in the internal authorization database
- byte 7, bit 7: RD, not set for master token
- bytes 8...: stamp, variable length
- byte 21: CRC-8 over  $\text{UID}_{0..3}$ ,  $\text{DCF}_{\text{hi}}$ ,  $\text{DCF}_{\text{lo}}$ , byte 7, byte 8...

Note that this is the same general format as with unsegmented non-master token media (though we've only seen one such medium to date). For all non-master token the highest DCF value observed was 0xEA60 which is less than the minimal DCF value for a master token (0xF000 for an IAM in level 12), so this field alone gives the distinction between master and non-master tokens. There might be other interpretations for the DCF: We didn't perform a comprehensive search over the DCF space, since the Legic software is rather picky and crashes when it encounters an unexpected DCF value.

The multiple possible values for the same master token type in the same hierarchical level can perform different functions. Setting  $\text{DCF}_{\text{lo}}$ , bits 0 through 6 to 0x31 gives a SAM63 in Legic lingo, while setting it to 0x30 gives a SAM64. SAM63 is also known as "Taufkarte", or launching card, while SAM64 is an "Enttaufkarte", or delauching card. This refers to the processes of storing and deleting the authorization for a stamp in a reader which are known as "taufen"/launch and "enttaufen"/delaunch, respectively.

### 4.3 Weaknesses

Due to the way the DCF field is used, an existing master token cannot be freely modified and indeed can only be changed into a lower level master token. However, no such restriction applies when writing to a fresh card (or when using a card emulator). On a new card the DCF field is set to `0xFFFF` so it is possible to write any master token to such a card. Within the normal Legic system there are protections in the reader firmware that prevent creating a master token without the proper authorization, but no such mechanism applies when using a custom reader.

This means that master tokens can be freely copied to empty cards, and indeed freely generated for any desired stamp value. We also found that the prefix match for IAM and SAM is unrestricted: We have created an IAM of stamp length 0, which will prefix-match any stamp value. This Uber-IAM can authorize an official reader to read and write segments with arbitrary stamps. It is, however, not possible to launch a reader with a SAM with stamp length 0, since WRC must be  $\geq 1$  for a launch process to take place. Also GAMs with stamps shorter than 2 bytes seem to be specifically locked out in the host software: when trying to load such a GAM, the software will stall for a few seconds and then pretend that the card was empty.

The problem that master tokens can be cloned is inherent in the work flow of the Legic system (and therefore most likely also present in Legic Advant): In order to use a master token to create a new segment, first the master token (an IAM or GAM) is presented to the reader, which then stores the authorization information internally. Afterwards the reader will allow, until a configurable timeout occurs, to create segments on normal cards with this stamp or a longer stamp. The master token is not inherently necessary for the segment creation: by the time the segment is actually being created, the master token can long be back in a safe. This clearly shows that the complete ‘essence of being’ of that master token has been transferred into the reader, which means it’s possible to read out and store all the data that is necessary to perform the functions that the master token allows. In the case of Legic Prime this only includes the stamp value and token type, but even if there was cryptographic key material in the master token, as might be the case with Legic Advant, it must be exportable. This export process cannot use any strong authentication, since the reader and the token share no previous association with each other, and the user is not prompted for a PIN or similar.

### 4.4 Improvement Potential

The concept of organizing access credentials in a trust hierarchy is not flawed in itself. In fact, the same idea is used very successfully –and securely– in virtual credential systems such as Microsoft’s Active Directory service. For an implementation of trust delegation to be secure, the delegation process has to be one-way in the sense that only higher level entities are trusted. Legic’s current implementation clearly is two-way as stamps can be shortened and lengthened at will, thereby moving up and down the trust hierarchy.

A sensible system would use a one-way function such as a cryptographic hash function to assure that the access credentials of tokens cannot be elevated. The stamp of a lower level card would be created as

$$\text{stamp}_{\text{child}} = \text{Hash}(\text{stamp}_{\text{parent}}, \text{metadata}_{\text{child}})$$

where the metadata is used to distinguish several child cards. Going one step beyond the functionality of the Legic Prime system, these stamps would not be exchanged in cleartext but rather used as secret keys in a strong encryption function such as 3DES or AES, which are available on several modern RFID tags. The idea of using hash trees for authentication hierarchies has already been discussed as early as 1988, in [6].

## 5 Conclusion

Systems must not rely on obscurity but should rather employ cryptography as a base for security functions. The Legic Prime security chain breaks in two places where cryptography is missing. First, cards and readers cannot authenticate each other, which allows an attacker to assume either role and read, write, or spoof cards and readers. Secret keys and a simple encryption or hash function would mitigate these problems.

The second place where the lack of cryptography enables attacks against Legic systems is their unique trust delegation model. Since no secret information exists that could distinguish higher permission from lower levels, any of the levels can be spoofed. The idea of having secure trust delegation, however, models many organizations' needs very well, which may have contributed to the popularity of Legic cards. Creating a system with secure delegation features is left for further research and development.

## References

- Article “ISO14443” in the openpcd wiki, section “LEGIC RF”, revision as of 00:32 (September 6, 2010), [http://www.openpcd.org/index.php?title=ISO14443&oldid=193#LEGIC\\_RF](http://www.openpcd.org/index.php?title=ISO14443&oldid=193#LEGIC_RF)
- ISO 14443 Part 2 Amendment 1, dRAFT 2nd P-DAM BALLOT TEXT
- PROXMARK III community, <http://www.proxmark.org/>
- Bono, S., Green, M., Stubblefield, A., Juels, A., Rubin, A., Szydlo, M.: Security analysis of a cryptographically-enabled RFID device. In: USENIX Security Symposium (2005)
- Nohl, K., Evans, D., Starbug, Ploetz, H.: Reverse-engineering a cryptographic RFID tag. In: USENIX Security Symposium (2008)
- Sandhu, R.S.: Cryptographic implementation of a tree hierarchy for access control. Information Processing Letters 27(2), 95–98 (1988), [http://dx.doi.org/10.1016/0020-0190\(88\)90099-3](http://dx.doi.org/10.1016/0020-0190(88)90099-3)
- Stigge, M., Plötz, H., Müller, W., Redlich, J.P.: Reversing crc-theory and practice (2006), [http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2006-05/SAR-PR-2006-05\\_.pdf](http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2006-05/SAR-PR-2006-05_.pdf)

# Might Financial Cryptography Kill Financial Innovation? – The Curious Case of EMV

Ross Anderson, Mike Bond, Omar Choudary,  
Steven J. Murdoch, and Frank Stajano

University of Cambridge Computer Laboratory,  
15 JJ Thomson Avenue, Cambridge CB3 0FD, UK  
`name.surname@c1.cam.ac.uk`

**Abstract.** The credit card system has been one of the world’s great successes because of its adaptability. By the mid-1990s, a credit card had become a mechanism for authenticating a transaction by presenting a username (the card number) and a password (the expiry date, plus often a CVV) that was already used in mail order and could be adapted with little fuss to the Internet. Now banks in Europe, and increasingly elsewhere, have moved to the EMV “Chip and PIN” system which uses not just smart cards but also “trusted” hardware. The cryptography supported by this equipment has made some kinds of fraud much rarer – although other kinds have increased, and the jury is still out on the net effect. In the USA in particular, some banks and others oppose EMV on the grounds that it will damage innovation to move to a monolithic and inflexible system.

We discuss the effects that cryptographic lock-down might have on competition and innovation. We predict that EMV will be adapted to use cards as keys; we have found, for example, that the DDA signature can be used by third parties and expect this to be used when customers use a card to retrieve already-purchased goods such as air tickets. This will stop forged credit cards being used to board airplanes.

We also investigate whether EMV can be adapted to move towards a world in which people can use bank cards plus commodity consumer electronics to make and accept payments. Can the EMV payment ecology be made more open and competitive, or will it have to be replaced? We have already seen EMV adapted to the CAP system; this was possible because only one bank, the card issuer, had to change its software. It seems the key to innovation is whether its benefits can be made sufficiently local and incremental. We therefore explore whether EMV can be adapted to peer-to-peer payments by making changes solely to the acquirer systems. Finally, we discuss the broader issue of how cryptographic protocols can be made extensible. How can the protocol designer steer between the Scylla of the competition authorities and the Charybdis of the chosen protocol attack?

## 1 Introduction

The credit card system has supported innovation, both internally and externally, for over half a century. In fact, that’s why they succeeded in the first place.

During the 1960s and 1970s, they competed with cheque cards that a customer could use to guarantee a cheque to any merchant up to a certain amount. Credit cards won this competition, and a key reason was that they had the flexibility to adapt to mail order and telephone order sales. The card companies found that they'd built not just a settlement system but a global system of authentication where the user name was the card number, and password was the expiry date (joined from the early 1990s by CVVs).

With the arrival of electronic commerce in the mid-1990s, companies such as Microsoft and Netscape tried to design proper cryptographic protocols to support payments (SEPP and STT, amalgamated into SET). However these protocols would have cost time and money to deploy and could not in practice compete with simple credit card transactions which were already available through a deployed infrastructure into which the new e-commerce websites could feed traffic on exactly the same basis as the existing mail-order firms with which they competed. Flexibility won out once more.

There is now a row brewing over the new EMV chip card system developed jointly by Europay, MasterCard and Visa through EMVCo from 1995 onwards. Over the past five years EMV has been deployed in most of Europe and is starting to appear in other countries such as Canada and India. But the USA remains obdurate. US opponents of EMV discuss, *inter alia*, the question of innovation. Can a much more complex payment system such as EMV adapt to new circumstances and to new market opportunities, or will it fossilise as a platform? Getting hundreds of vendors, thousands of banks and millions of merchants to change their systems simultaneously is extremely hard – that's why it took twenty years to get smart card payments going in the first place!

Yet we have already seen one cycle of innovation. The growth of phishing attacks on electronic banking since 2005 led to the development of the Chip Authentication Program (CAP). This protocol enables a bank to use its issued base of EMV cards to generate codes for two-factor authentication: it issues each customer with a small low-cost reader, and when the customer logs on to the bank website she is asked for an authentication code which she can generate by inserting her card in the CAP reader and typing her PIN. (We described CAP in more detail in [7].) One key fact is that, to introduce CAP, only the card issuing bank had to change anything; no action was required of acquiring banks or of network switches. This causes us to wonder, first, what innovations might be possible to EMV that involve software changes only at a single acquirer. This is typically the bank with which the merchant deposits its card transactions, but acquirers need not be banks; there are established non-bank players like PayPal, and also small startups which provide merchant acquirer services. Second, once we've got to grips with the crypto protocols used in EMV, we'll ask what innovations might be possible with EMV that don't require any banks to change their systems at all.

## 2 Micro-merchant Transactions

Two pervasive problems with payment systems are market concentration and the cost of becoming a merchant. Many countries have only a handful of large banks, and payment services are not very competitive. They suffer from network effects; the current big card brands, of Diners, AmEx, Visa and MasterCard, are what's left of hundreds of payment card schemes, most of which never achieved the necessary critical mass [8]. In the UK, for example, two banks acquire over two-thirds of all credit card transactions from shops; these are the leading banks in the Visa and MasterCard stables respectively. This concentration has harmed innovation. At the beginning of the dotcom boom, small startups who needed credit card acquisition facilities often had to put down large deposits and put up with large merchant discounts – in the range of 5–10% of sales. Non-bank competition, in the form of services such as PayPal and Google Checkout, eventually fixed this problem; these channels accommodate even “micro-merchants” – such as occasional traders who sell goods on eBay.

There are two logical next steps. The first – given that cheques are disappearing throughout Europe, with their abolition in the UK planned for 2017 – is peer-to-peer payments, that is, payments between private individuals. At present, people will write a cheque to a friend or relative to pay for some shopping, or to repay an ad-hoc loan; so what happens after cheques vanish? It would be convenient to be able to wave your bank card over your laptop, or hold it to your mobile phone, in order to make or receive a payment from anyone.

The second (related) step is more competition for cardholder-present transactions. Traditional payment terminals are pricey, and are tied to expensive business banking facilities. So if you sell a handful of goods face-to-face – say at car boot sales, or a church raffle, or perhaps surplus produce from your garden in the summer – cash has been the only real option. Doing a PayPal transaction from my phone to yours is too fiddly!

So an exciting development is the emergence in the USA of services like [squareup.com](http://squareup.com) which will supply you with a tiny credit-card reader to plug into your phone, plus a merchant account to bank your takings. This service enables anyone to become a ‘proper’ merchant and accept credit cards, quickly and cheaply. The service is sufficiently threatening that Verifone is starting to sell similar devices, and the banks are changing standards to require encryption on the link between the reader and the phone – not to block any realistic attack, but to raise the entry costs for other upstart competitors.

So far so good, for micro-merchants in the USA. But what about Canada and Europe? Can we invent a similar service for EMV cards? EMV PIN Entry Devices (PEDs) are a closed market at present. They are supposed to be tamper-resistant, in that it should be hard to bug a merchant terminal to record card and PIN data. But the tamper resistance offered by industry certification schemes has turned out to be illusory [5]; it is be easy to bug a terminal, and many of these devices are compromised every year. Yet even if the certification requirements don't keep out the bad guys, they seem to keep out competition. The PED

market is consolidating quickly, and with the recent announcement of a takeover of Hypercom by Verifone, it looks like we'll be reduced to two major players (the other being Ingenico).

EMVCo is working on contactless payments, where we can either use a credit card with a terminal as before, but with the wire replaced by a wireless channel (typically NFC); or use our mobile phone instead of the card. This won't help the small merchants much if it remains a closed system with certification rules.

In the face of market concentration, poor security, and the lack in the EMV world of a service like [squareup.com](http://squareup.com) to compete at the low end, we need a way to adapt EMV to support cardholder-present low-cost merchant accounts. What are the options?

### 3 Adapting EMV

We are exploring the technical options. Our starting point was that any changes to EMV should require change by either the issuer or the acquirer, but not both. Changing one bank's systems is hard enough; changing 10,000 banks' systems is too much.

#### 3.1 Typical EMV Transaction Flow

First we will describe a typical EMV transaction (this is a much shortened version of the description in [6] to which the reader can refer for the full gory details). As Figure 1 shows, the protocol can be split into three phases: card authentication, cardholder verification and transaction authorization.

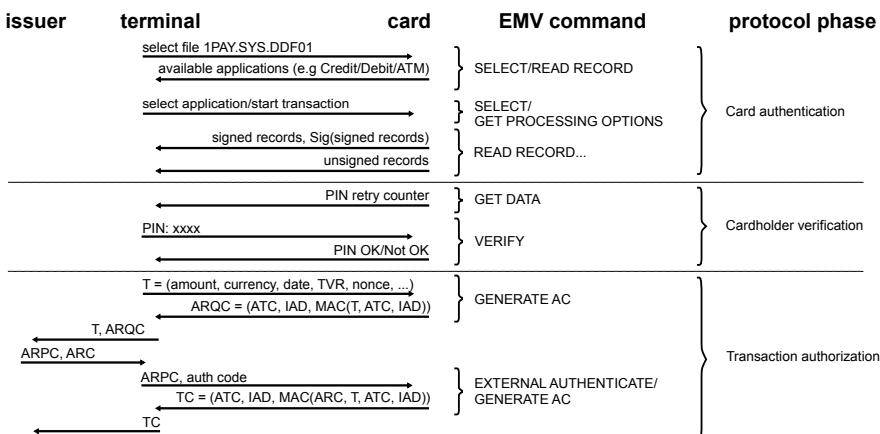


Fig. 1. A complete run of an EMV protocol instance

**Card Authentication.** When a card is inserted into a terminal, it requests a list of supported applications and starts a transaction by sending the `Get Processing Options` command to the card. It next sends `Read Record` commands to read cardholder information including card details (such as primary account number and expiry date), backwards compatibility data (the rest of the magnetic strip), and control parameters (including a list of acceptable cardholder verification methods). There is also an RSA digital signature over a subset of the records, together with a certificate chain to a card scheme root key embedded in the terminal.

In the low-cost variant of EMV, static data authentication (SDA), the card does only symmetric crypto and can only authenticate itself cryptographically to the issuing bank, which must be online for this to work. In the dynamic data authentication (DDA) variant, cards have RSA private keys which are used to sign a 32-bit nonce sent by the terminal.

**Cardholder Verification.** The card has a cardholder verification method (CVM) list stipulating when to use a PIN, or a signature, or nothing at all, to authenticate the cardholder, depending on the value of the transaction, its type (e.g. cash, purchase), and the terminal's capabilities. Assuming that the card wishes to verify a PIN, the customer enters it at the terminal which sends it to the card. If it's right, the card returns `0x9000` (this is not cryptographically authenticated by the terminal, but later by the bank from the transaction data).

**Transaction Authorization.** In the third step, the terminal asks the card to authenticate the transaction details. The terminal issues the `Generate AC` command to request an ARQC (authorization request cryptogram) from the card. The payload of this command is a description of the transaction, created by concatenating data elements specified by the card in the CDOL 1 (card data object list 1). Typically this includes details like the transaction amount, currency, type, a nonce generated by the terminal, the TVR (terminal verification results) and a sequence number (ATC – application transaction counter). Finally, the card returns the application cryptogram – a message authentication code (MAC), which is calculated over the transaction data with a symmetric key shared between the card and the issuing bank.

The ARQC is then sent by the terminal to the issuing bank, which performs various cryptographic, anti-fraud and financial checks; if it decides to authorise the transaction, it returns a two byte authorization response code (ARC), indicating how the transaction should proceed, and the authorization response cryptogram (ARPC), which is typically a MAC over  $\text{ARQC} \oplus \text{ARC}$ . These are forwarded by the terminal to the card, which validates them. The terminal asks the card for a transaction certificate (TC) cryptogram, which it sends to the issuer, and also stores in case of dispute. At this point it will typically print a receipt. There are quite a few variants of this transaction flow, such as where the card decides to accept a transaction offline and generates a TC without first seeing an ARPC; and this complexity has led to some known protocol vulnerabilities [6].

However, although these vulnerabilities mean that the EMV protocol does not entirely keep out the bad guys, so far it has managed to keep out competitors. The system is a closed one, and devices have to be certified tamper-resistant; even although the actual level of tamper-resistance of PIN entry devices is very low [5], the certification system is a closed one, new markets entrants have to sign up to the EMV agreements in order to participate. In this paper, we are interested in how a disruptive competitor might leverage the EMV issued card base and/or infrastructure in order to provide new payment mechanisms without having to get agreement from all the current EMV principals.

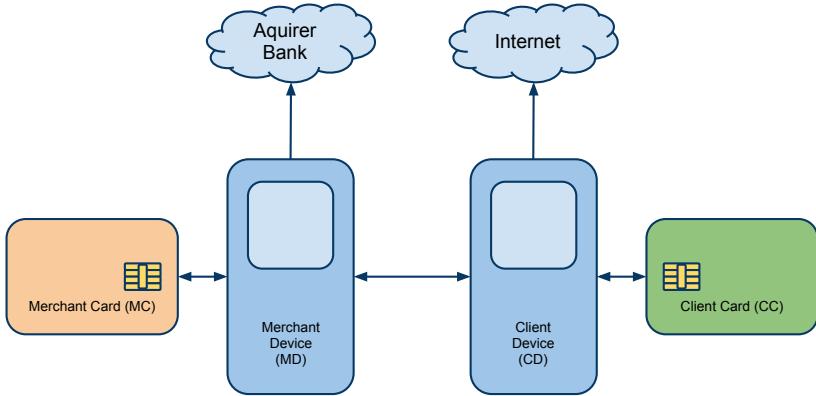
### 3.2 Breaking Tamper-Resistance in Court

One option might be for a new service provider to go to court to have the tamper-resistance certification standards set aside as a restrictive practice and thus break the Verifone/Ingenico duopoly. The service provider would then supply micromerchants with software for their phone or laptop that implements an EMV terminal (plus, until wireless communication becomes widespread, a cheap smartcard reader). The banks' lawyers would argue that in the absence of trustworthy hardware, malware could harvest card and PIN data; rogue software might also implement the no-PIN attack, so that stolen EMV cards could be used without knowledge of the PIN [6]. The market entrant's lawyers would argue that fraud would be managed in the same way that PayPal or Google Checkout do, by means of fraud-detection systems at the back end. But could we do any better?

### 3.3 Peer-to-Peer EMV – SDA

Our second possibility is that both customer and merchant have cheap smart card readers, rather than just the merchant. The idea is that the customer will use his own card in his own reader attached to his own phone, and the merchant likewise will use all her own equipment. This largely solves the problems of trusted path and trusted display; malware on the merchant's side can no longer compromise the customer's PIN or display a false payment amount to him. This would be a real improvement on existing systems, whether mag-stripe or EMV: at present, a merchant can program her terminal to display '\$5.00' yet send a transaction for '\$5,000.00' to the card (see Figure 4). There remains the risk of malware on the customer's equipment, which we'll discuss later.

The first variant on this theme is as follows (see Figure 2). Instead of issuing the merchant with a traditional EMV PED, the innovating bank or other acquirer promoting this new system issues her with software for her laptop or mobile phone that assembles the transaction data and submits it to a CAP transaction on her card. The eight-digit CAP code is used as the 32-bit 'unpredictable number' input into an EMV transaction that she sends to the customer. He presents it to his smart card and obtains the ARQC to send to the merchant.



**Fig. 2.** Framework for EMV peer-to-peer

She relays it in turn to her bank in a quite standard EMV transaction. Her bank verifies the CAP code to check that she is who she claims to be, and if so the main EMV transaction is fed to the switch for onward transmission to the customer's card issuer. From the switch onwards, there is no need to change anything as the EMV transaction flow is unchanged. Table 1 shows the new protocol flow.

The customer's card data can still be hijacked by malware on his own equipment. But if this is the worst that can happen, we have still managed to align incentives rather better than at present: everyone protects their own stuff. There is another minor technical attack in that a crooked merchant might send false transaction data to the customer, who cannot verify the CAP code. This can be fixed in various ways including having the customer send the transaction independently to the acquirer, or notifying the customer by SMS of all transactions. But there are other alternatives.

### 3.4 Peer-to-Peer EMV – Mixed-Mode

EMV cards come in three basic flavours: SDA, DDA and CDA, which are progressively more expensive. An acquiring bank or payment service provider that wants to offer low-cost merchant services can issue its own merchants at least with DDA or CDA cards, regardless of whether local cardholders have them or not. (Most countries use SDA or DDA.)

The merchant (see Figure 2) can now sign the transaction data and provide the first 32 bits of the signature to the customer as the unpredictable number. Things proceed as before, with a standard EMV transaction from the customer via the merchant and the acquiring bank to the customer's card issuer. It does not matter whether this transaction is SDA or DDA, and without loss of generality we can assume the former.

**Table 1.** Protocol flow for a P2P EMV transaction using SDA (see Figure 2 for the entities involved)

<b>dir</b>	<b>data</b>	<b>comment</b>
MD ↔ CD		MD and CD establish a connection via NFC, Bluetooth or other.
CD → MD	CDOL1, CDOL2	Client sends the transaction requirements.
MD → MC	$T = \{\text{Data as per CDOL, with UN=0}\}$	Merchant assembles transaction data as stated by the client's CDOL using Unpredictable Number 0.
MC → MD	$CK = \text{CAP}(T)$	Merchant's card produces 8-digit CAP code.
MD → CD	$T1 = \{T \text{ with UN replaced by CK}\}$	CAP code used as UN for transaction data sent to client.
CD → MD	$AC = \text{MAC}_k(Data_{ACC}, T1)$	Client sends Application Cryptogram (AC) to merchant, which in turn sends this to the acquirer bank.

There is one interesting technical twist. In the DDA protocol, the card signs some data including a 32-bit random nonce provided with the `Internal Authenticate` command, in a protocol designed to let the merchant know that the customer is present. In theory DDA cards can accept longer bit strings depending on the DDOL (Dynamic Data Object List). Tests have indicated that some UK-issued DDA cards accept inputs as large as 1024 bits (128 bytes) regardless of their DDOL. Such cards perform DDA even when we select an application for which DDA is not supported – as stated in the Application Interchange Profile (AIP). However other cards reject any DDA input not matching the DDOL specification. The former behaviour might be caused by incorrect DDA configuration or by design, so we might encounter both types of cards.

Anyway, there's a crypto design problem in respect of those DDA cards that will not sign long strings, as to how to use a device that signs a 32-bit payload to sign a whole transaction. It's no good to just sign the first 32 bits of the hash of the transaction data, as a collision attack can be found trivially. Signing each 32-bit block of a 160-bit SHA-1 hash would be sufficient but involves five transactions. Based on the transaction latency (see table 2) we observe that signing five hash components would take between 2 and 3 seconds, which is less than online authorisations take now.<sup>1</sup> So that's feasible, but we may wish to consider alternatives: the acquirer could issue CDA cards to its cardholders, so

<sup>1</sup> We have managed to obtain 5 consecutive DDA signatures in 2 seconds by omitting the earlier `read data` step in the transaction. Although the specifications say that DDA should be performed after this step, all DDA cards we have tested perform DDA even without reading data first.

**Table 2.** Performance data for different DDA transactions using two cards, one from Visa issued 2009 and another from MasterCard issued 2010. Time for consecutive signature generations includes reset time between transactions. The time measurements have been obtained using an oscilloscope connected to the Smart Card Detective [4], with a 4 MHz clock being provided, and 4 bytes being signed.

<b>Type of transaction</b>	<b>Transaction Time (ms)</b>		
	<b>without reading application data</b>	<b>reading application data</b>	
1 DDA signature Visa	426 (82 used for DDA generation)	1260	
1 DDA signature MasterCard	714 (178 used for DDA generation)	1776	
5 DDA signatures Visa	2190	6328	
5 DDA signatures MasterCard	3610	8936	

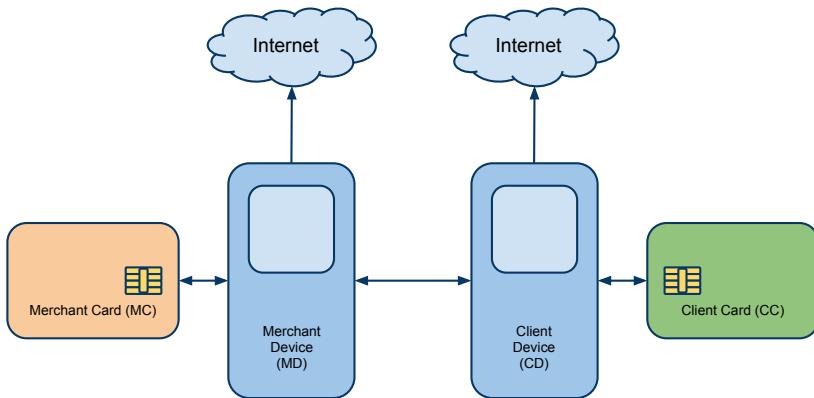
that when acting as merchants they could create full signatures; or the customer and merchant can commit to random numbers and thus jointly select a single block to be signed.

Another potential problem in using the DDA approach was the low availability of the system's public keys to the customer. DDA was designed for the terminal to verify a string signed by the card, whose key is certified using a closed PKI available to certified terminals only. This is perhaps less of an issue than one might have thought; we've been able to find the certificates for Visa cards online. It is also possible to extract root keys from a terminal, as they aren't as tamper-proof as they're advertised; another solution would be for individual cardholders and merchants to export their public key certificates to a third party. For example, if you use your RBS MasterCard to top up a PayPal account, you could export the MasterCard public-key certificate to PayPal directly. This opens up other possibilities too.

### 3.5 Going Outside the Banking System

In both the above cases, the banks generally need to consent to the new acquirer's P2P payment mode, and transfer money between customer accounts when requested. But we must bear in mind the possibility that in order to prevent disruption to their existing revenue streams, some banks will actively block any new modes of operation. Would the new acquirer have to go to court once more, and perhaps in many countries?

A radical alternative is to move the transfer of money out of the hands of the issuers and acquirers, and into the control of a non-bank payment system provider such as Google or PayPal. Customers would associate their cards' DDA public keys with their payment account, by uploading their public-key certificate and then signing a challenge to prove possession of the card.



**Fig. 3.** Framework for peer-to-peer transactions using EMV for authentication

When the customers wish to pay for goods, they present their card as normal, but only the initial stage of the EMV transaction occurs, where the card signs one or more terminal-provided nonces derived from transaction data (see above), thus proving to the merchant that the genuine card is present. This digital signature could be sent to the payment service provider, and money transferred. In the case of a customer with a PayPal account that automatically replenishes itself from a conventional credit card account, it might be logical to use that card to authorise PayPal transactions.

Another variant on this theme, which might be slower to take off but which should be less open to legal challenge by the banking industry incumbents, would be for the payment service provider to issue its own cards that would ‘embrace and extend’ – being EMV cards for normal transactions, CDA cards for merchant transactions in the new extended protocol, and also programmed to use their DDA keys to sign transactions in the new system.

In passing, we note that there is an interesting research problem to be tackled here for the DDA cards that only sign 32-bit strings. If the customer and merchant have mobile phones that are capable of public key cryptography, can they get a shared Diffie-Hellman key authenticated using this mechanism? Given that EMV was designed during the ‘Crypto Wars’ – the long tussles during the 1990s over the exportability of cryptographic devices – it may be that DDA signatures were limited to 32 bits specifically to make it harder for people to adapt them for protecting confidentiality. If this was the case, its success is at best marginal. The customer and merchant phones can exchange  $g^x$  and  $g^y$  in the usual way deriving the key  $g^{xy}$  and then authenticate  $h(g^x, g^y, g^{xy})$  by using its first and second 32-bit words as  $N_M$  and  $N_C$  in a signature exchange as set out in table 3. It may be just about possible for a middleperson to set up  $2^{16}$  transactions with different counterparties, and find colliding nonces; but this scarcely gives an industrialisable attack on a real payment system. What’s more, if a couple of

crooks were set on using their bank cards to obtain strong authentication for a home-brew crypto product, they could simply use multiple signatures. We leave as an exercise to the reader the question of whether robust authentication can be obtained without multiple signatures.

**Table 3.** Protocol flow for a P2P transaction using EMV for authentication (see Figure 3 for the entities involved)

dir	data	comment
MD ↔ CD	$g^x, g^y, g^{xy}$	MD and CD establish a connection via Diffie-Hellman
MD → CD	$M_{ID}, N_M$	Merchant sends ID and a nonce $N_M$ derived from the DH data
CD → MD	$C_{ID}, N_C, \text{Signed}_{DDA}(data_{CC}, N_M)$	Customer sends ID, a DH-derived nonce and DDA signature on $N_M$
MD → CD	$\text{Signed}_{DDA}(data_{MC}, N_C)$	Merchant sends DDA signature
CD → MD	non-EMV payment	Customer verifies merchant's signature and proceeds with PayPal or other non-EMV payment.

### 3.6 Merchant Authentication

An acquirer or non-bank payment service operator who was implementing a peer-to-peer payment system might want to think long and hard about merchant authentication. If nothing much authenticates the merchant name to the customer, then a micro-merchant might pretend to be a big-name retailer, in a new variant of the phishing attack. So a prudent acquirer who issued CDA cards to merchants might want to include merchant names in card certificates.

The issue that follows on at once from this is naming. An acquirer or payment service operator who certifies the names of payment recipients had better screen them for names similar to established brands, lest it be implicated in a fraud and sued by the brand owner.

Such due diligence is necessary but not sufficient. It is a hard problem to give each of millions of merchants a human-recognisable name that's unique in the universe! Even if well-known corporate brands get some protection from a name screening process, there will still be occasional confusion between small merchants' names, as well as the many disputes where a customer may or may not have bought something from merchant X but gets billed by merchant Y. (The first author recently had to sue his bank in the small claims court to get a refund in such a dispute [2].) So more work may be needed on usability: for example, there may be a case for receipts to contain a globally unique merchant number that can be used to identify the merchant unambiguously at the payment service operator's website.

## 4 Using a Bank Card as a General-Purpose Key

We remarked in the introduction that we would finally look to see if there were any innovative uses for EMV that did not require any bank – issuer or acquirer – to change its systems. Indeed there is one: using an EMV card as a key.

Although a signature on a 32-bit payload may be too small for general-purpose use over a hostile network, it is perfectly acceptable as a means of unlocking or otherwise controlling a device over a trustworthy path. Such a path also deals with the problem of the no-pin attack, in which a middleperson can use a card without knowing the PIN. We will discuss two examples – mobile phones and airline tickets.

If the phone of the future is to have applications that permit one-click payment – whether these are web-hosted apps such as Amazon’s bookstore, or NFC payments that don’t require a PIN – then the security-conscious phone user might use her DDA card as a means of unlocking the phone application that does these. It can also be used to ensure that the right account is selected: she would touch her phone with her NFC Visa card to make a tick payment from the Visa account, and with her NFC debit card to make the payment from her checking account instead. For that matter, a touch with a credit card might be used as a means of unlocking the phone itself, or any other programmable device that can communicate with it.

There is some history behind the idea of using credit cards as keys. A generation ago, hotels tried to use credit cards as room keys, and were blocked when banks objected; undeterred, they started using room keys with the same form factor but proprietary encodings. (And just as mag stripe card vendors developed a secondary market in door locks, so could the smartcard vendors, whose costs of developing DDA cards have been fully amortised by now.)

More recently airlines and rail operators have started asking customers to retrieve pre-booked tickets using the credit card with which they paid for them – a practice to which the banks do not nowadays object. The DDA signature mechanism gives a perfectly good way to implement this; it gives stronger assurance than simply presenting an easily copied static certificate. In this case there is probably not a practical issue with middleperson or collision attacks, as the customer has already done an online transaction that verifies the card number, the billing address and the availability of funds.

There is thus an entirely legitimate use of EMV cards in authenticating air travellers and ensuring that people cannot get on airplanes using forged credit cards. Given the extreme risk-aversion of the aviation security industry, perhaps DDA signatures will be mandated. Should that come to pass, our results show that the implementers will not have to pay rent to the Verifone/Ingenico duopoly. The signature mechanism can be implemented by adding low-cost smartcard readers to airport check-in machines. And given that DDA signature generation is independent of PIN entry, the use of PINs could be omitted or mandated according to the airport’s policy.

## 5 Trustworthy Hardware

The argument may still be made that if a large number of people start to use a peer-to-peer payment mechanism with essentially untrustworthy terminals – commodity smartcard readers connected to mobile phones – then eventually we can expect malware that hijacks customer phones or PCs, stealing card and PIN data. If hotel chains and other businesses start using customers' bank cards as general-purpose keys, then attacks using dedicated malicious hardware become conceivable too. Of course, research on CAP and on PEDs has shown that no hardware is truly trustworthy; shop terminals are routinely compromised, and a gang might distribute malicious CAPs. That said, dedicated payment terminals can definitely make the attacker's job harder.

An interesting possible line of development is to combine the concept of the PED (owned by the merchant) and the CAP (owned by the customer) to produce a device that both can use. A hardware device with both a trustworthy display and trustworthy input can simultaneously solve both the display problem and the malware problem. We initially proposed such a device as a solution to the trustworthy display problem in [3], and we have now built a prototype (the Smart Card Detective [4]) which can be seen in Figure 4. A pluggable serial port means that it can be connected to a conventional terminal using a wired smartcard, thus enabling a customer to defend herself against a corrupt merchant or a rogue PED; it could also be connected via USB to a low-cost merchant terminal as described in section 3.1 above; and it could also participate in a peer-to-peer transaction as described in 3.2 above by standing in for the customer's phone. We used this device to perform the tests reported in this paper.



**Fig. 4.** Device with trusted display which can demonstrate to the customer if the protocol is being executed correctly

While this particular design is simply a proof of concept, it illustrates that a low-cost trustworthy smartcard interface, with a display and a control button, is straightforward to design and manufacture.

## 6 Conclusions

Extensibility is the key problem for cryptographers designing real-world protocols and systems. On the one hand, systems such as EMV acquire considerable complexity under their own steam; the vulnerabilities reported in [6], for example, arose because the specification had become unmanageably complex.

Yet innovation will still happen. CAP was one example of how the EMV protocol was adapted to changing business demands. That was possible because the change affected only issuer systems. In this paper we have considered a different change, to adapt EMV to a lower-cost cardholder-present transaction model and ultimately to fully peer-to-peer payments between any two cardholders. This can be done, with the cooperation of some member banks, by changing only acquiring banks' systems. The lesson from this, we suggest, is to make protocols modular enough that they can be upgraded one side at a time.

We also showed that it was possible to use EMV cards as general-purpose signing oracles. Some cards will sign full-length data while others will only sign 32-bit strings; but there are some non-bank applications where such a signature is still very useful, such as authenticating an air passenger by verifying that he possesses the original credit card with which his ticket was purchased. For this reason it seems unlikely that the banking industry will be able to impose monopoly control forever on the uses to which bank cards are put.

As for uses that compete at least in part with the existing bank card system, we described how an acquirer could use the signing facility to support a different, non-bank, payment system. A customer might use her Visa or MasterCard to authenticate transactions on PayPal. The fact that some DDA cards don't do proper signatures is at most a small bump in the road, as there are various workarounds. The biggest limitation may be that DDA signature and PIN verification aren't linked at the protocol level, so in the absence of a trusted path from card to terminal, middleperson attacks may be possible. (But they are anyway on EMV; even if the PED is trustworthy, it doesn't give a trusted path.) Some banks might even fear that fixing this flaw would be against their best interests, as it would make their monopoly easier to break! But the potential interactions between security and competition are complex; at the very least, designers should bear in mind the chosen protocol attack [9].

In earlier work, we discussed the need for better governance within the card systems themselves. EMV has largely escaped from the control of EMVCo; its evolution is being pushed by dozens of vendors, and pulled by thousands of banks. The security mechanisms are not as good as they should be at keeping out attacks, yet they pose real obstacles to constructive innovation. This work showed that the interaction with non-banks, from airline boarding-pass machine vendors through innovative payment startups to the large payment service providers, could provide the next wave of disruptive innovation. The forces

at play there may pit competition against security; the oligopoly of the brands, banks and terminal vendors (which it's in the public interest to smash) against the chosen protocol attack, which (insofar as it harms cardholders) is in the public interest to avoid. Perhaps the Fed and the European Central Bank will have to step up to the plate and require that the EMV protocol be opened up and brought under proper governance (as suggested in [6]).

**Acknowledgement.** Omar Choudary is a recipient of the Google Europe Fellowship in Mobile Security, and this research is supported in part by this Google Fellowship. However the ideas in this paper were not discussed with any Google staff prior to submission and thus do not necessarily represent the views of Google.

## References

1. Anderson, R.: *Security Engineering – A Guide to Building Dependable Distributed Systems*. Wiley (2008)
2. Anderson, R.: How to get money back from a bank, on LightBlueTouchpaper.org (March 29, 2010), <http://www.lightbluetouchpaper.org/2010/03/29/how-to-get-money-back-from-a-bank/>
3. Anderson, R., Bond, M.: The Man-in-the-Middle Defence. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) *Security Protocols*. LNCS, vol. 5087, pp. 153–156. Springer, Heidelberg (2009)
4. The Smart Card Detective: a hand-held EMV interceptor, Omar Choudary, MPhil thesis at University of Cambridge, Computer Lab, <http://www.cl.cam.ac.uk/~osc22/scd/>
5. Drimer, S., Murdoch, S., Anderson, R.: Thinking inside the box: system-level failures of tamper proofing. In: IEEE Symposium on Security and Privacy, pp. 281–295; journal version as Failures of Tamper-Proofing in PIN Entry Devices. *IEEE Security and Privacy* 7(6), 39–45 (November-December 2009)
6. Murdoch, S., Drimer, S., Anderson, R., Bond, M.: Chip and PIN is Broken. In: IEEE Symposium on Security and Privacy, pp. 433–444 (2010)
7. 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)
8. Evans, D.S., Schmalensee, R.: Failure to Launch: Critical Mass in Platform Businesses. *Review of Network Economics* 9(4) (2010), <http://www.bepress.com/rne/vol9/iss4/1>
9. Kelsey, J., Schneier, B., Wagner, D.: Protocol Interactions and the Chosen Protocol Attack. In: Christianson, B., Lomas, M. (eds.) *Security Protocols 1997*. LNCS, vol. 1361, pp. 91–104. Springer, Heidelberg (1998)
10. Murdoch, S.J., Anderson, R.: Verified by Visa and Mastercard Securecode: Or, How Not to Design Authentication. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 336–342. Springer, Heidelberg (2010)

# **hPIN/hTAN: A Lightweight and Low-Cost E-Banking Solution against Untrusted Computers<sup>\*</sup>**

Shujun Li<sup>1</sup>, Ahmad-Reza Sadeghi<sup>2,3</sup>, Sören Heisrath<sup>3</sup>,  
Roland Schmitz<sup>4</sup>, and Junaid Jameel Ahmad<sup>1</sup>

<sup>1</sup> University of Konstanz, Germany

<sup>2</sup> Darmstadt University of Technology and Fraunhofer SIT, Darmstadt, Germany

<sup>3</sup> Ruhr-University of Bochum, Germany

<sup>4</sup> Stuttgart Media University, Germany

**Abstract.** In this paper, we propose hPIN/hTAN, a low-cost hardware token based PIN/TAN system for protecting e-banking systems against the strong threat model where the adversary has *full* control over the user's computer. This threat model covers various kinds of attacks related to untrusted terminal computers, such as keyloggers, screen scrapers, session hijackers, Trojan horses and transaction generators.

The core of hPIN/hTAN is a secure and easy user-computer-token interface. The security is guaranteed by the user-computer-token interface and two underlying security protocols for user/server/transaction authentication. The hPIN/hTAN system is designed as an open framework so that the underlying authentication protocols can be easily re-configured. To minimize the costs and maximize usability, we chose two security protocols dependent on simple cryptography (a cryptographic hash function).

In contrast to other hardware-based solutions, hPIN/hTAN depends on neither a second trusted channel nor a secure keypad nor external trusted center. Our prototype implementation does not involve cryptography beyond a cryptographic hash function. The minimalistic design can also help increase security because more complicated systems tend to have more security holes. As an important feature, hPIN/hTAN exploits human users' active involvement in the whole process to compensate security weaknesses caused by careless human behavior.

## **1 Introduction**

Nowadays e-banking becomes more and more popular all over the world. A 2010 survey of the American Bankers Association showed that e-banking is now the most preferred banking method of bank customers [2]. There is no doubt that most users consider security as the most important issue about e-banking. The earliest and simplest defense protecting e-banking systems is user authentication

---

\* Companion web page (with a full edition of this paper): <http://www.hooklee.com/default.asp?t=hPIN/hTAN>

based on static PINs. The end-to-end secure communications between the client and the e-banking server is typically achieved via the SSL/TLS protocol.

While SSL/TLS is considered secure, static PINs are prone to social engineering attacks, in which the users are spoofed to expose their PINs. One of the most prevailing social engineering attacks is phishing attack [16]. In its simplest form the attacker sends phishing emails to lure gullible users to disclose their PINs on a bogus e-banking web site. Once the attacker gets the PIN of a victim, he will be able to steal the victim's money by logging into the e-banking system. To provide higher security, two-factor authentication has been widely deployed by financial institutions for strengthening their e-banking systems. The most prominent two-factor authentication scheme used for e-banking is PIN/TAN, which uses static PINs for login and one-time TANs for online transactions.

While PIN/TAN can reduce the risk of simple social-engineering attacks like email based phishing, it does not offer any security against man-in-the-middle (MitM) attacks, in which the adversary controls the communication channel between the user and the e-banking server. In a typical MitM attack, the adversary establishes an SSL/TLS connection with the e-banking server and another one with the user, and then forwards the PIN and TANs from the user to the e-banking server as usual, but tampers with the transaction data in real time.

MitM attacks can be made stronger if the attacker partially/fully compromises the user's computer. This is possible due to the wide spread of malware over the Internet. Some malware can inject malicious code into the web browser, so that the attacker can do more than in MitM attacks: monitoring the user's input in the web browser, redirecting the user to a fake web site, modifying the contents of web pages shown in the web browser, and so forth. This kind of attacks are sometimes called man-in-the-browser (MitB) attacks [13]. Other malware such as Trojans or rootkits can even allow the attacker to take *full* control over the user's computer. In the worst case, all the software, hardware drivers, the operating system and even reprogrammable hardware are under the *full* control of the attacker, thus rendering the user's computer *totally* untrusted.

In this paper, we consider e-banking solutions against attacks related to *fully* untrusted computers, and call them "man-in-the-computer" (MitC) attacks. Depending on the contexts, MitC attacks have different names in the literature, e.g. malware-based attack or malicious software attack [32], Trojan attacks [24], content-manipulation attacks [20], transaction generator attack [15], and so on.

Since the main goal of MitC attacks is transactions manipulation rather than identity theft, it is clear that the corresponding solutions for secure e-banking aim at providing transaction authentication. Roughly speaking, there are two basic approaches to achieve transaction authentication: the first approach requires message authentication of the transaction data sent from the user to the server, and the second one requires secure transmission of the transaction data and a transaction-dependent TAN back to the user for re-confirmation of the requested transaction. Normally, the first approach involves a trusted input method of the transaction data and a trusted channel for secure data transmission from the user to the server, and the second one involves a trusted out-of-band (OOB)

or encrypted channel for data transmission from the server back to the user. The re-confirmation in the second approach is achieved by simply sending the transaction-dependent TAN back to the server without protection.

A typical solution in use is mTAN deployed by many financial institutions around the world [22, 4]. The mTAN system follows the second approach, and use the cellular network as the additional trusted OOB channel to send the transaction data and the transaction-dependent TANs back to the user via SMS. The user verifies the transaction data and then sends the TAN to the server to re-confirm the transaction. While mTAN is able to offer an acceptable level of security against MitC attacks, the OOB channel based on cellular network and a smart phone is not always available at the user side. Furthermore, the cellular network is not free from potential attacks [28]. In addition, the user’s smart phone may also be infected by malware [27] and is still prone to some more advanced attacks such as SIM card swap frauds [23] and insider attacks from the telecommunication service providers [14]. The high complexity of today’s smart phones may also lead to potential security holes induced by software bugs [21].

In addition to mTAN, there are many other e-banking solutions against MitC attacks. A lot of these solutions are based on hardware devices such as general-purpose personal computing devices (smart phones or PDAs), smart card readers or USB-tokens. Although many of them do work in practice, they all have non-trivial drawbacks, which include relatively high implementation/maintenance costs, dependence on an external network/server, low usability, doubtful security, and so forth. As far as we know, all hardware-based solutions depend on at least one of the following three components: second trusted channel, secure keypad, and encryption. Some solutions also require optical devices such as digital cameras or optical sensors.

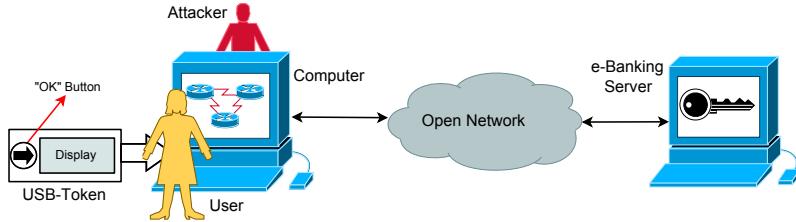
**Our contributions:** In this paper, we propose hPIN/hTAN, the first (to the best of our knowledge) hardware-based solution against MitC attacks that depends on none of the following components: second trusted channel, secure keyboard, trusted third-party, encryption. The main design goal of the hPIN/hTAN system is to achieve a better tradeoff between security and usability with a low-cost and easy-to-use USB-token. The security requirements are guaranteed through a secure user-computer-token interface and two security protocols. The interface can also reduce or compensate careless errors made by humans who may become the weakest link in the whole system.

**Paper organization:** In the next section, we introduce our hPIN/hTAN system in detail. The security analysis of hPIN/hTAN is given in Sec. 3. The usability and deployment issues are discussed in Sec. 4. In Sec. 5, we overview related work, their drawbacks and compare hPIN/hTAN with existing solutions. The last section concludes the paper and gives some planned work in the future.

## 2 The Proposed hPIN/hTAN System

Our hPIN/hTAN system is composed of two parts – hPIN and hTAN, which protect the login process and online transactions, respectively. The hPIN part

also protects the hTAN part from potential abuse by enabling it only after the user successfully passes the hTAN part. In the following, we discuss the model, notations, requirements and the two protocols involved, respectively.



**Fig. 1.** The threat model of the hPIN/hTAN system

**System Model:** As shown in Fig. 1, the involved parties in hPIN/hTAN are a human user  $U$ , a trusted USB-token  $T$  issued by the financial institute to the user, an untrusted terminal computer  $C$  (i.e., a MitC attacker), and the remote e-banking server  $S$ . In a typical scenario, the human user  $U$  plugs the USB-token  $T$  into a USB-port of the untrusted computer  $C$ , tries to access the remote e-banking server  $S$  and then makes some online transactions. We assume that the e-banking server  $S$  is trusted, which is a reasonable assumption in practice. The main threat we consider is the MitC attacker who is able to both observe and manipulate communications between  $U$  and  $C$ ,  $T$  and  $C$ ,  $S$  and  $C$ . Moreover, we assume the USB-token  $T$  is a trusted device to the user so that the MitC attacker has no access to any data stored inside  $T$ .

The notations used in this paper are summarized in the following table.

IDU	User ID.
$K_T$	Secret key shared between $T$ and $S$ .
PIN	$n$ -character PIN shared between $U$ and $T$ .
$\text{PIN}(i)$	The $i$ -th character of PIN.
$s$	Salt used to be hashed together with PIN.
STD	Sensitive transaction data that are authenticated.
NSTD	Non-sensitive transaction data that are not authenticated.
$h(\cdot)$	$m$ -bit cryptographic hash function.
$\text{HMAC}(K_T, \cdot)$	HMAC constructed based on $h(\cdot)$ .
$a \parallel b$	Concatenation of $a$ and $b$ .
$K_T^*$	$= K_T \oplus h(\text{PIN} \parallel s)$ (stored in $T$ ).
$\text{PIN}^*$	$= h(\text{PIN} \parallel K_T \parallel s)$ (stored in $T$ ).
$\mathcal{F}_i : \mathbb{X} \rightarrow \mathbb{Y}$	Random code mapping $\text{PIN}(i) \in \mathbb{X}$ to a printable character in $\mathbb{Y}$ .
$C_T, C_S$	Two counters stored in $T$ and $S$ .
$V_T, V_S$	Maximal numbers of consecutive failures allowed by $T$ and $S$ .

**Security Requirements:** Under the above system model, hPIN/hTAN is designed to achieve the following security requirements:

1. *PIN confidentiality*: the attacker cannot get the user's PIN in clear, except for a negligible probability;
2. *User authenticity*: the attacker cannot access the e-banking server without the presence of the legitimate user, except for a negligible probability;
3. *Server authenticity*: the attacker cannot cheat the user into connecting to a fake e-banking server, except for a negligible probability;
4. *Transaction integrity/authenticity*: the attacker cannot modify/forge a transaction without being detected, except for a negligible probability.

Note that the second and the third requirements are equal to mutual authentication between the user  $U$  and the server  $S$ .

**System Requirements:** The USB-token used in the hPIN/hTAN system is designed following a minimalistic principle. In addition to the basic components for building a USB device, it also includes a small display and an "OK" button. Two security protocols are embedded in the USB-token to implement user/server/transaction authentication. For our prototype system, we chose two security protocols based on an  $m$ -bit keyed hash function (HMAC). We avoid using any more cryptography to minimize the system complexity.

When a USB-token is manufactured, an  $m$ -bit secret key  $K_T$  and an initial PIN are assigned to it, where the PIN is composed of  $n$  characters in a finite set  $\mathbb{X}$ . The secret key  $K_T$  is crucial for the security of the hPIN/hTAN system, and is never shown in clear to the user and cannot be changed by the user. In contrast, the PIN is mainly used to protect the USB-token from theft and loss. As a whole, in the USB-token, the following data are stored:

$$\text{IDU}, s, K_T^* = K_T \oplus h(\text{PIN} \parallel s), \text{PIN}^* = \text{HMAC}(K_T, \text{PIN} \parallel s), C_T,$$

where  $C_T$  is used to signal locking the USB-token if more than  $V_T$  wrong PINs have been entered consecutively. The salt  $s$  is used to frustrate rainbow table attacks. Note that  $K_T$  is encrypted, and cannot be recovered without access to the correct PIN. The e-banking server stores the following data for the user:

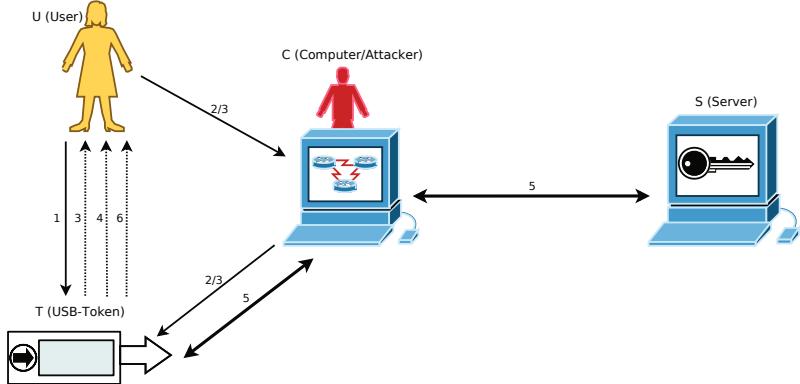
$$\text{IDU}, h(K_T), C_S,$$

where  $C_S$  is used to signal locking the user's account if more than  $V_S$  consecutive failures of user authentication have happened.

Based on the above system requirements, the following two subsections describe how the hPIN and hTAN parts work. Note that running both parts needs installation of a plugin to the web browser of the terminal computer, which is in charge of communications between the USB-token  $T$  and the computer  $C$ .

## 2.1 The hPIN Part

The hPIN part protects the login process via the following two components: authentication of the user to the USB-token, and mutual authentication between the USB-token and the e-banking server. The second component can be implemented by any mutual authentication protocol. In this paper, we choose



**Fig. 2.** The hPIN part, where solid lines denote real interactions/communications and dashed lines denote information display (the same hereinafter). The thick solid lines highlight the reconfigurable mutual authentication protocol.

the SKID3 protocol [9], a generalized edition of the unilateral authentication protocol defined in ISO/IEC 9798-4. More complicated mutual authentication protocols can certainly be used here, but the simple SKID3 protocol is sufficient to achieve the security requirements of hPIN/hTAN. Thanks to the simplicity of SKID3, the computational complexity of the hPIN/hTAN is very low. Figure 2 and the following description explain how the whole hPIN part works.

**Step 1:** U connects T to C, and presses the “OK” button on T.

**Step 2:** U enters IDU on the untrusted keyboard and sends it to T via C.

**Step 3:** For  $i = 1, \dots, n$ , T and U perform the following interactive protocol:

- T randomly generates a one-time code  $\mathcal{F}_i : \mathbb{X} \rightarrow \mathbb{Y}$ , shows all codewords  $\{\mathcal{F}_i(x) | x \in \mathbb{X}\}$  to U via its display;
- U enters  $\mathcal{F}_i(\text{PIN}(i))$  with the untrusted keyboard of C;
- T decodes  $\mathcal{F}_i(\text{PIN}(i))$  and performs  $i = i + 1$ .

**Step 4:** T verifies if  $\text{PIN}^* = h(\text{PIN} \parallel (K_T^* \oplus h(\text{PIN} \parallel s)) \parallel s)$ . If so, then T recovers the secret key as  $K_T = K_T^* \oplus h(\text{PIN} \parallel s)$ , stores  $h(K_T)$  in its volatile memory for future use in the hTAN part, shows a “PIN correct” message to the user U via its display, and goes to Step 5; otherwise T performs  $C_T = C_T + 1$ , shows an alert to U and stops. If  $C_T > V_T$ , T locks itself.

**Step 5:** T and S authenticate each other by following a mutual authentication protocol. When the SKID3 protocol is used, the mutual authentication process works as follows:

$T \rightarrow S: \text{UID}$ ,

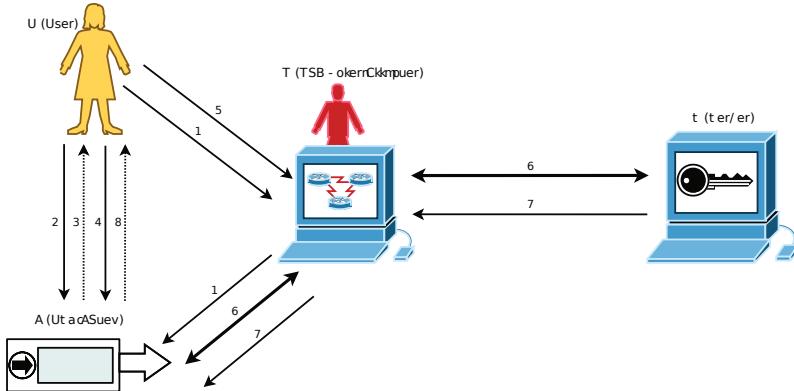
$S \rightarrow T: r_T$ ,

$T \rightarrow S: (\text{UID}, r_S, H_1 = \text{HMAC}(K_T, r_S \parallel r_T \parallel T))$ ,

$S \rightarrow T: H_2 = \text{HMAC}(K_T, r_T \parallel r_S \parallel S)$ ,

where  $r_S$  and  $r_T$  are nonces generated by S and T respectively.

**Step 6:** T shows a message on its display to inform U about the result of the mutual authentication protocol in Step 5.



**Fig. 3.** The hTAN protocol. The thick solid lines highlight the reconfigurable transaction/message authentication protocol.

After U successfully logs into the e-banking system with T, she can change the PIN if she wants. To do so, U asks C to signal T about the input of a new PIN. The new PIN can be entered in the same way as in Step 3 of the above hPIN process. After completing the PIN input, U presses the “OK” button on T twice and then T updates the values of  $K_T^*$  and PIN\*.

## 2.2 The hTAN Part

The hTAN part protects online transactions from MitC attacks after the user has successfully passed the hPIN process. As shown in the previous subsection, the hTAN part is enabled upon the completion of the hPIN process. The core of the hTAN part is a human-computer-token interactive protocol that allows *simultaneous* transaction input on the untrusted keyboard of C and transaction verification via the trusted display of T. This interactive protocol ensures that T receives the real transaction data U wants to make. After that, T runs a transaction authentication protocol to send the real transaction data to S. In our prototype of hPIN/hTAN, we use the same HMAC scheme involved in the hPIN part for construct the transaction authentication protocol, so that the whole system is based on a single hash function and a single HMAC scheme.

**Step 1:** U clicks a button on the e-banking web page to inform T about the start of a new online transaction attempt. Then, she inputs each STD item one after another on the untrusted keyboard of C by repeating Steps 1–4. To embed STD verification into the input process, each character in the STD is shown like passwords (e.g., as a number of asterisks) on the untrusted monitor of C, but in clear on the trusted display of T. This can naturally force U to verify the STD *simultaneously* while she is entering the STD. If U presses “Backspace” key, T shows an eye-catching warning message to inform the user for a few seconds and then the previously entered character is deleted. The goal of such a special design is explained later in Sec. 3.3.

**Step 2:** Upon completion of one STD item, U presses the “OK” button on T.  
**Step 3:** T highlights the current STD item for a few seconds, and prompts U to press the “OK” button again.

**Step 4:** U presses the “OK” button again to approve the current STD item.

**Step 5:** U inputs NSTD to T by filling a blank on the web page in clear.

**Step 6:** T sends STD and NSTD to S by running a transaction/message authentication protocol. Here, we use HMAC to build the following protocol:

$T \rightarrow S: (\text{IDU}, \text{STD}, \text{NSTD}, r_T^*)$ ,

$S \rightarrow T: (r_S^*, H_3 = \text{HMAC}(K_T, r_T^* \parallel r_S^* \parallel \text{STD}))$ ,

$T \rightarrow S: (\text{IDU}, H_4 = \text{HMAC}(K_T, r_S^* \parallel r_T^* \parallel \text{STD}))$ ,

where  $r_T^*$  and  $r_S^*$  are two new nonces generated by T and S, respectively.

**Step 7:** S checks if  $H_4 = \text{HMAC}(K_T, r_S^* \parallel r_T^* \parallel \text{STD})$ . If so, S executes the requested transaction and sets  $M = \text{"success"}$ , otherwise sets  $M = \text{"error"}$ .

Then, S sends  $H_5 = \text{HMAC}(K_T, r_T^* \parallel r_S^* \parallel M \parallel \text{STD})$  to T.

**Step 8:** T checks if  $H_5 = \text{HMAC}(K_T, r_T^* \parallel r_S^* \parallel \text{"success"} \parallel \text{STD})$ . If so, it shows “transaction executed”, otherwise “transaction failed”, on its display.

### 3 Security of hPIN/hTAN

In this section, we analyze the security of the hPIN/hTAN system, based on the assumption that  $h(\cdot)$  and the HMAC scheme are both cryptographic secure against attackers whose computational power is limited by  $2^{m/2}$ .

#### 3.1 PIN Confidentiality

The PIN protects the USB-token from theft and loss. Leaking the PIN to an attacker actually does not compromise the security of hPIN/hTAN (as long as the USB-token is not available to the attacker), but it may compromise the user’s privacy, since the PIN often relates to the user’s personal information such as birthday. In addition, many users share the same PIN/password (or part of it) over multiple e-banking systems, so leaking the PIN of one e-banking system protected by hPIN/hTAN may lead to compromise of other e-banking systems.

The PIN confidentiality is achieved by the use of the  $n$  random codes  $\mathcal{F}_1, \dots, \mathcal{F}_n$  in the hPIN process. In Step 2, the USB-token T does not send the  $n$  codewords to the untrusted computer C, but shows them on its own display. Since the USB-token is a trusted device, the attacker has no access to any of the  $n$  codes and thus is unable to decode the PIN from the user’s input  $\mathcal{F}_1(\text{PIN}(1)), \dots, \mathcal{F}_n(\text{PIN}(n))$ . Each PIN character is mapped to a printable character by a *different* code, the attacker cannot figure out repeatedly used characters in the PIN, either. Instead, the attacker can only exhaustively search all the possible PINs. This corresponds to a success probability  $|\mathbb{X}|^{-n} \ll 1$ , when  $|\mathbb{X}|^n \gg 1$ . Note that an offline attack on the PIN is not possible because no information about the PIN is transmitted to C. An online attack is also impossible because Step 1 requires physical access to T. The above facts imply that the attacker has no any clue to judge if a random guess is correct or not, thus making the brute-force attack useless.

### 3.2 User/Server Authenticity

The mutual authentication between  $U$  (actually  $T$ ) and  $S$  is guaranteed by the underlying security protocol in the hPIN part. For the SKID3 protocol, mutual authentication is guaranteed because the attacker can only passively forward communications between  $U$  (i.e.,  $T$ ) and  $S$ . That is, without the presence of  $U$  and  $T$ , the attacker cannot authenticate itself to  $S$ ; without the presence of  $S$ , the attacker cannot authenticate itself to  $T$ . Note that we do not attempt to prevent the attacker from reading communications between  $U$  (i.e.,  $T$ ) and  $S$ , since they have been exposed to the attacker by the untrusted computer  $C$ .

### 3.3 Transaction Authenticity/Integrity

Transaction authenticity/integrity is achieved by the hTAN part. There are two stages of transaction authentication: 1) the human-token-computer interactive protocol in Steps 1–4 guarantees the integrity of STD from  $U$  to  $T$ ; 2) the transaction/message authentication protocol in Step 6 guarantees the integrity of STD from  $T$  to  $S$ . Note that Step 8 is for the integrity of the “success” message from  $S$ , so it is independent of the integrity of STD and will not be discussed further.

The human-token-computer interactive protocol (Steps 1–4) ensures that  $T$  gets the STD without being manipulated. Since the user has to look at  $T$ ’s display to verify her input and then press the “OK” button twice to show confirmation,  $T$  will always receive the real STD that the user intends to input. Thanks to the use of the trusted display of  $T$ , the user can fully focus on the correctness of STD in the data input process. This is how *simultaneous* STD input and verification is achieved. The main goal of the special design on “Backspace” key is to prolong the time of deleting previously entered characters so that malicious deletion of STD characters by  $C$  can be easily noticed by  $U$ .

Although Steps 2–4 look like “verify after input”, the real purpose is to resist a competition attack: after  $U$  finishes typing the STD, the attacker sends one or more new digits to  $T$  and append them to  $U$ ’s STD. If this happens just before  $U$ ’s finger touches the “OK” button,  $U$  may wrongly confirm the attacker’s STD. By asking  $U$  to press the “OK” button in Step 2 and then press it again after a short delay, the above competition attack will become nearly impossible. To detect an ongoing competition attack,  $U$  does not need to re-verify the whole STD explicitly, but just pay attention to possible abrupt change of the STD. This is a very easy task since  $U$  keeps watching  $T$ ’s display during Steps 1–4.

One may wonder why “*simultaneous* input and verify” is better than the traditional “verify after input” process. There are three main reasons: 1) recent research [1] has shown that human users are not dependable in distinguishing manipulated e-banking transactions (especially when only a few characters are manipulated) under the “verify after input” setting of mTAN; 2) we believe that asking the user to input and verify STD simultaneously can reduce the total time of STD input and verification (see Sec. 4 for more detail); 3) we believe that the user’s active involvement at the very beginning of the hTAN process can help to enhance the user’s feeling and awareness of e-banking security.

After  $T$  gets STD from the user, it sends STD to  $S$  for execution. The transaction authentication and re-confirmation is ensured by the two STD-dependent HMAC values  $H_3$  and  $H_4$ . Under the assumption that the HMAC scheme is cryptographically secure, neither  $H_3$  nor  $H_4$  can be manipulated by the attacker with a non-negligible probability. Note that we also make the HMAC values depend on two new nonces to render replay attacks negligible.

## 4 Usability of hPIN/hTAN

We have developed a prototype implementation of hPIN/hTAN to test its usability. Three prototype USB-tokens have been produced and the hPIN/hTAN system has been implemented as firmware inside the USB-token and hostware running on a PC with Linux OS installed. Thanks to the simplistic design, the system is extremely lightweight: the size of the firmware is only around 10KB and the data memory requirement is less than 2KB. The actual costs of all components are about around 3–5 € per token. A virtual e-banking web site <http://www.hPIN-hTAN.net> was setup to simulate a genuine e-banking server. Figure 4 shows one prototype USB-token running Step 3 of the hPIN stage.



**Fig. 4.** One prototype USB-token running the hPIN user authentication step

A small-scale user study with 20 students and staff members of our universities shown that with a 4-digit PIN the median login time is 27.5 seconds and a transaction with 55 characters can be completed in around 70 seconds (1.27 seconds per character). The overall success rate of logins is  $60/66 \approx 91\%$ . We expect the login time may reduce significantly after the user becomes more familiar with both the system and the PIN. A survey of the participants showed that none of them had major difficulties understanding how the system works. Most users rated the overall usability of the hPIN/hTAN system as “very usable” and the mean opinion score is 3.65 on a 5-point scale. In the following, we give a qualitative analysis of the usability of hPIN/hTAN.

For the hPIN part, it is clear that the user interacts with  $T$  and  $C$  only in the first three steps and the following steps are done by  $T$  automatically. In Steps 1 and 2, the user only needs to press the “OK” button once and then input her ID, which does not add any additional usability problem compared with the

traditional PIN scheme. The user interface in Step 3 is a bit more complicated due to the use of the random codes. To enhance usability, we propose to show the codewords of each random code  $\mathcal{F}_i$  on T's display as follows (see also Fig. 4):

$$\begin{array}{ccccccccc} 0 & 1 & \cdots & 8 & 9 & \cdots \\ \mathcal{F}_i(0) & \mathcal{F}_i(1) & \cdots & \mathcal{F}_i(8) & \mathcal{F}_i(9) & \cdots \end{array}$$

The first row lists all possible PIN characters and the second shows the corresponding code of each PIN character. This allows the user to simply look for her PIN character  $\text{PIN}(i)$  and then input the character below it. With a list of codewords as shown above, an average user can input each  $\mathcal{F}_i(\text{PIN}(i))$  within a few seconds. This means the whole PIN can be input within  $O(n)$  seconds.

For the hTAN part, user interaction occurs only in Steps 1–5. Step 5 is about NSTD input, which is the same as the traditional TAN scheme, so we do not consider this step. The STD input in Step 1 is very similar to the normal text input in a web page. The only difference is that the user now should look at T's display rather than C's monitor to get visual feedback about what she is entering. By using a USB extension cable, the user can place T just above (or below) her keyboard so that she can easily see T's display. In this setup, the distance between the input device (C's keyboard) and T's display is much smaller than the distance between the input device and C's monitor, so the user is actually in a better condition of STD input. Steps 2–4 are very easy because the user either just waits and observes or simply presses a button on T. As a whole, we expect the additional time spent by an average user will be at the same order of traditional TAN schemes. Note that for TAN/PIN systems, the user has to look for the correct TAN on a paper list or wait for an SMS from the e-banking server, which can consume much more time than the hTAN process.

## 5 Related Work

As we mentioned in Sec. 1, transaction authentication is the key measure against MitC attacks, which can be achieved through two main approaches: 1) secure input and transmission of transaction data from U to S; 2) secure feedback from S to U for re-confirmation. In this section, we briefly overview previous solutions.

The first approach can be realized by transmitting the transaction data from U to S through an encrypted channel. For instance, in IBM ZTIC [32], a USB-token is used to establish a TLS channel for encrypting all communications between U and S. The USB-token has a trusted display and two buttons so that the user can explicitly confirm or cancel the transaction data. A low-tech solution called pTAN [7] is based on a paper list of secret permutation mappings, one of which is used for each transaction to conceal (encrypt) the input of transaction data from MitC attackers. Some other solutions are based on transaction-dependent TANs sent together with the transaction data to ensure transaction integrity. The TAN can be a MAC or a digital signature of the transaction data. A hardware device equipped with a secret key, such as a smart card reader [31, 30, 10] or a smart phone [25], is normally needed to calculate the TAN. To ensure that the TAN

is calculated from correct transaction data, either a trusted keypad is necessary or the trusted hardware device reads the transaction data from the computer screen optically.

The second approach requires a trusted channel for  $U$  to receive the feedback from  $S$ . Some solutions use an out-of-band (OOB) channel like the cellular network [22, 6]. Other solutions use an encrypted channel. Different kinds of hardware devices are used for decrypting data sent from  $S$ , including smart phones [18, 11] and special-purpose devices like personal AXS-tokens [3]. Some solutions [11, 3] also support direct readout of encrypted data from the computer screen. Visual cryptography and grille cipher are also used for this purpose [19, 8].

Among all existing solutions, the simplest ones are based on CAPTCHAs [29, 26], which use CAPTCHAs as a virtual channel that cannot be handled by automated MitC attackers. However, [17] shows that almost all e-banking CAPTCHAs are insecure against automated MitM attacks. In addition, human-assisted attacks may always be used to circumvent e-banking CAPTCHAs [5, 17].

Solutions based on low-tech “hardware” [26, 7, 19, 8] have a major advantage: the implementation costs are relatively low compared with solutions based on electronic devices. However, these solutions often require the user to perform mental computations and/or align a paper sheet with the computer screen, thus reducing usability. In addition, low-tech “devices” are often less portable than small electronic devices. This problem becomes even worse when the user has to bring more than one such low-tech “device” [26, 8]. Furthermore, when a user wants to make a large number of online transactions in a short period of time, low-tech “devices” can be quickly used up, leading to a denial of service.

To save implementation costs, many solutions use smart phones and PDAs as trusted devices [22, 6, 18, 11, 25]. The most severe problem with such general-purpose devices is the potential risks of being infected by mobile malware [27]. Even worse, in order to circumvent the language or functionality limits set by the manufacturers or the service providers, many users are willing to send their smart phones/PDAs to some private companies or alleged professionals to update the firmware, which makes it very easy for some attackers to inject malicious code into a large number of devices. In addition, as we point out for mTAN in Sec. 1, the high complexity of smart phones and PDAs leads to a higher risk of having security holes. If dependency on the cellular network is involved, then other weaknesses of mTAN will also be major problems.

In addition to mobile phones and PDAs, other trusted hardware devices used against MitC attacks include smart card readers [31, 30, 10], USB-tokens [32] and special-purpose devices like personal AXS-tokens [3]. All the smart card readers have a secure keypad as an essential component against MitC attacks. This not only increases the costs, but also reduces the device portability. In addition, some smart card readers are also improperly optimized to cause security flaws, and separation of the smart card and the reader leaves space for more potential risks [12]. The personal AXS-tokens do not have secure keypads, but are equipped with optical interfaces for reading data from the computer screen and biometric identification components, leading to a more expensive solution. Due to the need

of maintaining an encrypted channel, devices without a secure keypad have an encryption module, which also increases implementation costs.

In comparison with other existing solutions, hPIN/hTAN is designed to reduce implementation costs without compromising security and usability. It uses a USB-token as the trusted hardware device, so it does not suffer from the problems with mobile phones and PDAs. Instead of using a keypad for secure input, we propose human-involved interactive protocols to create a trusted path from the untrusted keyboard of the untrusted computer to the trusted device. We also intentionally make hPIN/hTAN independent of an additional trusted channel and strong encryption. Such a minimalistic design not only leads to lower costs and better usability, but also to less software bugs and security holes. Table 1 shows a comparison of hPIN/hTAN and selected hardware-based solutions.

**Table 1.** Comparison of hPIN/hTAN with selected hardware-based solutions

Solutions	Smart phone/PDA	Secure keypad	Encryption	Data channel	External party	Smart card
hPIN/hTAN	No	No	No	USB	No	No
mTAN [22, 4]	Yes	No	No	OOB	Yes	Yes
Sm@rtTAN plus [31]	No	Yes	No	No	No	Yes
Sm@rtTAN optic [30]	No	Yes	No	Optic	No	Yes
FINREAD [10]	No	Yes	Yes	USB	No	Yes
photoTAN [11]	Yes	Yes	Yes	Optic	No	No
QR-TAN [25]	Yes	Yes	Yes	Optic	No	No
IBM ZTIC [32]	No	No	Yes	USB	No	Optional
AXSionics [3]	No	No	Yes	Optic	Yes	No
MP-Auth [18]	Yes	Yes	Yes	Wireless	No	No

## 6 Conclusion

In this paper, we propose hPIN/hTAN, an enhanced PIN/TAN system based on a low-cost and easy-to-use USB-token, to protect e-banking systems from attacks related to untrusted computers, namely, man-in-the-computer (MitC) attacks. Our proposed system offers a better tradeoff between security and usability than existing solutions. The main feature of the system is the low complexity of the USB-token, which only needs to support a cryptographic hash function and some other simple functionalities. In addition, we carefully designed the protocols involved in the system to effectively exploit the human users' attention so that they will not be the weakest link in the system any more. Security analysis shows that hPIN/hTAN can achieve three security requirements: PIN confidentiality, user/server authenticity, and transaction authenticity/integrity.

We have developed a prototype system and performed a small-scale user study for demonstrating the usability of the hPIN/hTAN system. In the future we will

investigate more variants of the basic design, and try to figure out if some variants have even better overall performance than the basic hPIN/hTAN system reported in this paper. For instance, we will study if the USB channel can be replaced by an optic or wireless one to enhance usability but with acceptable additional costs. We also plan to run further user studies to show the real performance of hPIN/hTAN for average bank customers.

**Acknowledgments.** Shujun Li and Junaid Jameel Ahmad were supported by the Zukunftskolleg of the University of Konstanz, Germany, as part of the “Excellence Initiative” Program of the German Research Foundation (DFG). Shujun Li and Roland Schmitz thank Prof. Walter Kriha of the Stuttgart Media University for valuable discussions and comments on an early draft of the paper. The authors also thank all participants of our user study running at the University of Konstanz and the Stuttgart Media University.

## References

1. AlZomai, M., AlFayyadh, B., Jøsang, A., McCullagh, A.: An experimental investigation of the usability of transaction authorization in online bank security systems. In: Proc. AISC 2008. pp. 65–73 (2008)
2. American Bankers Association: ABA survey shows more consumers prefer online banking (2010),  
<http://www.aba.com/Press+Room/093010PreferredBankingMethod.htm>
3. AXSionics AG: Personal AXS-token (2009),  
<http://www.axsionics.ch/tce/frame/main/414.htm>
4. Bank Austria: mobileTAN information,  
<http://www.bank Austria.at/de/19741.html>
5. BBC News: PC stripper helps spam to spread (2007),  
<http://news.bbc.co.uk/2/hi/technology/7067962.stm>
6. Borchert, B.: Open sesame! – immediate access to online accounts via mobile camera phone, <http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/Open-Sesame/indexEN.php>
7. Borchert, B.: Knick-und-Klick-TAN, oder Permutations-TAN, pTAN (2009),  
<http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/pTAN>
8. Borchert, B., Beschke, S.: Cardano-TAN,  
<http://www2-fs.informatik.uni-tuebingen.de/studdipl/beschke>
9. Bosselaers, A., Preneel, B.: SKID. In: Bosselaers, A., Preneel, B. (eds.) RIPE 1992. LNCS, vol. 1007, pp. 169–178. Springer, Heidelberg (1995)
10. CEN (European Committee for Standardization): Financial transactional IC card reader (FINREAD). In: CEN Workshop Agreements (CWA) 14174 (2004)
11. Cronto Limited: Commerzbank and Cronto launch secure online banking with photoTAN – World’s first deployment of visual transaction signing mobile solution (2008), [http://www.cronto.com/download/Cronto\\_Commerzbank\\_photoTAN.pdf](http://www.cronto.com/download/Cronto_Commerzbank_photoTAN.pdf)
12. 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)
13. Gühring, P.: Concepts against man-in-the-browser attacks (2007),  
<http://www2.futureware.at/svn/sourcerer/CAcert/SecureClient.pdf>

14. IT-Online: World-first SMS banking scam exposes weaknesses (2009),  
<http://www.it-online.co.za/content/view/1092105/142/>
15. Jackson, C., Boneh, D., Mitchell, J.: Transaction generators: Root kits for web. In: Proc. HotSec 2007. pp. 1–4. USENIX (2007)
16. Jakobsson, M., Myers, S. (eds.): Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. John Wiley & Sons, Inc. (2007)
17. Li, S., Shah, S.A.H., Khan, M.A.U., Khayam, S.A., Sadeghi, A.R., Schmitz, R.: Breaking e-banking CAPTCHAs. In: Proc. ACSAC 2010. pp. 171–180 (2010)
18. Mannan, M., van Oorschot, P.: Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 88–103. Springer, Heidelberg (2007)
19. Naor, M., Pinkas, B.: Visual Authentication and Identification. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 322–336. Springer, Heidelberg (1997)
20. Oppiger, R., Rytz, R., Holderegger, T.: Internet banking: Client-side attacks and protection mechanisms. Computer 42(6), 27–33 (2009)
21. PC World: Nokia: We don't know why criminals want our old phones (2009),  
[http://www.pcworld.com/businesscenter/article/163515/nokia\\_we\\_dont\\_know\\_why\\_criminals\\_want\\_our\\_old\\_phones.html](http://www.pcworld.com/businesscenter/article/163515/nokia_we_dont_know_why_criminals_want_our_old_phones.html)
22. Postbank: mTAN now free for all customers (2008),  
[http://www.postbank.com/pbcom\\_ag\\_home/pbcom\\_pr\\_press/pbcom\\_pr\\_press\\_archives/pbcom\\_pr\\_press\\_archives\\_2008/pbcom\\_pr\\_pm1063\\_19\\_05\\_08.html](http://www.postbank.com/pbcom_ag_home/pbcom_pr_press/pbcom_pr_press_archives/pbcom_pr_press_archives_2008/pbcom_pr_pm1063_19_05_08.html)
23. Saturday Star: Victim's SIM swap fraud nightmare (2008),  
[http://www.iol.co.za/index.php?art\\_id=vn20080112083836189C511499](http://www.iol.co.za/index.php?art_id=vn20080112083836189C511499)
24. Schneier, B.: Two-factor authentication: Too little, too late. Comm. ACM 48(4), 136 (2005)
25. Starnberger, G., Froihofer, L., Goeschka, K.M.: QR-TAN: Secure mobile transaction authentication. In: Proc. ARES 2009, pp. 578–583. IEEE (2009)
26. Szydlowski, M., Kruegel, C., Kirda, E.: Secure input for web applications. In: Proc. ACSAC 2007. pp. 375–384. IEEE (2007)
27. The Financial Express: Russian phone virus that 'steals money' may spread global (2009), <http://www.financialexpress.com/news/russian-phone-virus-that-steals-money-may-spread-global/420770>
28. Toorani, M., Shirazi, A.A.B.: Solutions to the GSM security weaknesses. In: Proc. NGMAST 2008, pp. 576–581. IEEE (2008)
29. Volksbank Freiburg eG: iTANplus – mehr Sicherheit mit der indizierten TAN,  
<http://www.volksbank-freiburg.de/itan.cfm?CFID=10869033&CFTOKEN=34249989&rand=1246061956151>
30. Volksbank Rhein-Ruhr eG: Bankgeschäfte online abwickeln: Mit Sm@rtTAN optic bequem und sicher im Netz, <http://www.voba-rhein-ruhr.de/privatkunden/ebank/SMTOP.html>
31. Volksbank Solling eG: Sm@rt-TAN-plus, <http://www.volksbank-solling.de/flycms/de/html/913/-/Smart+TAN+plus.html>
32. Weigold, T., Kramp, T., Hermann, R., Höring, 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)

# Certified Lies: Detecting and Defeating Government Interception Attacks against SSL<sup>\*</sup> (Short Paper)

Christopher Soghoian and Sid Stamm

Center for Applied Cybersecurity Research, Indiana University  
`chris@soghoian.net, sid@sidstamm.com`

**Abstract.** This paper introduces the *compelled certificate creation attack*, in which government agencies may compel a certificate authority to issue false SSL certificates that can be used by intelligence agencies to covertly intercept and hijack individuals' secure Web-based communications.

## 1 Introduction

Consider a hypothetical situation where an executive is in a foreign country for a series of trade negotiations. After a day of meetings, she logs in to her corporate webmail account using her company-provided laptop and the hotel wireless network. Relying on the training she received from her company's IT department, she makes certain to look for the SSL encryption lock icon in her web browser, and only after determining that the connection is secure does she enter her login credentials and then begin to upload materials to be shared with her colleagues. However, unknown to the executive, the foreign government has engaged in a sophisticated man-in-the-middle attack, and is able to covertly intercept the executive's SSL encrypted connections. Agents from the state security apparatus leak details of her communications to the foreign company with whom she is negotiating, who use the information to gain an upperhand in the negotiations. While this scenario is fictitious, the vulnerability is not.

The security and confidentiality of millions of Internet transactions per day depend upon the Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocol. At the core of this system are a number of *Certificate Authorities* (CAs), each of which is responsible for verifying the identity of the entities to whom they grant SSL certificates. It is because of the confidentiality and authenticity provided by the CA based *public key infrastructure* that users around the world can bank online, engage in electronic commerce and communicate with their friends and loved ones about the most sensitive of subjects without having to worry about malicious third parties intercepting and deciphering their communications.

---

\* The full length version of this paper is available at [www.dubfire.net](http://www.dubfire.net). The authors hereby permit the use of this paper under the terms of the Creative Commons Attribution 3.0 United States license.

While not completely obvious, the CAs are all trusted equally in the SSL public key infrastructure, a problem amplified by the fact that the major web browsers trust hundreds of different firms to issue certificates for any site. Each of these firms can be compelled by their national government to issue a certificate for any particular website that all web browsers will trust without warning. Thus, users around the world are put in a position where their browser entrusts their private data, indirectly, to a large number of governments (both foreign and domestic) whom these individuals may not ordinarily trust.

In this paper, we introduce a new attack, the *compelled certificate creation attack*, in which government agencies compel (via a court order or some other legal process) a CA to issue false certificates that are then used by law enforcement and intelligence agencies to covertly intercept and hijack individuals' secure communications. In order to protect users from these powerful government adversaries, we introduce a lightweight defensive browser add-on that detects and thwarts such attacks.

## 2 Certificate Authorities and the Browser Vendors

*“[Browser vendors] and users must be careful when deciding which certificates and certificate authorities are acceptable; a dishonest certificate authority can do tremendous damage.”*

— RFC 2246, The TLS Protocol 1.0 [1]

CAs play a vital role in the SSL *public key infrastructure* (PKI). Each CA's main responsibility is to verify the identity of the entity to which it issues a certificate. Thus, when a user visits <https://www.bankofamerica.com>, her browser will inform her that the bank's certificate is valid, was issued by VeriSign, and that the website is run by Bank of America. It is because of the authenticity and confidentiality guaranteed by SSL that the user can continue with her transaction without having to worry that she is being phished by cyber-criminals.

CAs generally fall into one of three categories: Those trusted by the browsers (“root CAs”), those trusted by one of the root CAs (“intermediate CAs” or “subordinate CAs”), and those neither trusted by the browsers nor any intermediate CA (“untrusted CAs”). Furthermore, intermediate CAs do not necessarily have to be directly verified by a root CA — but can be verified by another intermediate CA, as long as the *chain of trust* eventually ends with a root CA. The problem, however, is that each of the hundreds of different root CAs are equally trusted by the browsers to issue certificates for any site.

From the end users' perspective, root CAs and intermediate CAs are functionally equivalent. A website that presents a certificate signed by either form of CA will cause the users' browser to display a lock icon and to change the color of the location bar. Whereas certificates verified by an untrusted CA and those self-signed by the website owner will result in the display of a security warning, which for many non-technical users can be scary [2], confusing, and difficult to bypass in order to continue navigating the site [3].

It is important to note that there are no technical restrictions in place that prohibit a CA from issuing a certificate to a malicious third party. Thus, both the integrity of the CA based public key infrastructure and the security users' communications depend upon hundreds of CAs around the world choosing to do the right thing. Unfortunately, as will soon be clear, any one of those CAs can become the weakest link in the chain.

### 3 Compelled Assistance

Many governments routinely compel companies to assist them with surveillance. Telecommunications carriers and Internet service providers are frequently required to violate their customers' privacy — providing the government with email communications, telephone calls, search engine records, financial transactions and geo-location information.

In the United States, the legal statutes defining the range of entities that can be compelled to assist in electronic surveillance by law enforcement and foreign intelligence investigators are remarkably broad [4]. Examples of compelled assistance using these statutes include a secure email provider that was required to place a covert back door in its product in order to steal users' encryption keys [5], and a consumer electronics company that was forced to remotely enable the microphones in a suspect's auto-mobile dashboard GPS navigation unit in order to covertly record their conversations [6].

Outside of the United States and other democratic countries, specific statutory authority may be even less important. The Chinese government, for example, has repeatedly compelled the assistance of telecommunications and technology companies in assisting it with its surveillance efforts [7, 8].

Just as phone companies and email providers can be forced to assist governments in their surveillance efforts, so too can SSL certificate authorities. The *compelled certificate creation attack* is thus one in which a government agency requires a domestic certificate authority to provide it with false SSL certificates for use in surveillance.

The technical details of this attack are simple, and do not require extensive explanation. Each CA already has an infrastructure in place with which it is able to issue SSL certificates. In this compelled assistance scenario, the CA is merely required to skip the identity verification step in its own SSL certificate issuance process.

When compelling the assistance of a CA, the government agency can either require the CA to issue it a specific certificate for each website to be spoofed, or, more likely, the CA can be forced to issue a intermediate CA certificate that can then be re-used an infinite number of times by that government agency, without the knowledge or further assistance of the CA. Furthermore, such an intermediate issuing CA can be installed into surveillance appliances already available on the market and quickly deployed to intercept any traffic [9].

## 4 Protecting Users

The major web browsers are currently vulnerable to the compelled certificate creation attack, and we do not believe that any of the existing privacy enhancing browser add-ons sufficiently protect users without significantly impacting browser usability.

In an effort to significantly reduce the impact of this attack upon end-users, we have created *Certlock*, a lightweight add-on for the Firefox browser. Our solution employs a Trust-On-First-Use (TOFU) policy (this is also known as ‘leap-of-faith’ authentication) [10, 11], reinforced with a policy that the country of origin for certificate issuing does not change in the future. Specifically, our solution relies upon caching CA information, that is then used to empower users to leverage country-level information in order to make common-sense trust evaluations.

In this section, we will outline the motivations that impacted the design of our solution, discuss our belief in the potential for users to make wise country-level trust decisions, and then explore the technical implementation details of our prototype add-on.

*Design Motivations.* The compelled certificate creation attack is a classic example of a low probability, high impact event [12]. The vast majority of users are extremely unlikely to experience it, but for those who do, very bad things are afoot. As such, it is vital that any defensive technique have an extremely low false positive rate, yet be able to get the attention of users when an attempted SSL session hijacking is detected.

*Country-Based Trust.* We believe that many consumers are quite capable of making basic trust decisions based on country-level information. We are not alone in this belief. Since March 2010, Google has been providing country-level warnings to users of its Google Mail service when it detects that their account has been accessed from a potentially suspect IP address in a different country [13].

Thus, a consumer whose banking sessions are normally encrypted by a server presenting a certificates signed by a US based CA might become suspicious if told that her US based bank is now using a certificate signed by a Tunisian, Latvian or Serbian CA.

To make this trust evaluation, she doesn’t have to study the detailed business policies of the foreign CA, she can instead rely on geographical prejudice, and ask herself why her Iowa based bank is suddenly doing business in Eastern Europe. In order to empower users to make such country-level evaluations of trust, CertLock leverages the wealth of historical browsing data kept by the browser.

Likewise, individuals located in countries with oppressive governments may wish to know if their communications with servers located in foreign democracies are suddenly being facilitated by a domestic (or state controlled) CA.

*Avoiding False Positives.* A simplistic defensive add-on aimed at protecting users from compelled certificate creation attacks could simply cache all certificates

encountered during browsing sessions, and then warn the user any time they encounter a certificate that has changed. In fact, such an add-on, Certificate Patrol, already exists [14].

Unfortunately, this approach is likely to generate too many false positives. Each time a website intentionally changes its certificate, the browser displays a warning that will needlessly scare and soon desensitize users. There are many legitimate scenarios where certificates change. For example: Old certificates expire; certificates are abandoned and or revoked after a data breach that exposed the server private key; and many large enterprises that have multiple SSL accelerator appliances serving content for the same domain use a different certificate for each device [15].

By adopting a Trust-On-First-Use policy, we assume that if a website starts using a different certificate issued by the same CA that issued its previous certificate, there is no reason to warn the user. This approach enables us to significantly reduce the false positive rate, while having little impact on our ability to protect users from a variety of threats.

We also believe that there is little reason to warn users if a website switches CAs within the same country. As our threat model is focused on a government adversary with the power to compel any domestic CA into issuing certificates at will, we consider CAs within a country to be equals. That is, a government agency able to compel a new CA into issuing a certificate could just as easily compel the original CA into issuing a new certificate for the same site. Since we have already opted to not warn users in that scenario (described above), there is no need to warn users in the event of a same-country CA change.

*Implementation Details.* Our Certlock solution is currently implemented as an add-on to the Firefox browser. Because the Firefox browser already retains history data for all visited websites, we have simply modified the browser to cause it to retain slightly more information. Thus, for each new SSL protected website that the user visits, a Certlock enabled browser also caches the following additional certificate information: (a) A hash of the certificate, (b) the country of the issuing CA, (c) the name of the CA, (d) the country of the website, (e) the name of the website and (f) the entire chain of trust up to the root CA.

When a user re-visits a SSL protected website, Certlock first calculates the hash of the site's certificate and compares it to the stored hash from previous visits. If it hasn't changed, the page is loaded without warning. If the certificate has changed, the CAs that issued the old and new certificates are compared. If the CAs are the same, or from the same country, the page is loaded without any warning. If, on the other hand, the CAs' countries differ, then the user will see a warning.

## 5 Related Work

Over the past decade, many people in the security community have commented on the state of the SSL public key infrastructure, and the significant trust placed in the CAs [16–18].

In 1998, James Hayes of the US National Security Agency published a paper that focused specifically on the threat of rogue insiders within a Certificate Authority [19]. Although the technical details of the threat outlined by Hayes are largely the same as the scenario on which we have focused (albeit with vastly different legal and policy consequences), Hayes did not address the threat of government compelled certificate creation. It is unclear if he was simply unaware of this scenario, or if the topic was too sensitive for him to discuss, given his employer. In his paper, Hayes proposed a technical solution to address the insider threat, which relied on users configuring various per-site attributes within their browser that would be used to evaluate each new site's certificate.

Crispo and Lomas also proposed a certification scheme designed to detect rogue CAs [20], while the Monkeysphere project has created a system that replaces the CA architecture with the OpenPGP web of trust [21].

Ian Grigg has repeatedly sought to draw attention to both the potential conflict of interest that some CAs have due to their involvement in other forms of surveillance, and the power of a court order to further compel these entities to assist government investigations [22–24]. In particular, in 2005, Grigg and Shostack filed a formal complaint with ICANN over the proposal to award VeriSign control of .net domain name registration, arguing that the firm's surveillance products created a conflict of interest [25].

In recent years, several browser-based tools have been created to help protect users against SSL related attacks. Kai Engert created Conspiracy, a Firefox add-on that provides country-level CA information to end-users in order to protect them from compelled certificate creation attacks. The Conspiracy tool displays the flag of the country of each CA in the chain of trust in the browser's status bar [26]. Thus, users must themselves remember the country of the CAs that issue each certificate, and detect when the countries have changed. We believe, like Herley [27], that this is an unreasonable burden to place upon end-users, considering how rarely the compelled certificate creation attack is likely to occur.

Wendlandt *et al.* created Perspectives, a Firefox add-on that improves the Trust-On-First-Use model used for websites that supply self-signed SSL certificates [28]. In their system, the user's browser securely contacts one of several notary servers, who in turn independently contact the webserver and obtain its certificate. In the event that an attacker is attempting to perform a man in the middle attack upon the user, the fact that the attacker-supplied SSL certificate, and those supplied by the Perspectives notary servers differ will be a strong indicator that something bad has happened. Unfortunately, the Perspectives system requires that users provide the Perspectives notaries with a real-time list of the secure sites they visit.

Alicherry and Keromytis have improved upon the Perspectives design with their DoubleCheck system [29], substituting Tor exit nodes for special notary servers. Because the Tor network anonymizes the individual user's IP address, there is no way for the Tor exit nodes to know who is requesting the certificate for a particular SSL website. While the authors solved the major privacy issues that plague the Perspectives scheme, their choice of Tor carries its own cost:

Latency. Their system adds an additional second of latency to every new SSL connection, and up to 15 seconds for visits to new self-signed servers. We believe that this additional latency is too much to ask most users to bear, particularly if the chance of them encountering a rogue CA is so low.

Herzberg and Jbara created TrustBar, a Firefox add-on designed to help users detect spoofed websites. The browser tool works by prominently displaying the name of the CA that provided the site's certificate, as well as allowing the user to assign a per-site name or logo, to be displayed when they revisit to each site [30].

Tyler Close created Petname Tool, a Firefox add-on that caches SSL certificates, and allows users to assign a per-site phrase that is displayed each time they revisie the site in the future. In the event that a user visits a spoofed website, or a site with the same URL that presents a certificate from a different CA, the user's specified phrase will not be displayed [31].

In May 2008, a security researcher discovered that the OpenSSL library used by several popular Linux distributions was generating weak cryptographic keys. While the two-year old flaw was soon fixed, SSL certificates created on computers running the flawed code were themselves open to attack [32, 33]. Responding to this flaw, German technology magazine Heise released the Heise SSL Guardian for the Windows operating system, which warns users of Internet Explorer and Chrome when they encounter a weak SSL certificate [34].

In December 2008, Stevens *et al.* demonstrated that flaws in the MD5 algorithm could be used to create rogue SSL certificates (without the knowledge or assistance of the CA). In response, CAs soon accelerated their planned transition to certificates using the SHA family of hash functions [35]. As an additional protective measure, Márton Anka developed an add-on for the Firefox browser to detect and warn users about certificate chains that use the MD5 algorithm for RSA signatures [36].

Jackson and Barth devised the ForceHTTPS system to protect users who visit HTTPS protected websites, but who are vulnerable to man in the middle attacks due to the fact that they do not type in the `https://` component of the URL [37]. This system has since been formalized into the HTTP Strict Transport Security (HSTS) standard proposal [38], to which multiple browsers are in the process of adding support. While this system is designed to enable a website to hint to the browser that future visits should always occur via a HTTPS connection, this mechanism could be extended to enable a website to lock a website to a particular CA, or CAs of a specific country.

## 6 Conclusion and Future Work

In this paper, we introduced the compelled certificate creation attack and presented evidence that suggests that governments may be subverting the CA based public key infrastructure. In an effort to protect users from these powerful adversaries, we introduced a lightweight defensive browser based add-on that detects

and thwarts such attacks. Finally, we use reductive analysis of governments' legal capabilities to perform an adversarial threat model analysis of the attack and our proposed defensive technology.

Our browser add-on is currently just a prototype, and we plan to improve it in the future. We plan to explore the possibility of expanding the country-level trust model to regions, such as the European Union, where, for example, residents of the Netherlands may be willing to trust Belgian CAs. We are also considering adding a feature that will enable users to voluntarily submit potentially suspect certificates to a central server, so that they can be studied by experts. Such a feature, as long as it is opt-in, does not collect any identifiable data on the user, and only occurs when potentially rogue certificates are discovered, would have few if any privacy issues.

Ultimately, the threats posed by the compelled certificate creation attack cannot be completely eliminated via our simple browser add-on. The CA system is fundamentally broken, and must be overhauled. DNSSEC may play a significant role in solving this problem, or at least reducing the number of entities who can be compelled to violate users' trust. No matter what system eventually replaces the current one, the security community must consider compelled government assistance as a realistic threat, and ensure that any solution be resistant to such attacks.

**Acknowledgements.** Thanks to Kevin Bankston, Matt Blaze, Kelly Caine, Jon Callas, Allan Friedman, Jennifer Granick, Markus Jakobsson, Dan Kaminsky, Moxie Marlinspike, Eddy Nigg, Eli O and Adam Shostack for their useful feedback.

## References

1. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), Obsoleted by RFC 4346, updated by RFCs 3546, 5746 (January 1999)
2. Nightingale, J.: SSL Question Corner. meandering wildly (blog) (August 5, 2008), <http://blog.johnath.com/2008/08/05/ssl-question-corner/>
3. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying wolf: An empirical study of SSL warning effectiveness. In: Proceedings of the 18th Usenix Security Symposium (August 2009)
4. Soghoian, C.: Caught in the cloud: Privacy, encryption, and government back doors in the web 2.0 era. Journal on Telecommunications and High Technology Law (forthcoming)
5. Singel, R.: PGP Creator Defends Hushmail. Wired News Threat Level Blog (November 19, 2007), <http://www.wired.com/threatlevel/2007/11/pgp-creator-def>
6. McCullagh, D.: Court to FBI: No spying on in-car computers. CNET News (November 19, 2003), [http://news.cnet.com/2100-1029\\_3-5109435.html](http://news.cnet.com/2100-1029_3-5109435.html)
7. Markoff, J.: Surveillance of skype messages found in china. The New York Times (October 1, 2008), <http://www.nytimes.com/2008/10/02/technology/internet/02skype.html>
8. Jacobs, A.: China requires censorship software on new pcs. The New York Times (June 8, 2009), <http://www.nytimes.com/2009/06/09/world/asia/09china.html>

9. Singel, R.: Law Enforcement Appliance Subverts SSL. Wired News Threat Level Blog (March 24, 2010), <http://www.wired.com/threatlevel/2010/03/packet-forensics/>
10. Stajano, F., Anderson, R.J.: The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks. In: Malcolm, J.A., Christianson, B., Crispo, B., Roe, M., et al. (eds.) Security Protocols 1999. LNCS, vol. 1796, pp. 172–182. Springer, Heidelberg (2000)
11. Arkko, J., Nikander, P.: Weak Authentication: How to Authenticate Unknown Principals without Trusted Parties. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2002. LNCS, vol. 2845, pp. 5–19. Springer, Heidelberg (2004)
12. Bussiere, M., Fratzscher, M.: Low probability, high impact: Policy making and extreme events. Journal of Policy Modeling 30(1), 111–121 (2008)
13. Diwanji, P.: Detecting suspicious account activity. The Official Gmail Blog (March 24, 2010), <http://gmailblog.blogspot.com/2010/03/detecting-suspicious-account-activity.html>
14. Certificate patrol (2010), <http://patrol.psyced.org/>
15. Kaminsky, D.: Email conversation with author (February 28, 2010)
16. Gillmor, D.K.: Technical Architecture shapes Social Structure: an example from the real world (February 21, 2007), <http://lair.fifthhorseman.net/~dkg/tls-centralization/>
17. Peter SJF Bance. Ssl: Whom do you trust? (April 20, 2005), <http://www.minstrel.org.uk/papers/2005.04.20-ssl-trust.pdf>
18. Ed Gerck. First published online by the MCWG at <http://mcwg.org/cert.htm> (April 1997). Invited talk at the Black Hat Briefings 1999, Las Vegas, NV, July 7-8 (1999). Published by The Bell, ISSN 1530-048X, Vol. 1, No. 3, p. 8 (July 2000), <http://www.thebell.net/papers/certover.pdf>
19. Hayes, J.M.: The problem with multiple roots in web browsers - certificate masquerading. In: WETICE 1998: Proceedings of the 7th Workshop on Enabling Technologies, pp. 306–313. IEEE Computer Society, Washington, DC (1998)
20. Crispo, B., Lomas, M.: A Certification Scheme for Electronic Commerce. In: Lomas, M. (ed.) Security Protocols 1996. LNCS, vol. 1189, pp. 19–32. Springer, Heidelberg (1997)
21. Monkeysphere (2010), <http://web.monkeysphere.info/>
22. Grigg, I.: VeriSign's conflict of interest creates new threat. Financial Cryptography (blog) (September 1, 2004), <http://financialcryptography.com/mt/archives/000206.html>
23. Grigg, I.: PKI considered harmful (October 14, 2008), [http://iang.org/ssl/pki\\_considered\\_harmful.html](http://iang.org/ssl/pki_considered_harmful.html)
24. Grigg, I.: Why the browsers must change their old SSL security (?) model. In: Financial Cryptography (blog) (March 24, 2010), <http://financialcryptography.com/mt/archives/001232.html>
25. Grigg, I., Shostack, A.: VeriSign and Conflicts of Interest (February 2, 2005), <http://forum.icann.org/lists/net-rfp-verisign/msg00008.html>
26. Engert, K.: Conspiracy — A Mozilla Firefox Extension (March 18, 2010), <http://kuix.de/conspiracy/>
27. Herley, C.: So long, and no thanks for the externalities: the rational rejection of security advice by users. In: NSPW 2009: Proceedings of the 2009 Workshop on New Security Paradigms Workshop, pp. 133–144 (September 2009)

28. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: improving ssh-style host authentication with multi-path probing. In: ATC 2008: USENIX 2008 Annual Technical Conference on Annual Technical Conference, pp. 321–334. USENIX Association, Berkeley (2008)
29. Alicherry, M., Keromytis, A.D.: Doublecheck: Multi-path verification against man-in-the-middle attacks. In: ISCC 2009: IEEE Symposium on Computers and Communications, pp. 557–563. IEEE, Piscataway (2009)
30. Herzberg, A., Jbara, A.: Security and identification indicators for browsers against spoofing and phishing attacks. ACM Trans. Internet Technol. 8(4), 1–36 (2008)
31. Close, T.: Petname tool (2005), <http://www.warterken.com/user/PetnameTool/>
32. Ahmad, D.: Two Years of Broken Crypto: Debian’s Dress Rehearsal for a Global PKI Compromise. IEEE Security and Privacy 6, 70–73 (2008)
33. Yilek, S., Rescorla, E., Shacham, H., Enright, B., Savage, S.: When private keys are public: results from the 2008 Debian OpenSSL vulnerability. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, pp. 15–27. ACM, New York (2009)
34. The H Security. heise SSL Guardian: Protection against unsafe SSL certificates (July 4, 2008), [www.h-online.com/security/features/Heise-SSL-Guardian-746213.html](http://www.h-online.com/security/features/Heise-SSL-Guardian-746213.html)
35. 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: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 55–69. Springer, Heidelberg (2009)
36. Anka, M.: SSL Blacklist 4.0 (January 31, 2010),  
<http://www.codefromthe70s.org/sslblacklist.aspx>
37. Jackson, C., Barth, A.: Forcehttps: protecting high-security web sites from network attacks. In: WWW 2008: Proceeding of the 17th International Conference on World Wide Web, pp. 525–534. ACM, New York (2008)
38. Hodges, J., Jackson, C., Barth, A.: Strict Transport Security (December 18, 2009),  
<lists.w3.org/Archives/Public/www-archive/2009Dec/att-0048/draft-hodges-strict-transport-sec-06.plain.html>

# Proximax: Measurement-Driven Proxy Dissemination

Damon McCoy<sup>1</sup>, Jose Andre Morales<sup>2</sup>, and Kirill Levchenko<sup>1</sup>

<sup>1</sup> University of California at San Diego  
`{dlmccoy,klevchen}@cs.ucsd.edu`

<sup>2</sup> Institute for Cyber Security, University of Texas at San Antonio  
`jose.morales@utsa.edu`

**Abstract.** Many people currently use proxies to circumvent government censorship that blocks access to content on the Internet. Unfortunately, the dissemination channels used to distribute proxy server locations are increasingly being monitored to discover and quickly block these proxies. This has given rise to a large number of *ad hoc* dissemination channels that leverage trust networks to reach legitimate users and at the same time prevent proxy server addresses from falling into the hands of censors. To address this problem in a more principled manner, we present *Proximax*, a robust system that continuously distributes pools of proxies to a large number of channels. The key research challenge in *Proximax* is to distribute the proxies among the different channels in a way that maximizes the usage of these proxies while minimizing the risk of having them blocked. This is challenging because of two conflicting goals: widely disseminating the location of the proxies to fully utilize their capacity and preventing (or at least delaying) their discovery by censors.

## 1 Introduction

Internet censorship is a fact of life for most of the world’s population. While the stated purpose of such censorship is to protect users from harmful content, political news and commentary are often blocked as well. Reports of government censorship in the wake of anti-government protests in Burma and Iran [1–4] underscore the growing role of the Internet in enabling political speech and organization, as well as the steps taken by governments to control it.

To circumvent censorship, users rely on external Internet *proxies* [5, 6]. In the simplest case, this is simply a SOCKS proxy supporting encrypted connections (e.g. TLS). Encrypting the connection to the proxy provides content *confidentiality*, thus bypassing content filtering.<sup>1</sup> Widespread use of proxies has in turn led to *secondary* censorship: governments identifying and blocking the proxies themselves [7–9] (at the network level).

Individuals and organizations providing proxy service are thus faced with the additional challenge of advertising their resources to their target audience

---

<sup>1</sup> In addition to confidentiality, proxies also provide *anonymity* by aggregating all users behind a single host.

while preventing the same information from falling into the hands of censorship authorities. The Tor network [10], for example, has recently added *bridge relays* that offer an improved level of censorship resistance by relay traffic to publicly-advertised Tor core routers (which have become blocked by censors) [11]. In response, the Chinese government has enumerated and blocked all Tor bridge relays advertised via the website distribution channel [9].

Today, addresses of open proxies are distributed via *ad hoc* mailing lists or via social networking sites such as Facebook and Twitter. Such “trust networks” provide a degree of protection against discovery by censorship authorities; however they also limit the population served by these proxies. Negotiating this trade-off between publicity and secrecy is no easy task: advertising to more people means greater effectiveness, but also greater risk being blocked.

Our proposed solution is to cast the problem as that of maximizing *yield*, that is, the number of user-hours of service provided by a set of proxies. Proxy addresses are given to a set of *registered users* to advertise in any manner they wish. *Proximax* provides a means of estimating each user’s effectiveness, and a policy for choosing the most effective users for advertising proxies, with respect to our objective of maximizing total system yield.

## 2 Design

*Proximax* is a proxy distribution system in which users themselves are the means of disseminating proxy addresses. Users disseminate proxy addresses to their friends in any manner they wish, whether via a private mailing list, social networking site, or in person. Their friends, in turn, pass the information on to their friends, and so on.

There is a special set of users—*registered users*—who learn proxy addresses directly from the system; all other users learn about proxies from other users. *Proximax* is responsible for determining the effectiveness of each registered user, measured as the number of end users attracted to a proxy, and how long the proxy lasted before being blocked. Based on this estimate, when a new proxy becomes available, *Proximax* chooses which registered users should advertise it. In the remainder of this section we describe how such a system can be implemented; in Section 3 we formally describe how to estimate attracted usage and choose which registered users should be used to advertise a new resource.

### 2.1 Challenges

Anti-censorship systems typically distribute lists of proxies to end users through informal social networks such as message boards, Twitter, instant messages, and mailing lists [6, 12, 13]. In contrast to previously proposed open proxy distribution systems [14, 15], *Proximax* tracks both the usage rate and the risk of proxies distributed via different channels being blocked. The goal of *Proximax* is to maximize the useful service of the limited number of proxies that we have to disseminate. It is important to note that the amount of useful service of a proxy

depends both on how many end users it attracts and on how long the proxy remains in operation (before being blocked). We measure of useful service as the number of user-hours a proxy provides, which we refer to as its *yield*. Our objective is to maximize total system yield given a limited number of proxies. To do so we must balance two conflicting goals: widely disseminating proxy addresses to fully utilize their capacity and evading discovery by censors.

## 2.2 System Tasks

The operation of *Proximax* consists of three main tasks illustrated in Figure 1. We present an overview of these tasks here. **Disseminating Proxies.** We assume that there is a trusted group of administrators who run the *Proximax* system. Administrators have a full list of proxies. Each registered user receives an individualized host name which they can disseminate to friends. The fact that the host name is individualized allows us to track how many additional end users a channel brings to a set of proxies, which affects their standing in the system (more on that later). We envision that a channel could be a private email list, social networking site, or a censorship-resistant publishing tool, such as Collage [16]. These addresses may be discovered by the adversary and blocked either by infiltrating the distribution chain or some other method. Our system is built on the assumption that this will eventually happen with all addresses. We also assume that *Proximax* can collect statistics on the usage of these proxies by country to determine if they should be removed from the system when they are either blocked or fail.

**Managing Channels.** As previously stated, each registered user has an individualized host name (which take the form of a unique domain name registered with DNS). In order to make it difficult to discover and ban channels we piggyback on the DNS infrastructure, using a technique, commonly employed by botnets and malware distributors, called *fast flux* [17]. As part of this technique *Proximax* will register multiple proxies to the same domain name and uses round-robin DNS along with short Time-To-Live (TTL) values to create a constantly changing list of proxies for that single domain name. This additionally allows *Proximax* to automatically load balance resources by adding and removing proxies based on current utilization levels.

The adversary can block the channel (DNS blocking) at which point *Proximax* will issue another individualized host name to that channel. The adversary can

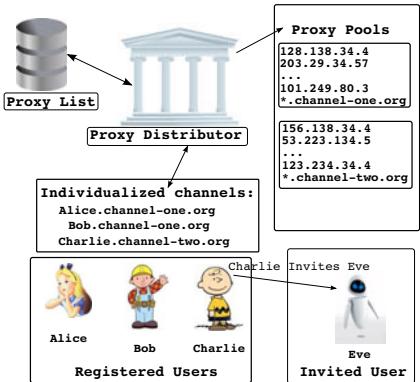


Fig. 1. *Proximax* system components

also block all of the individual proxy addresses (IP level blocking) assigned to that channel when it was discovered<sup>2</sup>. If they block all the individual proxies but not the channel *Proximax* might still want to provide a new host name to the channel to force the adversary to discover the new name.

**Inviting Users.** There must be some mechanism within *Proximax* to allow the registration of new users. We feel that an open registration process would allow the adversary to flood the system with fictitious registered users. Thus, we choose to only allow new users to enter the system through an invitation by an existing registered user. These invites should be limited and the number of new invitations will be based on the current under-utilization of the system (if any) to increase the usage rate of resources. Which potential new registered users are granted invitations will be based on their pre-existing performance, if any, as a non-registered user and historical performance of the registered user issuing the invitation. The performance of current registered users invited by the same inviter is also considered. If the inviting registered user has never issued an invitation, granting a new invitation is based solely on the inviter's direct performance.

### 3 Analysis

The preceding section described the overall system for allocating and managing proxies; in this section we fill in the details of how user effectiveness (i.e. attracted user-hours) is estimated, and how this estimate is used to decide which registered users should be chosen to advertise a new proxy.

#### 3.1 Model

Abstractly, we model our problem as one of choosing a set of *dissemination channels* (or more simply *channels*) to use to advertise a set of *resources*. In our case, proxies play the role of resources and registered users the role of dissemination channels. Advertising a resource attracts a certain level of resource usage based on the channel used to advertise the resource. Advertising a resource also carries the risk of the resource being discovered and blocked, rendering it unusable.

Formally, we have a set of  $m$  identical resources and  $n$  channels. Each channel has an associated level of *usage*, denoted  $u_j$ . In our system, we measure usage as the number of user-hours the channel attracts per day. We assume that the usage level is stable or changes only slowly over time, and thus easy to estimate.

**Table 1.** Parameters and notation used in Section 3

---

$m$	Number of resources.
$n$	Number of channels.
$\gamma$	Intrinsic resource risk (parameter).
$R_i$	Set of resources advertised via channel $i$ .
$t_i$	Resource $i$ lifetime (measured).
$\lambda_j$	Channel $j$ risk (unknown).
$u_j$	Channel $j$ attracted usage.
$A_i$	Total resource risk; Eq. (1).
$U_i$	Total resource $i$ usage; Eq. (1).

---

<sup>2</sup> This will affect other channels that are sharing this same resource.

Each channel also contributes a level of risk to the resource. We model this risk as a Poisson process; that is, we assume that during each infinitesimal period of time there is a fixed probability of the resource being blocked or shut down. This risk is quantified by the Poisson process rate parameter  $\lambda_j$ , where the probability of a resource being shut down at or before time  $t$  as a result of being advertised on channel  $j$  is  $1 - e^{-\lambda_j t}$ . If the Poisson processes associated with each channel are independent, then the rate parameters are additive: a resource is advertised on two channels  $j$  and  $j'$  has rate parameter  $\lambda_j + \lambda_{j'}$ , so that the probability of a resource being shut down at or before time  $t$  as a result of being advertised on channel  $j$  and channel  $j'$  is  $1 - e^{-(\lambda_j + \lambda_{j'})t}$ . Note that this holds only if each channel is independent with respect to its risk of being censored; while this may not be a realistic assumption, we believe it provides a reasonable first-order approximation.

Because resources are identical and can be advertised immediately (Section 2), there is no benefit to advertising more than one available resource on a channel.<sup>3</sup> We also assume that resources have a small user-specified *intrinsic risk*, denoted  $\gamma$ , which models the possibility that a resource will not be available indefinitely even if it is not advertised.<sup>4</sup> Let  $A_i$  denote the set of channels advertising resource  $i$ . Then the *total risk* and *total usage* of resource  $i$  are, respectively:

$$\Lambda_i = \gamma + \sum_{j \in A_i} \lambda_j \quad \text{and} \quad U_i = \sum_{j \in A_i} u_j. \quad (1)$$

The *yield* of a resource is the product of its usage and lifetime. For example, a proxy with a usage level of 100 user-hours per day lasted 5 days before being blocked, its yield would be  $100 \times 5 = 500$  user-hours. The expected lifetime of a resource is the inverse of its risk, that is,  $1/\Lambda_i$ . The expected yield is thus  $U_i/\Lambda_i$ . Our goal is to maximize the expected total system yield, which is simply the sum of the expected yields of each resource. We do this by choosing which resources to advertise on which channels, which requires estimating channels' attracted usage and risk. We assume that it is possible to measure the usage rate attracted by each channel, as described in Section 2. To avoid sharp fluctuations, the estimate may be smoothed using an exponentially-weighted moving average.

### 3.2 Estimating Risk

Because a resource may be advertised using multiple channels, the risk associated with a given channel cannot be sampled directly. When a resource is blocked, we have no way of telling which channel is responsible (in our case, through which channel the censorship authorities found the resource). However from the sample of resource lifetimes, we can compute a maximum likelihood estimate of the risk parameters. Let  $t_i$  denote the lifetime of resource  $i$ . The log-likelihood function of our sample of  $m$  resource is:

<sup>3</sup> We assume each resource is not capacity-limited. The model can be extended to capacity-limited resources as well.

<sup>4</sup> Alternately, we can view intrinsic risk as a kind of *discount factor* discouraging resource underutilization. Section 3.3 for more on this parameter.

$$\ell = \log \prod_{i=1}^m \Lambda_i e^{-\Lambda_i t_i} = \sum_{i=1}^m (\log \Lambda_i - \Lambda_i t_i). \quad (2)$$

The partial derivatives of  $\ell$  with respect to  $\lambda_j$  are:

$$\frac{\partial \ell}{\partial \lambda_j} = \sum_{i \in R_j} (\Lambda_i^{-1} - t_i) \quad (3)$$

The above optimization can be carried out numerically, however a first-order approximation is to attempt to equate  $\Lambda_i t_i = 1$ . In some cases this may lead to an unsatisfiable system of constraints, so we minimize the sum of squares of  $\Lambda_i t_i - 1$ , which is analytically tractable. Denote this sum:

$$E = \sum_{i=1}^m (\Lambda_i t_i - 1)^2, \quad (4)$$

from which we get the system of equations

$$\frac{\partial E}{\partial \lambda_j} = 2 \sum_{i=1}^m (\Lambda_i t_i - 1) t_i = 0. \quad (5)$$

If the number of channels outnumbers the number of resources (that is,  $m < n$ ), the resulting system of equations will be under-constrained. For example, if two channels are used identically (advertising the same set of resources), there is no way to separate their respective risks. To force a unique solution, we introduce another optimization step, minimizing the sum of squares of the risk parameters  $\lambda_j$ , subject to the linear constraints above. This gives us an estimate of the attracted risk associated with each channel, which we use in deciding which channels to use to advertise resources.

### 3.3 Resource Advertisement Policy

So far we have described how to estimate a channel's usage rate and risk, which we use to choose a resource to advertise on the channel. Our goal is to maximize the expected total yield, which is simply the sum of the expected yields of each resource (recall that the expected yield of a resource is its usage divided by risk,  $U_i/\Lambda_i$ ). When a channel becomes available, we simply choose the resource whose yield would increase the most as a result of being advertised on the new channel. Let  $\hat{j}$  denote the index of this channel. The increase in the yield of resource  $i$  is then given by

$$\Delta_i = \frac{u_{\hat{j}} + U_i}{\lambda_{\hat{j}} + \Lambda_i} - \frac{U_i}{\Lambda_i} = \frac{u_{\hat{j}} \Lambda_i - U_i \lambda_{\hat{j}}}{(\lambda_{\hat{j}} + \Lambda_i) \Lambda_i}. \quad (6)$$

Our advertisement policy is thus to choose the resource  $i$  which maximizes  $\Delta_i$ .

Note that the numerator of Eq. (6) implies that the increase in yield is positive if and only if  $u_{\hat{j}}/\lambda_{\hat{j}} > U_i/\Lambda_i$ , in other words, the channel expected yield  $u_{\hat{j}}/\lambda_{\hat{j}}$

must be greater than the resource's current expected yield  $U_i/A_i$ . This means that, for some low-usage high-risk channels, the best choice is *not to use them at all*. Intuitively, this is because the risk associated with a channel affects *all* users of a resource, so that the increased risk to the resource is not justified by the additional usage attracted by the channel.

Equation 6 highlights the trade-off inherent in *any* policy of advertising resources, namely the tradeoff between attracting more users or minimizing the risk to the resource. Our “knob” in controlling this trade-off is the intrinsic resource risk parameter  $\gamma$ . The intrinsic resource risk is the risk of losing a resource *even if we don't advertise it at all*. We conjecture several cases of losing a non-advertised resource such as: a proxy being setup and not well maintained, a proxy server taken offline and re-purposed, and proxy maintainers simply losing interest in our cause.

## 4 Discussion

*Proximax* is designed to be highly usable, easy to implement, and practical. To achieve these goals the system is left vulnerable to a number of attacks. We discuss some of these attacks and possible counter-measures that would likely impact the usability, implementation complexity, or practicality of *Proximax*.

**Independence of Adversaries.** To our knowledge, censoring nation states currently act independently, and do not share lists of discovered proxies with each other. Thus some proxies may only be blocked in some countries. Furthermore, certain channels may attract users in the same country due to factors such as common language or personal contacts. *Proximax* could optimize for this situation by detecting specifically which adversaries have blocked a resource and re-assign this resource to another channel that attracts users in a country where the proxy has not been blocked.

**Usage Inflation.** *Proximax* assumes that all usage of a resource is done by legitimate end users that are circumventing a censorship system. However, an adversary can be invited to join our system and inflate their standing (yield) by making dummy connections in order to accrue user-hours. This would cause the system to group the attacker with more resources which they can block. This problem is not specific to *Proximax*, of course; any system which can be fooled into allocating resources to fictitious users is vulnerable. To mitigate this, we can attempt to diversify the user-base by sub-linearly scaling usage credit assigned for attracting very similar users (e.g., users from the same IP address prefix).

**Delayed Blocking.** A smart adversary could infiltrate the system and gather large numbers of proxy addresses and channels and delay acting on this information for weeks or months before blocking them all at once. As part of our system we assume that proxies will be blocked or fail within days or weeks, thus delaying the blocking of proxies to gather more information would likely help prolong the expected longevity of proxies. This in turn would increase the yield of our limited resources, which is the goal of *Proximax*. Thus, we do not consider this much of a risk, since an adversary's user account reputation will drop quickly as she acts upon gathered information.

## 5 Conclusion

In this paper we have presented *Proximax*, an adaptive system for distributing addresses of open proxies that maximizes the yield of user-hours given a limited set of proxy resources. We sketch the different tasks of the system and show how to build an analytical model that uses measurements of the usage rate and blocking risk to intelligently allocate proxy resources among a large number of distribution channels. To our knowledge this is the first system to attempt to build a proxy distribution system that automatically adjusts the resources allocated to each channel and groups registered users together in shared pools of proxies based on similar blocking risk rates. As future work we plan on implementing *Proximax* to gain real measurements that will drive the refinement of our system and analytical model.

## References

1. Wilson, D.: Burmese internet taken down with ddos attack (2010), <http://www.techeye.net/internet/burmese-internet-taken-down-with-ddos-attack>
2. Moe, W.: No change as junta clamps down on suu kyi news (2010), [http://www.irrawaddy.org/article.php?art\\_id=20151](http://www.irrawaddy.org/article.php?art_id=20151)
3. Mackinnon, R.: No quick fixes for internet freedom (2010), <http://online.wsj.com/article/SB10001424052748704104104575622080860055498.html>
4. Huffington, A.: Facebook, twitter and the search for peace in the middle east (2010), [http://www.huffingtonpost.com/arianna-huffington/facebook-twitter-and-the-\\_b\\_788378.html?ir=Technology](http://www.huffingtonpost.com/arianna-huffington/facebook-twitter-and-the-_b_788378.html?ir=Technology)
5. Freegate - dynamic internet technology, <http://www.dit-inc.us/freegate>
6. Tor bridges specification, [http://gitweb.torproject.org/tor.git?a=blob\\_plain;hb=HEAD;f=doc/spec/bridges-spec.txt](http://gitweb.torproject.org/tor.git?a=blob_plain;hb=HEAD;f=doc/spec/bridges-spec.txt)
7. The proxy fight for iranian democracy (2009), <http://www.renesys.com/blog/2009/06/the-proxy-fight-for-iranian-de.shtml>
8. MacKinnon, R.: China's censorship arms race escalates (2009), [http://www.circleid.com/posts/20090928\\_chinas\\_censorship\\_arms\\_race\\_escalates/](http://www.circleid.com/posts/20090928_chinas_censorship_arms_race_escalates/)
9. Lewman, A.: Bridges and china, new thread (2010), <http://archives.seul.org/or/talk/May-2010/msg00264.html>
10. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
11. Dingledine, R., Mathewson, N.: Design of a blocking-resistant anonymity system (2007), [https://svn.torproject.org/svn/tor/tags/tor-0\\_2\\_0\\_19\\_alpha/doc/design-paper/blocking.pdf](https://svn.torproject.org/svn/tor/tags/tor-0_2_0_19_alpha/doc/design-paper/blocking.pdf)
12. Proxy china - free anonymous surfing, <http://www.proxychina.org/>
13. Psiphon design overview 1.0 (2009), [http://psiphon.ca/documents/Psiphon\\_Design\\_Overview\\_1\\_0.pdf](http://psiphon.ca/documents/Psiphon_Design_Overview_1_0.pdf)
14. Sovran, Y., Libonati, A., Li, J.: Pass it on: Social networks stymie censors. In: 7th International Workshop on Peer-to-Peer Systems, IPTPS (2008)
15. Mahdian, M.: Fighting Censorship with Algorithms. In: Boldi, P. (ed.) FUN 2010. LNCS, vol. 6099, pp. 296–306. Springer, Heidelberg (2010)
16. Burnett, S., Feamster, N., Vempala, S.: Chipping away at censorship firewalls with user-generated content. In: Usenix Security (August 2010)
17. Riden, J.: How fast-flux service networks work, <http://www.honeynet.org/node/132>

# BNymble

## More Anonymous Blacklisting at Almost No Cost (A Short Paper)

Peter Lofgren and Nicholas Hopper

University of Minnesota

**Abstract.** Anonymous blacklisting schemes allow online service providers to prevent future anonymous access by abusive users while preserving the privacy of all anonymous users (both abusive and non-abusive). The first scheme proposed for this purpose was Nymble, an extremely efficient scheme based only on symmetric primitives; however, Nymble relies on trusted third parties who can collude to de-anonymize users of the scheme. Two recently proposed schemes, Nymbler and Jack, reduce the trust placed in these third parties at the expense of using less-efficient asymmetric crypto primitives. We present BNymble, a scheme which matches the anonymity guarantees of Nymbler and Jack while (nearly) maintaining the efficiency of the original Nymble. The key insight of BNymble is that we can achieve the anonymity goals of these more recent schemes by replacing only the infrequent “User Registration” protocol from Nymble with asymmetric primitives. We prove the security of BNymble, and report on its efficiency.

## 1 Introduction

Anonymity networks like Tor [4] and JonDo [5] allow users to access online services while concealing the parties to any particular communication, by relaying this information through several intermediaries. While these networks are an important tool for circumventing online censorship and protecting freedom of speech, they are also a “mixed blessing” for the providers of online services. In particular, while anonymous access can expand the range of users that are able or willing to contribute to an online service, it can also allow misbehaving users to abuse the online service in a way that makes it difficult to hold them accountable. As a result, several service providers – including Wikipedia and Slashdot – have chosen to block contributions from known anonymity providers, despite the potential loss of interesting contributions.

To address this problem, Johnson *et al.* [9] (inspired by [8]) proposed the notion of an *anonymous blacklisting scheme*, which allows service providers (SPs) to maintain a “blacklist” such that non-abusive users can access the service anonymously; while users on the blacklist cannot access the service, but remain anonymous. Anonymous blacklisting schemes would allow SPs to benefit from anonymous contributions and simultaneously limit abuse.

The first such construction was Nymble [9,15,14]. Nymble constructs unlinkable authentication token sequences using hash chains. A pair of Trusted Third

Parties (TTPs), the Nymble Manager (NM) and Pseudonym Manager (PM), help SPs to link future tokens from abusive users so their access can be blocked. Unfortunately, these TTPs can easily collude to de-anonymize any user.

Since the proposal of Nymble, several schemes have attempted to improve on this trust requirement. On one end, schemes such as BLAC [11,12] and EPID [2] support anonymous blacklisting of misbehaved users with no TTP. In these schemes, SPs simply add authentication tokens associated with misuse to a blacklist. When a user produces a new authentication token, she must then prove that each token on the blacklist is not linked to her new token, requiring the SP to perform a modular exponentiation for each blacklist element for every access. PEREA [13] improved on this, reducing the cost of each authentication to  $O(k)$  modular exponentiations, by having each user prove that each of its last  $k$  tokens are not in a cryptographic accumulator of blacklisted tokens.

On the other hand, recent schemes such as Nymbler [7] and Jack [10] retain the TTPs from Nymble, while preventing colluding TTPs from fully de-anonymizing users. These schemes replace the symmetric primitives in Nymble with asymmetric primitives, essentially removing the dependence on blacklist size in exchange for weaker anonymity guarantees compared with BLAC. However, because they replace symmetric with asymmetric primitives, the cost of authentication and/or linking in these schemes are significantly higher than in the original Nymble.

**Our Contributions.** We start with a key insight: the attack that Nymbler and Jack prevent is collusion between the Pseudonym Manager and the SP or NM. Fortunately, the protocols involving the PM are the least frequently invoked, so their cost can be increased with comparatively little effect on the overall cost of authentication. We replace the Nymble PM’s linkable pseudorandom function with an information-theoretically unlinkable blind signature, while leaving the rest of Nymble unchanged. The resulting scheme, which we call BNymble, provides the same anonymity guarantees as Jack and Nymbler while preserving the lower cost of authentication and linking from Nymble. We report on experiments with a prototype implementation of BNymble, showing that the total cost of authentication increases by as little as 11% over Nymble, and compare this with the higher costs of Nymbler and Jack.

## 2 Background and Related Work

**Nymble.** Nymble [9,15,14] was the first anonymous blacklisting scheme to appear in the literature. In Nymble, in addition to the SP and the user, there are two Trusted Parties, the Pseudonym Manager (PM) and the Nymble Manager (NM). Nymble uses an authenticated symmetric encryption scheme  $E$ , a pseudorandom function  $F$ , a message authentication code  $MAC$  and two cryptographic hash functions (modeled as random oracles),  $f$  and  $g$ ; there are two secret keys  $KP$  and  $KN$  known to the PM and NM, respectively, additionally, the MAC key  $\kappa_{pn}$  is shared by the PM and NM and MAC Key  $\kappa_{ns}$  is shared by the NM and SP. Nymble divides time into “linkability windows,” during which a user’s actions can be linked together and these are then divided further into  $w$

“time periods”. Each user is assumed to have some unique identity,  $uid$ . Tsang *et al.* [15,14,9] suggest 24-hour windows, 5-minute periods, and IP address  $uids$ .

At the beginning of each linkability window  $d$ , the user connects directly to the PM to request a pseudonym  $\rho = F_{KP}(d, uid)$ ,  $\tau = MAC_{\kappa_{pn}}(\rho)$ . The user then connects anonymously to the NM, sending  $\rho, \tau$ ; if the tag  $\tau$  is correct, the NM forms a sequence of  $w + 1$  seeds  $s_0 = F_{KN}(\rho, d)$ ,  $s_i = f(s_{i-1})$ ; tokens  $t_i = g(s_i)$ ; and ciphertexts  $c_i = E_{KN}(t_0, s_i)$ . The NM gives the user *nymbles*  $\nu_i = (i, t_i, c_i, MAC_{\kappa_{ns}}(i, t_i, c_i))$ . Then at the  $i$ -th time period, the user connects anonymously to the SP, checks that  $t_0$  is not blacklisted, and provides  $\nu_i$ ; the SP grants access if the MAC tag is correct and  $t_i$  is not blacklisted. To complain, the SP sends  $\nu_i$  to the NM, who decrypts  $c_i$  to get  $t_0, s_i$ , and computes  $t_{i+1}, \dots, t_w$  and sends these and the “canonical nymble”  $t_0$  to the SP to add to the blacklist.

**Collusion of TTPs.** There are four possible collusive scenarios between a PM, NM and SP. First, the PM and NM can collude to learn which users connect to which SPs. Second, the NM and SP can collude to link all of a user’s actions within a single linkability window. Third, the PM, NM, and SP can all collude together to deanonymize all of the user’s activities, across linkability windows. The final scenario, involving the PM and SP, is not a privacy threat in Nymble.

**Nymbler.** In Nymbler [7], the PM is replaced by a Credential Manager (CM), who issues an anonymous credential on a secret  $x_{uid}$  to each user. The user then uses this credential to create his own series of seeds and tokens, with  $s_0 = h^{x_{uid}}$  using  $f(x) = x^2 \bmod n$ , and  $g(x) = \gamma^x$  over a trapdoor discrete logarithm group chosen by the NM. The user obtains blind signatures  $\sigma_1, \dots, \sigma_w$  on the tokens  $t_1, \dots, t_w$  from the NM, using efficient zero-knowledge proofs to show that they are correctly formed. The SP, on receiving  $\nu_i = (t_i, \sigma_i)$  can check the signature, and the NM can extract a seed from  $t_i = \gamma^{s_i}$  by computing the discrete logarithm (a costly but feasible computation using the trapdoor). The use of blind signatures prevents the NM and CM from colluding to link users to SPs; the use of anonymous credentials prevents the NM, CM, and SP from colluding to de-anonymize users.<sup>1</sup>

**Jack.** Jack [10] follows Nymbler in replacing the PM with a CM that issues credentials on a secret  $x_{uid}$ . The user creates her own nymbles by encrypting a pseudonym  $h^{x_{uid}}$  under the NM’s public key; the SP maintains a cryptographic accumulator of blacklisted pseudonyms. When the user connects to the SP, she presents her encrypted pseudonym along with a proof of correctness — the pseudonym corresponds to the  $x_{uid}$  in her credential, is encrypted correctly, and is not in the accumulator. To block a user, the NM decrypts the pseudonym and the SP adds it to the accumulator. As in Nymbler, the use of anonymous credentials prevents deanonymization or linking across linkability windows, and since the user creates nymbles noninteractively, the NM and CM cannot collaborate to link users to SPs.

---

<sup>1</sup> We note that [7] discuss generating  $x_{uid}$  so that it is not secret to the CM. In this case the CM and SP can collude to deanonymize users, so [6] suggests distributing the CM so that collusion between at least  $k$  CM agents and the SP is required.

**Blind Signatures.** BNymble uses Chaum’s blind signature scheme [3]. In this scheme, the signer has public key  $N$ , an RSA modulus, and secret key  $d = 3^{-1} \bmod \phi(N)$ . We utilize a cryptographic hash function  $H : \mathcal{M} \rightarrow \mathbb{Z}_N^*$ , modeled as a random oracle. When a user wishes to obtain a blinded signature on the message  $x \in \mathcal{M}$ , she picks  $r \in_R \mathbb{Z}_N^*$ , and hands  $\beta = H(x)r^3 \bmod N$  to the signer, who returns  $\zeta = \beta^{1/3} \bmod N = H(x)^{1/3}r \bmod N$ . Finally, the user computes  $\sigma = \zeta/r = H(x)^{1/3} \bmod N$ . It is easy to see that signing transcripts  $(\beta, \zeta)$  are information-theoretically unlinkable to the signatures  $(x, H(x)^{1/3} \bmod N)$ ; Bellare *et al.* [1] prove that it is infeasible to create  $n + 1$  valid signatures from  $n$  queries under the *one-more RSA inversion problem*.

### 3 BNymble Protocol

**Overview.** In BNymble, we modify the User Registration protocol and the Nymble Acquisition protocol. In each linkability window, a user Alice first connects directly to the PM and demonstrates control over her IP address or other limited resource. She also chooses a random “blind nym” (bnym) and blinds it for signing. The PM records her *uid* (IP address) and if a signature has not already been issued for that *uid* in that linkability window, the PM signs and returns her bnym. Alice then unblinds her bnym. In the nymble acquisition phase, she opens an anonymous connection to the NM and presents her signed bnym. If the signature is valid, the NM computes seed  $s_0 = F_{KN}(\text{bnym})$  and proceeds as before. We now describe this procedure in more detail.

**System Setup.** In addition to the setup in Nymble, at the beginning of each linkability window  $i$  the PM chooses an RSA modulus  $N$  for signing bnyms and transmits  $(i, N)$  to the NM via an authenticated channel. For each linkability window, the PM clears the set of used IP addresses. The system includes a cryptographic hash function  $H : \mathcal{M} \rightarrow \mathbb{Z}_N^*$ , modeled as a random oracle.<sup>2</sup>

**User Registration.** Alice obtains a blind nym as follows:

1. Alice downloads the PM’s public key for the current linkability window,  $N$ , and prepares a bnym for signing by choosing a random message  $x \in_R \mathcal{M}$  and a blinding factor  $r \in_R \mathbb{Z}_N^*$  and then computing  $\beta = H(x)r^3 \bmod N$ .
2. Alice connects directly to the PM and transmits  $\beta$  for signing. The PM verifies that her IP address has not previously been used this window, and then responds with  $\zeta = \beta^{1/3} \bmod N = H(x)^{1/3}r \bmod N$ .
3. Alice unblinds the signature by computing  $\sigma = \zeta/r = H(x)^{1/3} \bmod N$ .

**Credential Acquisition.** Alice obtains nymbles for window  $d$  as follows:

1. Alice connects anonymously to the NM and presents her bnym  $(x, \sigma = H(x)^{1/3} \bmod N)$ .

---

<sup>2</sup> Note that if  $N$  is a  $\lambda$ -bit RSA modulus, and  $H' : \mathcal{M} \rightarrow \{0, 1\}^{k+\lambda}$  is a random oracle, then  $H(x) = H'(x) \bmod N$  will be  $O(2^{-k})$ -statistically close to the required oracle.

2. The NM verifies that  $\sigma = H(x)^{1/3} \bmod N$ . The NM computes the sequence of  $w + 1$  seeds  $s_0 = F_{KN}(x, \sigma, d)$ ,  $s_i = f(s_{i-1})$ ; tokens  $t_i = g(s_i)$ ; and ciphertexts  $c_i = E_{KN}(t_0, s_i)$ .
3. The NM gives the user nymbles  $\nu_i = (i, t_i, c_i, MAC_{\kappa ns}(i, t_i, c_i))$ .

The remaining Nymble protocols are identical to those described in [14].

## 4 Evaluation

### 4.1 Security Analysis

BNymble preserves Nymble's security properties: *Blacklistability*, *Rate-limiting*, *Non-frameability* and *Anonymity*, assuming the *one-more RSA inversion problem* [1] is computationally intractable.

**Blacklistability.** An honest Pseudonym Manager will only issue one bnym per user. Thus for a coalition of  $c$  users to authenticate after all have been blacklisted, they would either have to forge a bnym, violating the assumed intractability of the *one-more RSA inversion problem*, or they would have to break blacklistability using only  $c$  pseudonyms, violating the blacklistability of Nymble.

**Non-Frameability.** Since distinct users have distinct *uids*, an honest PM will only refuse to grant a bnym to a user if that user has already received a bnym in that linkability window. Also, an honest NM will grant a different set of nymbles to each bnym. Thus there is no way for one user to frame another without violating the non-frameability of Nymble.

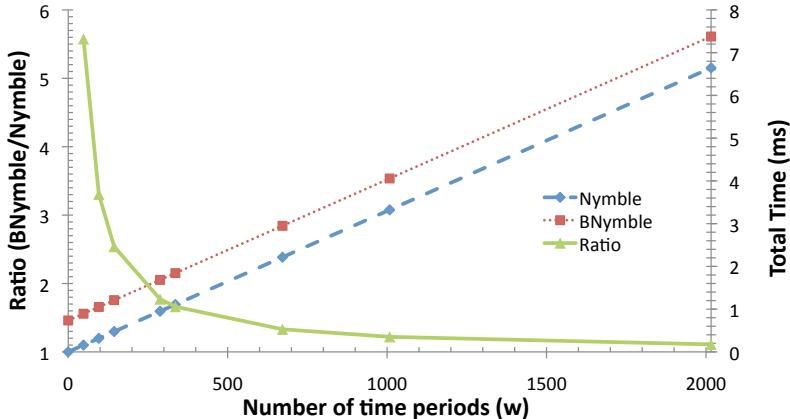
**Anonymity.** Anonymity in [9,15,14] is defined with respect to SPs only (that is, assuming non-colluding PM and NM). It is easy to see that since the nymbles in BNymble are generated according to the same process, the same property holds. We also can define anonymity in a much stronger sense: let the adversary control the PM, NM, and SP, and choose two users  $U$  and  $V$ . We allow the adversary to ask each user to register and acquire nymbles for any linkability window and any SP of the adversary's choosing, for any number  $k$  of window/SP pairs. The adversary then specifies a single, new linkability window;  $U$  and  $V$  execute the user registration protocol (with the adversary), and then execute the credential acquisition protocol in a random ordering. The adversary wins if he can guess whether  $U$  or  $V$  acquired nymbles first. The protocol is *anonymous* if no adversary can win with probability non-negligibly greater than  $1/2$ . (Notice that since the adversary sees the nymbles issued, this implies that for any time period, the nymbles themselves are also indistinguishable.) Because bnyms are information-theoretically independent of both *uids* and bnyms from other windows, every adversary wins this game with probability exactly  $1/2$ .

### 4.2 Efficiency

In order to compare the cost of the various TTP-based anonymous blacklisting systems, we measured the costs of the basic cryptographic operations required of the users, NM, PM, and SP in each of the systems. Table 1 shows these costs.

**Table 1.** Cost of cryptographic operations in each “nymble-like” anonymous black-listing system. Nymble acquisition and verification costs are per nymble. All times measured on a 2.67GHz quad-core Xeon W3520 with 12GB RAM.

	Nymble	BNymble	Jack	Nymbler
User Registration (ms)	0.0008	0.70	9.12	9.12
Nymble acquisition (ms)	0.0027	0.0027	264	649
Nymble verification (ms)	0.0006	0.0006	208	0.0011



**Fig. 1.** Total cryptographic cost of user registration, nymble acquisition and nymble verification as a function of number of time periods per linkability window. With one week linkability windows and 5-minute time periods, the total cost of BNymble is only 11% higher than Nymble.

User registration in BNymble is obviously the most expensive phase, but it is also the least executed protocol - occurring once per linkability window. Figure 1 shows how this one-time cost compares to the total cost of authentication for various linkability window sizes. At  $w = 288$ , as suggested by [9], The total cost of authentication in BNymble is less than a factor of 2 greater than Nymble, compared to 5 orders of magnitude from Nymbler and Jack. Longer linkability windows decrease this difference further - with 5-minute time periods and a one-week linkability window ( $w = 2016$ ), the difference is only 11%.

## 5 Extensions and Future Work

**Coin Recovery.** In Nymble, a user’s nymbles for a given linkability window are a deterministic function of his IP address and the four secret keys. This means that if a user loses his nymbles, or another user with the same IP address wishes to authenticate anonymously, he can repeat the user registration and nymble acquisition protocols and get the same chain a second time. Because bnyms in BNymble are randomized and chosen by users, a literal implementation cannot support this feature. However, we can allow the user the option to choose his

bnyms and blinding factor pseudorandomly, based on the hardened cryptographic hash of a strong password and the index of the current linkability window. The PM would then be modified so that when a client with the same IP address requests a second bnym for the same linkability period, the blinded signature from the first request is returned, allowing the client to recover his bnym. (Since blinded signatures are information-theoretically unlinkable there is no privacy risk in doing so.)

We note that Nymble can also support “fate-sharing” of multiple users behind a Network Address Translator (NAT) based on the deterministic nature of its pseudonyms. We leave the extension of BNymble to handle this case as an important question for future work.

**Identity Logging.** We note that, in contrast to other “Nymble-like protocols,” the BNymble PM is required to keep a log of *uids* (IP addresses) to which a bnym has been issued for each linkability window. This is obviously undesirable. While traditional approaches to limiting the usefulness of this log can be applied,<sup>3</sup> these approaches do not help if the PM is compromised. Some protection can be obtained by introducing a *List Manager*, who computes an RSA key pair  $(M, d)$ . User registration then becomes a slightly longer interaction: the user first connects directly to the PM, and sends a blinded signature request. The PM responds with  $x = F_K(uid, d)$ . The user connects anonymously to the LM, sends  $x$  and receives  $y = x^{1/3} \bmod M$ , and sends  $y$  to the PM. The PM checks that  $y^3 = F_K(uid, d)$ , and if it is, verifies that  $\text{hash}(y)$  is not in the log. If successful, the PM returns the blinded signature  $\zeta$  and adds  $\text{hash}(y)$  to the log.

**Extended Blacklisting.** We note that, using the previous scheme to store (protected) lists of active *uids* per linkability window, and using the first component of the *bnyms*,  $x \in \mathcal{M}$  as the “canonical nymble,” BNymble can support extended blacklisting using the same techniques in [6], except that when a *uid* was not present during a linkability window with a non-empty blacklist, we can have the *PM* issue a random bnym for the window without updating the log.

**Resisting Traffic Analysis.** One potential concern in BNymble is side channels based on timing information: the times of registration, nymble acquisition, and first use of a service are likely to be correlated. (We note that a somewhat similar problem exists in Nymbler: after the user obtains a credential for her IP, she (anonymously) contacts the NM and sends the value  $h$  corresponding to the SP she wishes to obtain service from.) To minimize the impact of this side channel, we recommend that users first entering the system compute a random delay  $\Delta$  and wait  $\Delta$  minutes after registration and before nymble acquisition. Additionally, the PM should allow users to obtain bnyms for linkability period  $d+1$  during the last half of linkability period  $w$ . Users that perform this advance registration will be indistinguishable and will help to provide cover traffic for the newly registered users.

**Acknowledgments.** We thank Roger Dingledine, Ian Goldberg, Ryan Henry, Apu Kapadia, and Zi Lin for helpful discussions about this paper. This work was

---

<sup>3</sup> For example, replace *uid* with  $F_K(uid)$  and discarding  $K$  after the linkability window

supported by NSF grants 0546162 and 0917145 and the University of Minnesota Undergraduate Research Opportunities Program.

## References

1. 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)
2. Brickell, E., Li, J.: Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In: *WPES 2007: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, pp. 21–30. ACM, New York (2007)
3. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. *Commun. ACM* 28(10), 1030–1044 (1985)
4. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: *SSYM 2004: Proceedings of the 13th Conference on USENIX Security Symposium*, pp. 21–21. USENIX Association, Berkeley (2004)
5. GmbH, J.: Jondonym: Private and secure web surfing (September 2010), <http://anonymous-proxy-servers.net/>
6. Henry, R., Goldberg, I.: Extending nymble-like systems. Tech. Rep. Technical Report CACR 2010-23, Unviersity of Waterloo (2010)
7. Henry, R., Henry, K., Goldberg, I.: Making a nymbler nymble using verbs. Tech. rep., University of Waterloo Technical Report CACR 2010-05 (2010)
8. Holt, J.E., Seamons, K.E.: Nym: Practical pseudonymity for anonymous networks. Tech. Rep. 4, BYU Internet Security Research Lab (2006)
9. Johnson, P.C., Kapadia, A., Tsang, P.P., Smith, S.W.: Nymble: Anonymous IP-Address Blocking. In: Borisov, N., Golle, P. (eds.) *PET 2007. LNCS*, vol. 4776, pp. 113–133. Springer, Heidelberg (2007)
10. Lin, Z., Hopper, N.: Jack: Scalable accumulator-based nymble system. In: *WPES 2010: Proceedings of the 9th ACM Workshop on Privacy in the Electronic Society*. ACM (2010)
11. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: Blacklistable anonymous credentials: blocking misbehaving users without ttps. In: *CCS 2007: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 72–81. ACM, New York (2007)
12. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: BLAC: Revoking Repeatedly Misbehaving Anonymous Users Without Relying on TTPs. Tech. rep., Dartmouth Computer Science TR2008-635 (2008)
13. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: Perea: Towards practical ttp-free revocation in anonymous authentication. In: *CCS 2008: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 333–344. ACM (2008)
14. Tsang, P.P., Kapadia, A., Cornelius, C., Smith, S.W.: Nymble: Blocking Misbehaving Users in Anonymizing Networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)* (September 2009)
15. Tsang, P.P., Kapadia, A., Cornelius, C., Smith, S.W.: Nymble: Blocking misbehaving users in anonymizing networks. Tech. rep., Dartmouth Computer Science TR2008-637 (2008)

# Towards Secure Bioinformatics Services

Martin Franz<sup>1</sup>, Björn Deiseroth<sup>1</sup>, Kay Hamacher<sup>2</sup>, Somesh Jha<sup>3</sup>,  
Stefan Katzenbeisser<sup>1</sup>, and Heike Schröder<sup>1</sup>

<sup>1</sup> Technische Universität Darmstadt, Security Engineering Group

<sup>2</sup> Technische Universität Darmstadt, Computational Biology Group

<sup>3</sup> University of Wisconsin, Computer Sciences Department

**Abstract.** In this paper we show how privacy of genomic sequences can be protected while they are analyzed using Hidden Markov Models (HMM), which is commonly done in bioinformatics to detect certain non-beneficial patterns in the genome. Besides offering strong privacy guarantees, our solution also allows protecting the intellectual property of the parties involved, which makes the solution viable for implementation of secure bioinformatics services. In particular, we show how two mutually mistrusting parties can obliviously run the forward algorithm in a setup where one party knows a HMM and another party knows a genomic string; while the parties learn whether the model fits the genome, they neither have to disclose the parameterization of the model nor the sequence to each other. Despite the huge number of arithmetic operations required to solve the problem, we experimentally show that HMMs with sizes of practical importance can obliviously be evaluated using computational resources typically found in medical laboratories. As a central technical contribution, we give improved protocols for secure and numerically stable computations on non-integer values.

## 1 Introduction

It is commonly believed that advances in and economies of scale of biological sequencing will allow to sequence human genomes at low cost in the foreseeable future, effectively paving the route for personalized medicine [19,20]. Genomic data will be used by healthcare providers to check for disease predispositions or drug intolerances of individual patients. It can be foreseen that a service-based industry will emerge, where special providers offer services to match genomic sequences against models of specific diseases, as well as offering personalized programs of drug application to such patients.

In this context two important security problems arise: First, genomic data must be considered extremely privacy sensitive and should thus be strongly protected from abuse. Second, mathematical models for diseases and the related genomic details are valuable intellectual property of the service provider, which is the basis of his business model. Enabling such bioinformatics services thus requires security mechanisms that can achieve both goals at the same time: privacy protection of genomic data and protection of the involved intellectual property.

So far, existing works considered basic bioinformatics algorithms that search for patterns (represented by regular expressions or substrings), perform sequence alignments or compute typical quantities, such as the edit distance (e.g. see [3,12,18]). In this paper

we show how to evaluate much more complex probabilistic queries represented by Hidden Markov Models (HMMs) on genomic data in a secure way. HMMs are widely used in computational biology to detect locations with certain properties, e.g., disease related signals or general properties of important, disease related enzymes, such as kinases.

In particular we consider the following scenario: Party  $\mathcal{A}$ , who is a health care provider acting on behalf of a patient, has sequenced the patient's genome.  $\mathcal{A}$  wants to interact with a provider  $\mathcal{B}$ , who offers a service to check parts of the genome against a model, encoded as HMM, for a specific disease.  $\mathcal{A}$  wants to see "how good" the model fits the genome, thereby determining the likelihood of a disease predisposition, while being bound to preserve the patient's privacy. At the same time, party  $\mathcal{B}$  does not want to disclose the HMM since the model itself is his business secret that distinguishes his service from other providers. In this paper we show how  $\mathcal{A}$  and  $\mathcal{B}$  can achieve both goals by providing a way to run the HMM *forward algorithm* [6] in an oblivious manner.

This requires techniques for efficiently and accurately performing Secure Multiparty Computation (SMC) on real values, since the forward algorithm operates on (sometimes extremely small) probabilities. A practical SMC-solution was first proposed in [9], which encodes (heavily quantized) real values in a logarithmic representation and provides protocols to obliviously perform all basic arithmetic operations on encryptions of such values. As a contribution of this paper we significantly improve performance of the primitives in [9] so that several hundred thousand of such arithmetic operations can be performed within a few minutes.

In summary, we make the following contributions in this paper:

- We improve the most complex operation (which is the addition) of the framework presented in [9]; while the original solution required  $\mathcal{O}(|\mathcal{T}|)$  computation, our improved version requires only  $\mathcal{O}(\sqrt{|\mathcal{T}|})$  work.
- We further give a private implementation of the forward algorithm which is commonly used to process and analyze genomic material and protein sequences. The resulting protocol allows to protect both the privacy of genomic data and the intellectual property of the service provider (i.e., the HMM).
- We implemented the algorithm in the widely used HMMER [7] framework and tested it on realistic models and sequences. We show that the developed protocols allow to analyze medium-size models on standard computing equipment in a couple of minutes, despite performing about 300.000 arithmetic operations on small encrypted probabilities.

## 2 Related Work

Several constructions for Secure Multiparty Computation [21] are known in the literature. Various approaches for securely evaluating a function have been developed for different function representations, namely combinatorial circuits [10,11], Ordered Binary Decision Diagrams [13], branching programs [15], or one-dimensional look-up tables [14]. While these methods target computations performed over the integers only, extensions were proposed that can handle rational numbers [8], real values in fixed point notation [4] and real values in logarithmic notation [9].

Some works deal with the secure analysis of genomic sequences: [3] considers the problem of securely computing the edit distance between two strings; [12] presents a

secure algorithm to perform sequence alignments using the Smith-Waterman algorithm; finally [18] proposes an algorithm that allows to run arbitrary queries, formulated as regular expressions, on genomic sequences in an oblivious way. Some papers deal with the problem of securely evaluating HMMs [17,16]. However, neither of these protocols comes with a satisfactory security proof and sufficient experimental analysis to assure the accuracy of the results.

### 3 Efficient Computations on Encrypted Non-integer Values

As a central cryptographic tool, we use a semantically secure additively homomorphic public-key encryption scheme introduced by Damgård, Geisler and Krøigaard (DGK) in [5]. In DGK a message  $m \in \mathbb{Z}_u$  is encrypted by computing  $c = g^m h^r \bmod n$ , where  $n$  is a RSA-modulus,  $u$  is a prime number and  $r$  is a randomly chosen integer. In our application  $u$  is from a very small range, which results in a very small plaintext space  $\mathbb{Z}_u$ . For our construction it will be essential that the plaintext space is a small finite field, since this will allow us to perform very efficient operations (such as polynomial interpolation in  $\mathbb{Z}_u$  or efficient decryption). Thus we will assume that the prime number  $u$  is chosen only slightly larger than the values which occur during the computations (for typical applications values  $u$  with bitlength 10-20 bits will be sufficient). In the sequel we will denote a DGK encrypted value  $m$  by  $[m]$ .

*Secure computations on non-integer values.* In [9], a framework was presented which allows secure computations on non-integer numbers. Rather than using a fixed point representation, the values are approximated using a logarithmic representation. This promises a constant relative representation error both for very small and for very large values. Each non-integer value  $v \in R \subset \mathbb{R}$  is represented as a triplet  $(\rho, \sigma, \tau)$ , where  $\rho$  is a flag indicating whether  $v$  is equal to zero,  $\sigma$  stores the sign of  $v$  and  $\tau = \lceil -S \cdot \log_B(\frac{|v|}{C}) \rceil$  for positive parameters  $B, C, S$ . The framework provides protocols **LSUM**, **LSUB**, **LDIV**, and **LPROD**, which allow two parties to obliviously perform the four basic arithmetic operations on triplets  $([\rho], [\sigma], [\tau])$  which have been encrypted using a semantically secure homomorphic cryptosystem. It can easily be seen that a product (**LPROD**) of such encrypted numbers  $x, y$  can be computed in a straightforward manner using the homomorphic properties of the encryption, by observing that  $\tau_{xy} = \tau_x + \tau_y$ . Unfortunately, the operation **LSUM** is more involved. To this end, the parties have to compute  $\tau_{x+y} = \tau_y - S \cdot \lceil \log_B(1 + B^{(\tau_x - \tau_y)/S}) \rceil$ . Computing  $\tau_{x+y}$  from  $\tau_x$  and  $\tau_y$  using standard methods from Secure Multiparty Computation is inefficient. Therefore, [9] uses an oblivious table lookup to obtain the value  $\tau_{x+y}$ , so that none of the parties is able to learn anything about the value which was queried, and in turn, which value was retrieved by the oblivious look-up operation.

In the remainder of this section we will show how this table look-up operation can significantly be improved in order to obtain a more efficient **LSUM** implementation. In particular, we describe a two-party protocol which allows to perform an oblivious table lookup for some value  $x \in X$ , where  $X = [x_l; x_u]$  is some interval with bounds  $x_l, x_u \in \mathbb{Z}$ . The table will be denoted by  $\mathcal{T} = (x_i, f(x_i))_{x_i \in X}$ ; we use  $\mathcal{T}(x)$  to denote the entry of the table  $\mathcal{T}$  at position  $x$ . The construction is given in the well known two party scenario, where one party  $\mathcal{A}$  holds the private key for some homomorphic

**Table 1.** Table lookup in [9]

$x$	$f(x)$
$x_1$	$f(x_1)$
$x_2$	$f(x_2)$
$\vdots$	$\vdots$
$x_{ \mathcal{T} }$	$f(x_{ \mathcal{T} })$

**Table 2.** New table lookup

$y \setminus z$	0	1	$\dots$	$k - 1$
0	$f(x_1)$	$f(x_2)$	$\dots$	$f(x_k)$
1	$f(x_{k+1})$	$f(x_{k+2})$	$\dots$	$f(x_{2k})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k - 1$	$f(x_{ \mathcal{T} -k+1})$	$f(x_{ \mathcal{T} -k+2})$	$\dots$	$f(x_{ \mathcal{T} })$

encryption scheme, and another party  $\mathcal{B}$  holds an encryption  $[x]$  of a value  $x$  and learns an encryption  $[\mathcal{T}(x)]$  of  $\mathcal{T}(x)$ , but neither  $x$  nor the plain table entry  $\mathcal{T}(x)$ .

*Efficient Private Function Evaluation.* Creating and transmitting the full table used in a **LSUM** is rather costly. Therefore, the main idea is to transform the lookup in a large table of size  $|\mathcal{T}|$  into two lookups in smaller tables of size  $k := \sqrt{|\mathcal{T}|}$  and one evaluation of a polynomial. This is, from a setting as depicted in Table 1, we go to a setting as depicted in Table 2. In the remainder of this section we assume that  $x$  is a positive value (this can always be achieved by shifting the interval  $X = [x_l; x_u]$  to  $X' = [0; x_u - x_l]$  and changing Table 1 accordingly). For simplicity, we will further assume that the table  $\mathcal{T}$  has  $2^{2\ell}$  entries, thus  $k = \sqrt{|\mathcal{T}|} = 2^\ell$ .

We first split up  $x$  into two values  $y$  and  $z$  which consist of the  $\ell/2$  most significant resp. least significant bits of  $x$ , i.e.  $x = y \cdot k + z$ . Now, we perform two simultaneous table lookups. In the first lookup, the value  $z$  is mapped to a value  $\tilde{x} \in \{\tilde{x}_0, \dots, \tilde{x}_{k-1}\}$ , where the values  $\tilde{x}_1, \dots, \tilde{x}_{k-1}$  are chosen at random from  $\mathbb{Z}_u$ . Next, polynomials  $P_i$  are generated in a way that on the evaluation points  $\tilde{x}_1, \dots, \tilde{x}_{k-1}$  the polynomials take on values  $f(x_i)$ , i.e.  $P_y(\tilde{x}_z) = f(y \cdot k + z)$ . In the second lookup, the value  $y$  is used to select the polynomial  $P_y$  which is finally evaluated on  $\tilde{x}$  to obtain the result  $f(x) = f(y \cdot k + z) = P_y(\tilde{x})$ .

Thus, the full protocol consists of the following steps:

1. (Offline): Prepare representation as polynomials: First we choose  $k$  random values  $\tilde{x}_0, \dots, \tilde{x}_{k-1}$ . Next, using Newton-interpolation, we compute  $k$  polynomials  $P_y(\tilde{x}_z) = f(y \cdot k + z)$  such that  $P_y(\tilde{x}_z) = f(y \cdot k + z)$ .
2. Extract the bits of  $[x]$  to obtain values  $[y], [z]$ , such that  $x = y \cdot k + z$ .
3. Transfer value a  $\tilde{x}$  and a polynomial  $P$  with tables of size  $k \approx \sqrt{|\mathcal{T}|}$ : For this we can use the table-lookup as presented in [9] or use oblivious transfer (OT). First, use an oblivious table look-up to obtain the value  $\tilde{x}_z$ ; next, run the protocol a second time to obtain the polynomial  $P_y$ .
4. Reconstruct the value  $[\mathcal{T}(x)]$  by evaluating the polynomial  $P_y$  on the value  $\tilde{x}_z$ .

For the full protocol, a detailed description of all steps and a security proof we refer the reader to the full version of this paper.

## 4 Secure Bioinformatics

In this section we describe how the computational framework presented in Section 3 can be applied to algorithms that securely analyze genomic and gene product related sequences by using Hidden Markov Models (HMM). HMMs are probabilistic methods that model a stream of symbols by a Markov process, which is driven by “hidden”

states of the process, unobservable to an outsider. In different states the dynamics of the process differs and thus the statistics of emitted symbols varies with time. States in bioinformatics applications can be indicators for particular disease related properties of sequences that code for e.g. non-beneficially mutated proteins or sites of potential post-translationally modifications.

*Hidden Markov Models.* A Hidden Markov Model  $\lambda = (A, B, \pi)$  is characterized by the following elements:

- Each HMM contains a set  $S$  of  $N$  hidden states:  $S = \{S_1, S_2, \dots, S_N\}$ .
- A set  $V$  of  $M$  distinct observation symbols per state (the output alphabet):  $V = \{v_1, v_2, \dots, v_M\}$ .
- The state transition probability matrix  $A = \{a_{ij}\}$ , where  $a_{ij}$  is the probability of moving from state  $S_i$  to state  $S_j$ :  $a_{ij} = \text{Prob}[q_{t+1} = S_j | q_t = S_i]$  for  $1 \leq i \leq N$  and  $1 \leq j \leq N$ , with proper normalization  $\forall_{1 \leq i \leq N} \sum_j a_{ij} = 1$ .
- The emission probabilities  $B = \{b_j(v_k)\}$  in state  $S_j$ , where  $b_j(v_k)$  is the probability of emitting symbol  $v_k$  at state  $S_j$ :  $b_j(k) = \text{Prob}[v_k \text{ at } t | q_t = S_j]$  for  $1 \leq j \leq N$  and  $1 \leq k \leq M$ .
- The initial state distribution  $\pi = \{\pi_i\}$ , where  $\pi_i$  is the probability that the start state is  $S_i$ :  $\pi_i = \text{Prob}[q_1 = S_i]$  for  $1 \leq i \leq N$ .

In typical bioinformatics applications (such as the one discussed in the introduction) a fundamental problem arises: Given a Hidden Markov Model  $\lambda$  and an observed sequence  $O = o_1 o_2 \dots o_T$ , compute the probability  $\text{Prob}[O | \lambda]$  that the HMM can generate the sequence. This value indicates the significance of the observation. This problem is commonly solved by applying the forward algorithm.

*Secure Forward Algorithm.* We consider the following two-party scenario. Party  $\mathcal{A}$  knows a genomic sequence  $O$ , while party  $\mathcal{B}$  commands over a specific HMM  $\lambda$ .  $\mathcal{A}$  could be a health care provider, who has sequenced a patient's genome  $O$ , but wishes to preserve the privacy of the patient.  $\mathcal{B}$  is a drug company or some bioinformatics institute, which wants to preserve its intellectual property contained within the parameterization of the HMM. Both parties are interested to learn how good the model fits to the genome  $O$  by running the forward algorithm, whereas neither party wants to share its input with each other. The overall probability reported by the algorithm can be, for example, the confidence of  $\mathcal{B}$  that  $\mathcal{A}$ 's patient develops a particular disease.

To compute the probability  $\text{Prob}[O | \lambda]$ , we can employ the forward algorithm. Consider the so-called forward variable  $\alpha_t(i) = \text{Prob}[o_1, \dots, o_t, q_t = S_i | \lambda]$  which indicates how likely it is to end up in state  $S_i$  after processing  $t$  steps, assuming that  $o_1, \dots, o_t$  have been emitted. For  $\alpha_1(i)$  we have  $\alpha_1(i) = \pi_i b_i(o_i)$ . Given  $\alpha_t(i)$ , the probabilities  $\alpha_{t+1}(i)$  can be computed inductively by  $\alpha_{t+1}(i) = b_i(o_{t+1}) \cdot \sum_{j=1}^N \alpha_t(j) \cdot a_{ji}$ . Finally, we have  $\text{Prob}[O | \lambda] = \sum_{i=1}^N \alpha_T(i)$ .

A full description of the realization of the forward algorithm using the framework of Section 3 can be found in Protocol 1. Note that in the protocol for clearness of presentation we omit to explicitly label encoded elements: all values  $a_{ij}, b_i(v_j), \pi_i$  and  $\alpha_i$  should be read as encoded values according to Section 3. In the initialization step, party  $\mathcal{A}$  provides party  $\mathcal{B}$  with an encrypted version of the sequence, where each symbol of  $O$  is encoded as a binary vector of length  $M$  (the position of the one in the vector encodes

**Protocol 1.** Secure Forward Algorithm

**Input:** Party  $\mathcal{A}$ : Sequence  $O = o_1 o_2 \dots o_T$   
 Party  $\mathcal{B}$ : HMM  $\lambda = (A, B, \pi)$

**Output:**  $\text{Prob}[O | \lambda]$

1: Initialization:

Party  $\mathcal{A}$ :

For each  $o_i$  prepare a vector  $\Theta_i = \{\theta_{i1}, \dots, \theta_{iM}\}$  in a way that  $\theta_{ij} = 1$  if  $v_j = o_i$  and  $\theta_{ij} = 0$  otherwise.

Encrypt  $\Theta_i$  component wise and send  $[\Theta_i]$  to party  $\mathcal{B}$

2: Party  $\mathcal{B}$ :

Compute emission probabilities:

**for**  $i = 1$  to  $N$

**for**  $j = 1$  to  $T$

$$[\rho_{b_i(o_j)}] = \prod_{k=1}^M [\theta_{ik}]^{\rho_{b_i(v_k)}}$$

$$[\tau_{b_i(o_j)}] = \prod_{j=k}^M [\theta_{ik}]^{\tau_{b_i(v_k)}}$$

$$[b_i(o_j)] := ([\rho_{b_i(o_j)}], [\tau_{b_i(o_j)}])$$

**end**

**end**

3: Party  $\mathcal{B}$ :

**for**  $i = 1$  to  $N$   
 $[\alpha_1(i)] = \text{LPROD}([\pi_i], [b_i(o_1)])$   
**end**

4: Induction:  
**for**  $t = 1$  to  $T - 1$   
**for**  $i = 1$  to  $N$   
 $[\Sigma_1^1] = \text{LPROD}([\alpha_t(1)], [a_{1i}])$   
**for**  $j = 2$  to  $N$   
 $tmp = \text{LPROD}([\alpha_t(j)], [a_{ji}])$   
 $[\Sigma_1^j] = \text{LSUM}([\Sigma_1^{j-1}],)$   
**end**  
 $[\alpha_{t+1}(i)] = \text{LPROD}([b_i(o_{t+1})], [\Sigma_1^N])$   
**end**  
**end**

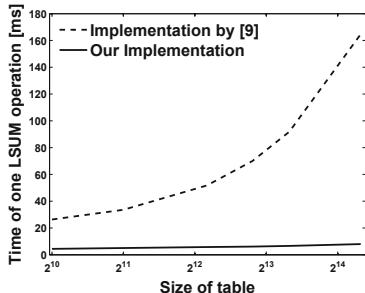
5: Termination:  
 $[\Sigma_1^2] = \text{LSUM}(\alpha_T(1), \alpha_T(2))$   
**for**  $i = 3$  to  $N$   
 $\text{LSUM}([\Sigma_1^{i-1}], [\alpha_T(i)])$   
**end**  
**return**  $[\text{Prob}[O | \lambda]] := [\Sigma_1^N]$

the symbol). This allows party  $\mathcal{B}$  to easily compute encryptions  $[b_i(o_j)]$  of the emission probabilities by using the homomorphic properties of the Paillier cryptosystem in step 2. In addition, party  $\mathcal{B}$  initializes the values  $\alpha_1(i)$  as the product of the probabilities  $\pi_i$  and the emission probabilities  $b_i(o_1)$  for the first observation symbol. In step 3 the parties compute interactively the forward-variables  $\alpha_t(i)$ , and in step 4 the result of the forward algorithm  $[\text{Prob}[O | \lambda]]$ , which can be decrypted by  $\mathcal{A}$ .

## 5 Implementation and Experimental Results

We have implemented the optimized framework for performing secure computations on non-integer values as well as the secure forward algorithm in C++ using the GNU GMP library version 5.0.1. Tests were run on a computer with a 2.1 GHz 8 core processor and 4GB of RAM running Ubuntu version 9.04. The parties  $\mathcal{A}$  and  $\mathcal{B}$  were implemented as two multi-threaded entities of the same program. Therefore our tests do not include network latency.

*Complexity of LSUM.* We have implemented the **LSUM** operation using the optimizations described in Section 3. We compare our results to those of [9]. Figure 1 depicts the time complexity of the protocol of [9] and our optimized version. Both programs were run for different table sizes  $|\mathcal{T}|$  in the range between  $|\mathcal{T}| = 1.000$  and  $|\mathcal{T}| = 20.000$ , the implementation by [9] was allowed to reuse tables 150 times while our construction reused each set of polynomials 10 times. To allow for a fair comparison, we restricted both implementations to run only on one core of the processor (thus, no parallelization was allowed in the tests). Figure 1 depicts the results: The  $y$ -axis shows the computational complexity in milliseconds (wall clock time) required to perform one **LSUM**, for



**Fig. 1.** Performance of the **LSUM** operation depending on the table size  $|\mathcal{T}|$

different table sizes ( $x$ -axis, logarithmic scale). While the solution of [9] (dotted line) grows linear in  $|\mathcal{T}|$  it can be seen that our improved solution grows moderately from 4.41ms for  $|\mathcal{T}| = 1000$  to 8.64ms for  $|\mathcal{T}| = 20.000$ .

*Complexity of private HMM analysis.* In order to demonstrate the practicality of the secure forward algorithm developed in Section 4), we implemented this algorithm in HMMER [1], version 2.3.2, which is widely used in the bioinformatics community to perform analysis of DNA and protein sequences. Real values were encoded and encrypted as described in Section 3, while the **LSUM** operation was realized by using our optimized construction.

We tested our implementation with several HMMs from the PFAM database [2]. Among them are models which are relevant for identification of protein domains, that play crucial roles in oncogenic transformations and other diseases connected to signaling in cells. In particular, we chose HMMs of three different sizes: A small model (SH3\_1, PF00018) of length 48, a medium model (Ras, PF00071) of length 162 and a large model (BID, PF06393) with 196 states. For the small, medium and large models we chose tables of size 1400, 3520 and 4600, respectively, in the **LSUM** operation. We experimentally chose the parameters of the number representation in a way that minimized the overall quantization error.

We measure the computational complexity of Protocols 1. The first row of Table 2 depicts the average runtime (wall clock time) of a single run of the forward algorithm. Note that in these tests we allowed parallel computations on multiple cores of the processor (e.g., we allowed to parallelize computation of the polynomials, run multiple **LSUM** operations in parallel, etc). Even though we did not fully optimize our programs at this point (only 3 out of 8 cores were used on average), we believe that the results are nevertheless insightful: For example, running the forward algorithm on the medium sized model requires approximately 8 minutes, despite performing 297.929 invocations of **LPROD** and 162.491 invocations of **LSUM**.

The communication complexity measures the traffic between the two parties. This value mainly depends on the size of the RSA modulus  $n$ , which was set to 1024 bits. The keysize for the garbled circuits was set to 80 bit. The second row of Table 2 depicts the communication complexity.

	<b>Small</b>	<b>Medium</b>	<b>Large</b>
Computation	33	499	632
Communication	69.6	964.3	1472.3
<b>LSUM</b>	14.161	162.491	225.984
<b>LPROD</b>	25.971	297.929	414.337

**Fig. 2.** Computational complexity (seconds), Communication complexity (MB), operation count for **LSUM** and **LPROD** of the forward algorithm for different model sizes

**Acknowledgements.** Kay Hamacher gratefully acknowledges financial support by Fonds der chemischen Industrie. This work was supported by the German Research Foundation (DFG) and the Center for Advanced Security Research Darmstadt (CASED).

## References

1. Hmmer, biosequence analysis using profile hidden markov models, <http://hmmer.wustl.edu/>
2. Pfam version 24.0, <http://pfam.sanger.ac.uk>
3. Atallah, M.J., Kerschbaum, F., Du, W.: Secure and private sequence comparisons. In: ACM Workshop on Privacy in the Electronic Society, pp. 39–44 (2003)
4. Catrina, O., Saxena, A.: Secure Computation with Fixed-Point Numbers. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 35–50. Springer, Heidelberg (2010)
5. Damgård, I.B., Geisler, M., Krøigaard, M.: Efficient and Secure Comparison for On-Line Auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007)
6. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological Sequence Analysis – Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press (1998)
7. Eddy, S.R.: Profile hidden markov models. Bioinformatics 14(9), 755–763 (1998)
8. Fouque, P.-A., Stern, J., Wackers, J.-G.: Cryptocomputing with Rational. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 136–146. Springer, Heidelberg (2003)
9. Franz, M., Deisereth, B., Hamacher, K., Katzenbeisser, S., Jha, S., Schroeder, H.: Secure computations on real-valued signals. In: Proceedings of the IEEE Workshop on Information Forensics and Security - WIFS 2010 (2010)
10. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In: ACM Symposium on Theory of Computing — STOC 1987, May 25–27, pp. 218–229. ACM (1987)
11. Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation via Ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
12. Jha, S., Kruger, L., Shmatikov, V.: Towards practical privacy for genomic computation. In: IEEE Symposium on Security and Privacy, pp. 216–230 (2008)
13. Kruger, L., Jha, S., Goh, E.-J., Boneh, D.: Secure function evaluation with ordered binary decision diagrams. In: ACM CCS, pp. 410–420 (2006)
14. Naor, M., Nissim, K.: Communication complexity and secure function evaluation. Electronic Colloquium on Computational Complexity (ECCC) 8(062) (2001)
15. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: ACM Symposium on Theory of Computing, pp. 590–599 (2001)
16. Polat, H., Du, W., Renkes, S., Oysal, Y.: Private predictions on hidden markov models. Artif. Intell. Rev. 34(1), 53–72 (2010)
17. Smaragdis, P., Shashanka, M.: A framwork for secure speech recognition. IEEE Transactions on Audio, Speech and Language Processing 15(4), 1404–1413 (2007)
18. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient DNA searching through oblivious automata. In: ACM CCS, pp. 519–528 (2007)
19. van 't Veer, L.J., Bernards, R.: Enabling personalized cancer medicine through analysis of gene-expression patterns. Nature 452(7187), 564–570 (2008)
20. West, M., Ginsburg, G.S., Huang, A.T., Nevins, J.R.: Embracing the complexity of genomic data for personalized medicine. Genome Research 16(5), 559–566 (2006)
21. Yao, A.C.-C.: Protocols for Secure Computations (Extended Abstract). In: Annual Symposium on Foundations of Computer Science — FOCS 1982, pp. 160–164. IEEE (1982)

# Quo Vadis? A Study of the Evolution of Input Validation Vulnerabilities in Web Applications

Theodoor Scholte<sup>1</sup>, Davide Balzarotti<sup>2</sup>, and Engin Kirda<sup>2,3</sup>

<sup>1</sup> SAP Research, Sophia Antipolis  
theodoor.scholte@sap.com

<sup>2</sup> Institut Eurecom, Sophia Antipolis

{balzarotti,kirda}@eurecom.fr

<sup>3</sup> Northeastern University, Boston

**Abstract.** Web applications have become important services in our daily lives. Millions of users use web applications to obtain information, perform financial transactions, have fun, socialize, and communicate. Unfortunately, web applications are also frequently targeted by attackers. Recent data from SANS institute estimates that up to 60% of Internet attacks target web applications.

In this paper, we perform an empirical analysis of a large number of web vulnerability reports with the aim of understanding how input validation flaws have evolved in the last decade. In particular, we are interested in finding out if developers are more aware of web security problems today than they used to be in the past. Our results suggest that the complexity of the attacks have not changed significantly and that many web problems are still simple in nature. Hence, despite awareness programs provided by organizations such as MITRE, SANS Institute and OWASP, application developers seem to be either not aware of these classes of vulnerabilities, or unable to implement effective countermeasures. Therefore, we believe that there is a growing need for languages and application platforms that attack the root of the problem and secure applications by design.

## 1 Introduction

The web has become part of everyone’s daily life, and web applications now support us in many of our daily activities. Unfortunately, web applications are prone to various classes of vulnerabilities. Hence, much effort has been spent on making web applications more secure in the past decade (e.g., [4][15][28]).

Organizations such as MITRE [15], SANS Institute [4] and OWASP [28] have emphasized the importance of improving the security education and awareness among programmers, software customers, software managers and Chief Information Officers. These organizations do this by means of regularly publishing lists with the most common programming errors. Also, the security research community has worked on tools and techniques to improve the security of web applications. These techniques include static code analysis [9,14,33,34,35], dynamic

tainting [23,24,27], combination of dynamic tainting and static analysis [32], prevention by construction or by design [8,13,29,36] and enforcement mechanisms executing within the browser [1,7,10,31]. Some of these techniques have been commercialized and can be found in today’s development toolsets. An example is Microsoft’s FxCop [6] which can be integrated into some editions of Microsoft Visual Studio.

Although a considerable amount of effort has been spent by many different stake-holders on making web applications more secure, we lack quantitative evidence that this attention has improved the security of web applications over time. In particular, we are interested in finding out and understanding how two common classes of vulnerabilities, namely SQL injection and Cross Site Scripting, have evolved in the last decade.

We chose to focus our study on SQL Injection and Cross-Site Scripting vulnerabilities as these classes of web application vulnerabilities have the same root cause: improper sanitization of user-supplied input that result from invalid assumptions made by the developer on the input of the application. Moreover, these classes of vulnerabilities are prevalent, well-known and have been well-studied in the past decade. Thus, it is likely that there is a sufficient number of vulnerability reports available to allow an empirical analysis.

In this paper, by performing an automated analysis, we attempt to answer the following questions:

1. *Do attacks become more sophisticated over time?*

We automatically analyzed over 2600 vulnerabilities and found out that the vast majority of them was not associated to any sophisticated attack techniques. Our results suggest that the exploits do not intend to evade any input validation, escaping or encoding defense mechanisms. Moreover, we do not observe any particular increasing trend with respect to complexity.

2. *Do well-known and popular applications become less vulnerable over time?*

Our results show that an increasing number of applications have exactly one vulnerability. Furthermore, we observe a shift from popular applications to non-popular applications with respect to SQL Injection vulnerabilities, a trend that is, unfortunately, not true for Cross-Site Scripting.

3. *Do the most affected applications become more secure over time?*

We studied in detail the ten most affected open source applications resulting in two top ten lists – one for Cross-Site Scripting and one for SQL Injection. In total, 199 vulnerabilities were associated with these applications. We investigated the difference between *foundational* and *non foundational* vulnerabilities and found that the first class is decreasing over time. Moreover, an average time of 4.33 years between the initial software release and the vulnerability disclosure date suggests that many of today’s reported Cross-Site Scripting vulnerabilities were actually introduced into the applications many years ago.

The rest of the paper is organized as follows: The next section describes our methodology and data gathering technique. Section 3 presents an analysis of the SQL Injection and Cross-Site Scripting reports and their associated exploits. In Section 4, we present the related work and then briefly conclude the paper in Section 5.

## 2 Methodology

To be able to answer how Cross Site Scripting and SQL Injections have evolved over time, it is necessary to have access to significant amounts of vulnerability data. Hence, we had to collect and classify a large number of vulnerability reports. Furthermore, automated processing is needed to be able to extract the exploit descriptions from the reports. In the next sections, we explain the process we applied to collect and classify vulnerability reports and exploit descriptions.

### 2.1 Data Gathering

One major source of information for security vulnerabilities is the CVE dataset, which is hosted by MITRE [19]. According to MITRE's FAQ [21], CVE is not a vulnerability database but a vulnerability identification system that ‘aims to provide common names for publicly known problems’ such that it allows ‘vulnerability databases and other capabilities to be linked together’. Each CVE entry has a unique CVE identifier, a status (‘entry’ or ‘candidate’), a general description, and a number of references to one or more external information sources of the vulnerability. These references include a source identifier and a well-defined identifier for searching on the source’s website. Vulnerability information is provided to MITRE in the form of *vulnerability submissions*. MITRE assigns a CVE identifier and a candidate status. After the CVE Editorial Board has reviewed the candidate entry, the entry may be assigned the ‘Accept’ status.

For our study, we used the CVE data from the National Vulnerability Database (NVD) [25] which is provided by the National Institute of Standards and Technology (NIST). In addition to CVE data, the NVD database includes the following information:

- Vulnerability type according to the Common Weakness Enumeration (CWE) classification system [20].
- The name of the affected application, version numbers, and the vendor of the application represented by Common Platform Enumeration (CPE) identifiers [18].
- The impact and severity of the vulnerability according to the Common Vulnerability Scoring System (CVSS) standard [17].

The NIST publishes the NVD database as a set of XML files, in the form: `nvdcve-2.0-year.xml`, where year is a number from 2002 until 2010. The first file, `nvdcve-2.0-2002.xml` contains CVE entries from 1998 until 2002. In order to build timelines during the analysis, we need to know the discovery date, disclosure date, or the publishing date of a CVE entry. Since CVE entries originate

from different external sources, the timing information provided in the CVE and NVD data feeds proved to be insufficient. For this reason, we fetch this information by using the disclosure date from the corresponding entry in the Open Source Vulnerability Database (OSVDB) [11].

For each candidate and accepted CVE entry, we extracted and stored the identifier, the description, the disclosure date from OSVDB, the CWE vulnerability classification, the CVSS scoring, the affected vendor/product/version information, and the references to external sources. Then, we used the references of each CVE entry to retrieve the vulnerability information originating from the various external sources. We stored this website data along with the CVE information for further analysis.

## 2.2 Vulnerability Classification

Since our study focuses particularly on Cross-Site Scripting and SQL Injection vulnerabilities, it is essential to classify the vulnerability reports. As mentioned in the previous section, the CVE entries in the NVD database are classified according to the Common Weakness Enumeration classification system. CWE aims to be a dictionary of software weaknesses. NVD uses only a small subset of 19 CWEs for mapping CVEs to CWEs, among those are Cross-Site Scripting (CWE-79) and SQL Injection (CWE-89).

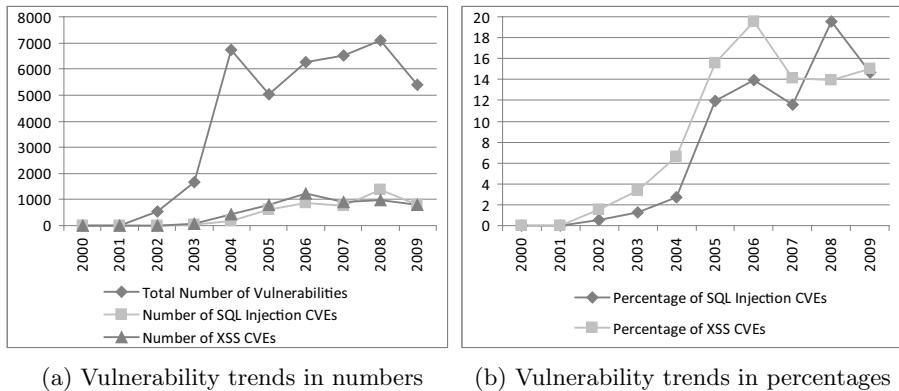
Although NVD provides a mapping between CVEs and CWEs, this mapping is not complete and many CVE entries do not have any classification at all. For this reason, we chose to perform a classification which is based on both the CWE classification and on the description of the CVE entry. In general, a CVE description is formatted according to the following pattern: {description of vulnerability} {location description of the vulnerability} *allows* {description of attacker} {impact description}. Thus, the CVE description includes the vulnerability type.

For fetching the Cross-Site Scripting related CVEs out of the CVE data, we selected the CVEs associated with CWE identifier ‘CWE-79’. Then, we added the CVEs having the text ‘Cross-Site Scripting’ in their description by performing a case-insensitive query. Similarly, we classified the SQL Injection related CVEs by using the CWE identifier ‘CWE-89’ and the keyword ‘SQL Injection’.

## 2.3 The Exploit Data Set

To acquire a general view on the security of web applications, we are not only interested in the vulnerability information, but also in the way each vulnerability can be exploited. Some external sources of CVEs that provide information concerning Cross-Site Scripting or SQL Injection-related vulnerabilities also provide exploit details. Often, this information is represented by a script or an *attack string*.

An attack string is a well-defined reference to a location in the vulnerable web application where code can be injected. The reference is often a complete URL that includes the name of the vulnerable script, the HTTP parameters, and some characters to represent the placeholders for the injected code. In addition



(a) Vulnerability trends in numbers

(b) Vulnerability trends in percentages

**Fig. 1.** Cross-Site Scripting and SQL Injection vulnerabilities over time

of using placeholders, sometimes, real examples of SQL or Javascript code may also be used. Two examples of attack strings are:

```
http://[victim]/index.php?act=delete&dir=&file=[XSS]
http://[victim]/index.php?module=subjects&func=viewpage&pageid=[SQL]
```

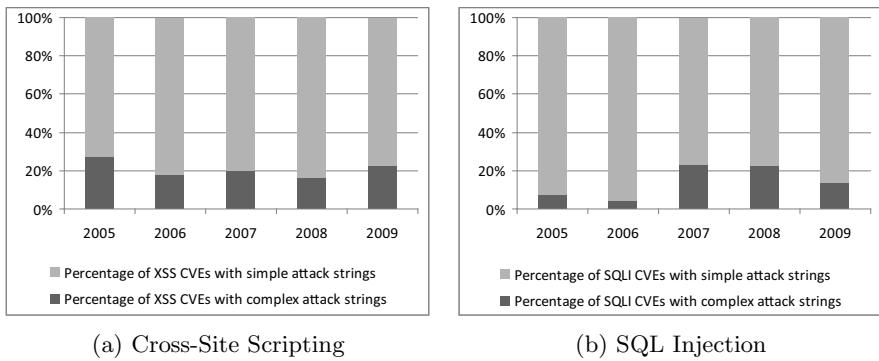
At the end of each line, note the placeholders that can be substituted with arbitrary code by the attacker.

The similar structure of attack strings allows our tool to automatically extract, store and analyze the exploit format. Hence, we extracted and stored all the attack strings associated with both Cross-Site Scripting and the SQL Injection CVEs.

### 3 Analysis of the Vulnerabilities Trends

The first question we wish to address in this paper is whether the number of SQL Injection and Cross-Site Scripting vulnerabilities reported in web applications has been decreasing in recent years. To answer this question, we automatically analyzed the 39,081 entries in the NVD database from 1998 to 2009. We had to exclude 1,301 CVE entries because they did not have a corresponding match in the OSVDB database and, as a consequence, did not have a disclosure date associated with them. For this reason, these CVE entries are not taken into account for the rest of our study. Of the remaining vulnerability reports, we identified a total of 5222 Cross-Site Scripting entries and 4810 SQL Injection entries.

Figure 1a shows the number of vulnerability reports over time and figure 1b shows the percentage of reported Cross-Site Scripting and SQL Injection vulnerabilities over the total CVE entries.

**Fig. 2.** Complexity of exploits over time

Our first expectation based on intuition was to observe the number of reported vulnerabilities follow a classical bell shape: beginning with a slow start when the vulnerabilities are still relatively unknown, then a steep increase corresponding to the period in which the attacks are disclosed and studied, and finally a decreasing phase when the developers start adopting the required countermeasures.

In fact, the graphs show an initial phase (2002-2004) with very few reports, followed by a steep increase of Cross-Site Scripting and SQL Injection vulnerability reports in the years 2004, 2005 and 2006. Note that this trend is consistent with historical developments. Web security started increasing in importance after 2004, and the first XSS-based worm was discovered in 2005 (i.e., “Samy Worm”). Hence, web security threats such as Cross-Site Scripting and SQL Injection started receiving more focus after 2004.

Unfortunately, the number of reported vulnerabilities has not significantly decreased since 2006. In other words, the number of vulnerabilities found in 2009 is comparable with the number reported in 2006. In the rest of this section, we will formulate and verify a number of hypotheses to explain the possible reasons behind this phenomenon.

### 3.1 Attack Sophistication

**Hypothesis 1.** *Simple, easy-to-find vulnerabilities have now been replaced by complex vulnerabilities that require more sophisticated attacks.*

The first hypothesis we wish to verify is whether the overall number of vulnerabilities is not decreasing because the simple vulnerabilities discovered in the early years have now been replaced by new ones that involve more complex attack scenarios. For example, the attacker may have to carefully craft the malicious input in order to reach a subtle vulnerable functionality, or to pass certain input transformations (e.g., uppercase or character replacement). In particular, we are interested in identifying those cases in which the application developers were aware of the threats, but implemented insufficient, easy to evade sanitization routines.

One way to determine the “complexity” of an exploit is to analyze the attack string, and to look for evidence of possible evasion techniques. As mentioned in Section 2.3, we automatically extracted the exploit code from the data provided by external vulnerability information sources. Sometimes, these external sources do not provide exploit information for every reported Cross-Site Scripting or SQL Injection vulnerability, do not provide exploit information in a parsable format, or do not provide any exploit information at all. As a consequence, not all CVE entries can be associated with an *attack string*. On the other hand, in some cases, there exist several ways of exploiting a vulnerability, and, therefore, more *attack strings* may be associated with a single vulnerability report. In our experiments, we collected attack strings for a total of 2632 distinct vulnerabilities.

To determine the exploit complexity, we looked at several characteristics that may indicate an attempt from the attacker to evade some form of input sanitization. The selection of the characteristics is inspired by so-called injection cheat sheets that are available on the Internet [16][30].

In particular, we classify a Cross-Site Scripting attack string as complex (i.e., in contrast to simple) if it contains one or more of the following characteristics:

- Different cases are used within the script tags (e.g., `ScRiPt`).
- The script-tags contains one or more spaces (e.g., `< script>`)
- The attack string contains ‘landingspace-code’ which is the set of attributes of HTML-tags (e.g., `onmouseover`, or `onclick`)
- The string contains encoded characters (e.g., `&#41;`)
- The string is split over multiple lines

For SQL Injection attack strings, we looked at the following characteristics:

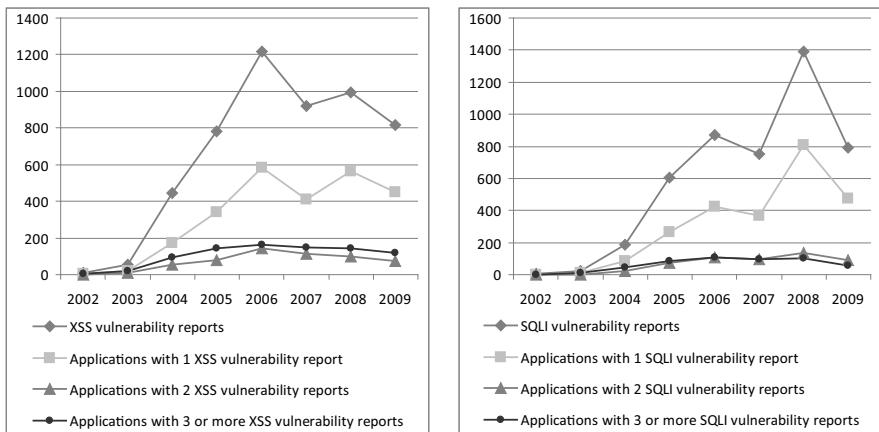
- The use of comment specifiers (e.g., `/**/`) to break a keyword
- The use of encoded single quotes (e.g., `'%27', '&#x27'; '&#39', 'Jw=='`)
- The use of encoded double quotes (e.g., `'%22', '&#x22;', '&#34', 'Ig=='`)

If none of the previous characteristics is present, we classify the exploit as “simple”. Figures 2a and 2b show the percentage of CVEs having one or more complex attack strings<sup>1</sup>. The graphs show that the majority of the available exploits are, according to our definition, not sophisticated. In fact, in most of the cases, the attacks were performed by injecting the simplest possible string, without requiring any tricks to evade input validation.

Interestingly, while we observe a slight increase in the number of SQL Injection vulnerabilities with sophisticated attack strings, we do not observe any significant increase of Cross-Site Scripting attack strings. This may be a first indication that developers are now adopting (unfortunately insufficient) defense mechanisms to prevent SQL Injection, but that they are still failing to sanitize the user input to prevent Cross-Site Scripting vulnerabilities.

---

<sup>1</sup> The graph starts from 2005 because there were less than 100 vulnerabilities having exploit samples available before that year. Hence, results before 2005 are statistically less significant.



**Fig. 3.** The number of affected applications over time

To conclude, the available empirical data suggests that an increased attack complexity *is not* the reason behind the steadily increasing number of vulnerability reports.

### 3.2 Application Popularity

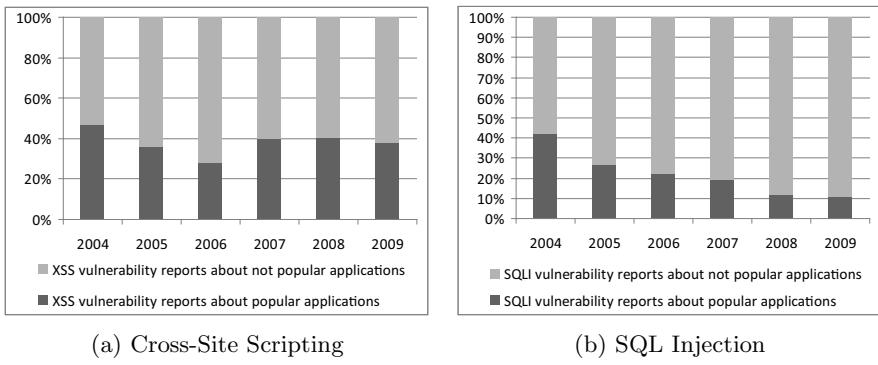
Since the complexity does not seem to explain the increasing number of reported vulnerabilities, we decided to focus on the type of applications. We started by extracting the vulnerable application's name from a total of 8854 SQL Injection and Cross-Site Scripting vulnerability reports in the NVD database that are associated to one or more CPE identifiers.

Figures 3a and 3b plot the number of applications that are affected by a certain number of vulnerabilities over time. Both graphs clearly show how the increase in the number of vulnerabilities is a direct consequence of the increasing number of vulnerable applications. In fact, the number of web applications with more than one vulnerability report over the whole time frame is quite low, and it has been slightly decreasing since 2006.

Based on this finding, we formulated our second hypothesis:

**Hypothesis 2.** *Popular applications are now more secure while new vulnerabilities are discovered in new, less popular, applications.*

The idea behind this hypothesis is to test whether more vulnerabilities were reported about well-known, popular applications in the past than they are today. That is, do vulnerability reports nowadays tend to concentrate on less popular, or recently developed applications?



**Fig. 4.** Vulnerability reports about applications and their popularity over time

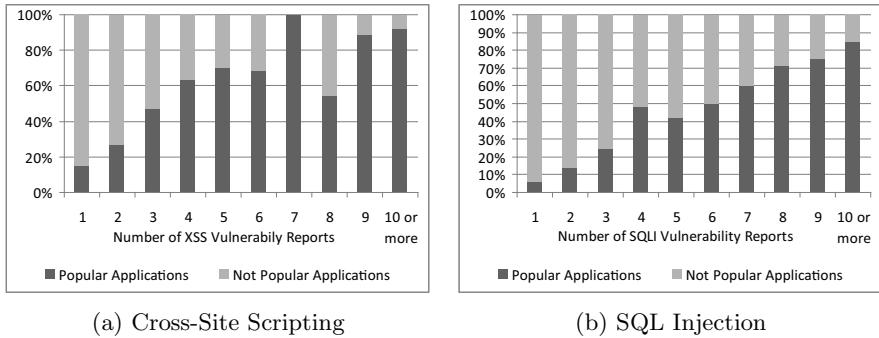
The first step consists of determining the popularity of these applications in order to be able to understand if it is true that popular products are more aware of (and therefore less vulnerable to) Cross-Site Scripting and SQL Injection attacks.

We determined the popularity of applications through the following process:

1. Using Google Search, we performed a search on the vendor and application names within the [Wikipedia](#) domain.
2. When one of the returned URLs contain the name of the vendor or the name of the application, we flag the application as being ‘popular’. Otherwise, the application is classified as being ‘unpopular’.
3. Finally, we manually double-checked the list of popular applications in order to make sure that the corresponding Wikipedia entries describe software products and not something else (e.g., when the product name also corresponds to a common English word).

After the classification, we were able to identify 676 popular and 2573 unpopular applications as being vulnerable to Cross-Site Scripting. For SQL Injection, we found 328 popular and 2693 unpopular vulnerable applications. Figure 4 shows the percentages of vulnerability reports that are associated with popular applications. The trends support the hypothesis that SQL Injection vulnerabilities are indeed moving toward less popular applications – maybe as a consequence of the fact that well-known product are more security-aware. Unfortunately, according to Figure 4a, the same hypothesis is not true for Cross-Site Scripting: in fact, the ratio of well-known applications vulnerable to Cross-Site Scripting has been relatively constant in the past six years.

Even though the empirical evidence also does not support our second hypothesis, we noticed one characteristic that is common to both types of vulnerabilities: popular applications, probably because they are analyzed in more detail, typically have a higher number of reported vulnerabilities. The results, shown in Figures 5a and 5b, suggest that it would be useful to investigate how these vulnerabilities have evolved in the lifetime of the applications.



**Fig. 5.** Popularity of applications across the distribution of the number of vulnerability reports

### 3.3 Vulnerability Lifetime

So far, we determined that a constant, large number of simple, easy-to-exploit vulnerabilities are still found in many web applications today. Also, we determined that that the high number of reports is driven by an increasing number of vulnerable applications, and not by a small number of popular applications. Based on these findings, we formulate our third hypothesis:

**Hypothesis 3.** *Even though the number of reported vulnerable applications is growing, each application is becoming more secure over time.*

This hypothesis is important, because, if true, it would mean that web applications, in particular the well-known products, are becoming more secure. To verify this hypothesis, we studied the lifetimes of Cross-Site Scripting and SQL Injection vulnerabilities in the ten most-affected open source applications according to the NIST NVD database.

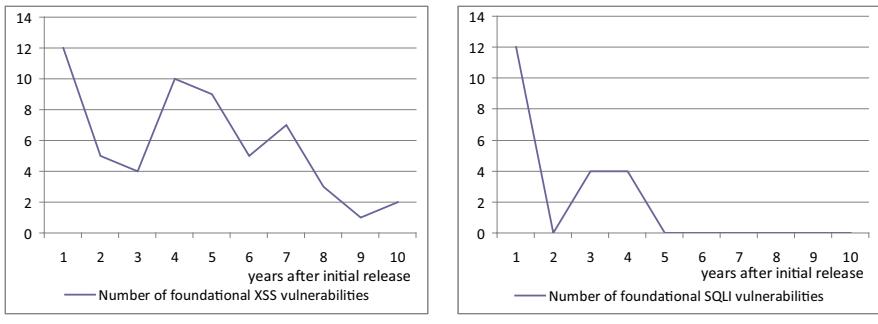
By analyzing the changelogs, for each application, we extracted in which version a vulnerability was introduced and in which version the vulnerability was fixed. In order to obtain reliable insights into the vulnerabilities lifetime, we excluded the vulnerability reports that were not confirmed by the respective vendor. For our analysis, we used the CPE identifiers in the NVD database, the external vulnerability sources, the vulnerability information provided by the vendor, and we also extract information from the version control systems (CVS, or SVN) of the different products.

Table 1a and Table 1 show a total of 147 Cross-Site Scripting and 52 SQL Injection vulnerabilities in the most affected applications. The tables distinguish *foundational* and *non-foundational* vulnerabilities. Foundational vulnerabilities are vulnerabilities that were present in the first version of an application, while non-foundational vulnerabilities were introduced after the initial release.

We observed that 39% of the Cross-Site Scripting vulnerabilities are foundational and 61% are non-foundational. For SQL Injection, these percentages are 42% and 58%. These results suggest that most of the vulnerabilities are introduced by new functionality that is built into new versions of a web application.

**Table 1.** Foundational and non-foundational vulnerabilities in the ten most affected open source web applications

(a) Cross-Site Scripting			(b) SQL Injection		
	Foundational	Non-Foundational		Foundational	Non-Foundational
bugzilla	4	7	bugzilla	1	8
drupal	0	22	coppermine	1	3
joomla	5	4	e107	0	3
mediawiki	3	21	joomla	4	0
mybb	9	2	moodle	0	3
phorum	3	5	mybb	9	3
phpbb	4	2	phorum	0	4
phpmyadmin	14	13	phpbb	3	0
squirrelmail	10	4	punbb	4	2
wordpress	6	9	wordpress	0	4
Total	58	89	Total	22	30



**Fig. 6.** Time elapsed between initial release and vulnerability disclosure

Finally, we investigated how long it took to discover the *foundational* vulnerabilities. Figure 6a and Figure 6b plot the number of foundational vulnerabilities that were disclosed after a certain amount of time had elapsed after the initial release of the applications. The graphs show that most SQL Injection vulnerabilities are usually discovered in the first year after the release of the product. For Cross-Site Scripting vulnerabilities, the result is quite different. Many foundational vulnerabilities are discovered even 10 years after the code was initially released. This observation suggests that it is very problematic to find Cross-Site Scripting vulnerabilities compared to SQL Injection vulnerabilities. We believe that this difference is caused by the fact that the attack surface for SQL Injection attacks is much smaller when compared with Cross-Site Scripting. Therefore, it is easier for developers to identify (and protect) all the sensitive entry points in the application code.

The difficulty of finding Cross-Site Scripting vulnerabilities is confirmed by the average elapsed time between the initial software release and the disclosure of foundational vulnerabilities. For SQL Injection vulnerabilities, this value is 2 years, while for Cross-Site Scripting is 4.33 years.

## 4 Related Work

Our work is not the first study of vulnerability trends based on CVE data. In [2], Christey et al. present an analysis of CVE data covering the period 2001 - 2006. The work is based on manual classification of CVE entries using the CWE classification system. In contrast, [22] uses an unsupervised learning technique on CVE text descriptions and introduces a classification system called '*topic model*'. While the works of Christey et al. and Neuhaus et al. focus on analysing general trends in vulnerability databases, our work specifically focuses on web application vulnerabilities, and, in particular, Cross-Site Scripting and SQL Injection. We have investigated the reasons behind the trends.

Clark et al. present in [3] a vulnerability study with a focus on the early existence of a software product. The work demonstrates that re-use of legacy code is a major contributor to the rate of vulnerability discovery and the number of vulnerabilities found. In contrast to our work, the paper does not focus on web applications, and it does not distinguish between particular types of vulnerabilities.

Another large-scale vulnerability analysis study was conducted by Frei et al. [5]. The work focuses on zero-day exploits and shows that there has been a dramatic increase in such vulnerabilities. Also, the work shows that there is a faster availability of exploits than of patches.

In [12], Li et al. present a study on how the number of software defects evolve over time. The data set of the study consists of bug reports of two Open Source software products that are stored in the Bugzilla database. The authors show that security related bugs are becoming increasingly important over time in terms of absolute numbers and relative percentages, but do not consider web applications.

Ozment et al. [26] studied how the number of security issues relate to the number of code changes in OpenBSD. The study shows that 62 percent of the vulnerabilities are *foundational*; they were introduced prior to the release of the initial version and have not been altered since. The rate at which foundational vulnerabilities are reported is decreasing, somehow suggesting that the security of the same code is increasing. In contrast to our study, Ozment et al.'s study does not consider the security of web applications.

To the best of our knowledge, we present the first vulnerability study that takes a closer, detailed look at how two popular classes of web vulnerabilities have evolved over the last decade.

## 5 Discussion and Conclusion

Our findings in this study show that the complexity of Cross Site Scripting and SQL Injection exploits in vulnerability reports have not been increasing. Hence, this finding suggests that the majority of vulnerabilities are not due to sanitization failure, but due to the absence of input validation. Despite awareness programs provided by MITRE [19], SANS Institute [4] and OWASP [28], application developers seem to be neither aware of these classes of vulnerabilities, nor are able to implement effective countermeasures.

Furthermore, our study suggests that a main reason why the number of web vulnerability reports have not been decreasing is because many more applications are now vulnerable to flaws such as Cross-Site Scripting and SQL Injection. In fact, we observe a trend that SQL Injection vulnerabilities occur more often in an increasing number of unpopular applications.

Finally, when analyzing the most affected applications, we observe that years after the initial release of an application, Cross-Site Scripting vulnerabilities concerning the initial release are still being reported. Note that this is in contrast to SQL Injection vulnerabilities. We believe that one of the reasons for this observation could be because SQL Injection problems may be easier to fix (e.g., by using stored procedures).

The empirical data we collected and analyzed for this paper supports the general intuition that web developers are bad at securing their applications. The traditional practice of writing applications and then testing them for security problems (e.g., static analysis, blackbox testing, etc.) does not seem to be working well in practice. Hence, we believe that more research is needed in securing applications by design. That is, the developers should not be concerned about problems such as Cross Site Scripting or SQL Injection. Rather, the programming language or the platform should make sure that the problems do not occur when developers produce code (e.g., similar to solutions such as in [29] or managed languages such as C# or Java that prevent buffer overflow problems).

**Acknowledgments.** The research leading to these results was partially funded by the European Union Seventh Framework Programme (FP7/2007-2013) from the contract N 216917 (for the FP7-ICT-2007-1 project MASTER) and N 257007. This work has also been supported by the POLE de Compétitivité SCS (France) through the MECANOS project and the French National Research Agency through the VAMPIRE project. We would also like to thank Secure Business Austria for their support.

## References

1. Bates, D., Barth, A., Jackson, C.: Regular expressions considered harmful in client-side xss filters. In: WWW 2010: Proceedings of the 19th International Conference on World Wide Web, pp. 91–100. ACM, New York (2010)
2. Christey, S.M., Martin, R.A.: Vulnerability type distributions in cve (2007), <http://cwe.mitre.org/documents/vuln-trends/index.html>
3. Clark, S., Frei, S., Blaze, M., Smith, J.: Familiarity breeds contempt: The honeymoon effect and the role of legacy code in zero-day vulnerabilities. In: Annual Computer Security Applications Conference (2010)
4. Dhamankar, R., Dausin, M., Eisenbarth, M., King, J.: The top cyber security risks (2009), <http://www.sans.org/top-cyber-security-risks/>
5. Frei, S., May, M., Fiedler, U., Plattner, B.: Large-scale vulnerability analysis. In: LSAD 2006: Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense, pp. 131–138. ACM, New York (2006)
6. Microsoft Inc. Msdn code analysis team blog (2010), <http://blogs.msdn.com/b/codeanalysis/>

7. Jim, T., Swamy, N., Hicks, M.: Defeating script injection attacks with browser-enforced embedded policies. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 601–610. ACM, New York (2007)
8. Johns, M., Beyerlein, C., Giesecke, R., Posegga, J.: Secure Code Generation for Web Applications. In: Massacci, F., Wallach, D., Zannone, N. (eds.) ESSoS 2010. LNCS, vol. 5965, pp. 96–113. Springer, Heidelberg (2010)
9. Jovanovic, N., Kruegel, C., Kirda, E.: Pixy: A static analysis tool for detecting web application vulnerabilities (short paper). In: SP 2006: Proceedings of the 2006 IEEE Symposium on Security and Privacy, pp. 258–263. IEEE Computer Society, Washington, DC (2006)
10. Kirda, E., Kruegel, C., Vigna, G., Jovanovic, N.: Noxes: a client-side solution for mitigating cross-site scripting attacks. In: SAC 2006: Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 330–337. ACM, New York (2006)
11. Kouns, J., Todd, K., Martin, B., Shettler, D., Tornio, S., Ingram, C., McDonald, P.: The open source vulnerability database (2010), <http://osvdb.org/>
12. Li, Z., Tan, L., Wang, X., Lu, S., Zhou, Y., Zhai, C.: Have things changed now?: an empirical study of bug characteristics in modern open source software. In: ASID 2006: Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability, pp. 25–33. ACM, New York (2006)
13. Livshits, B., Erlingsson, Ú.: Using web application construction frameworks to protect against code injection attacks. In: PLAS 2007: Proceedings of the 2007 Workshop on Programming Languages and Analysis for Security, pp. 95–104. ACM, New York (2007)
14. Livshits, V.B., Lam, M.S.: Finding security errors in Java programs with static analysis. In: Proceedings of the 14th Usenix Security Symposium, pp. 271–286 (August 2005)
15. Martin, B., Brown, M., Paller, A., Kirby, D.: 2010 cwe/sans top 25 most dangerous software errors (2010), <http://cwe.mitre.org/top25/>
16. Mavituna, F.: Sql injection cheat sheet (2009),  
<http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
17. Mell, P., Scarfone, K., Romanosky, S.: A complete guide to the common vulnerability scoring system version 2.0 (2007),  
<http://www.first.org/cvss/cvss-guide.html>
18. MITRE. Common platform enumeration, cpe (2010), <http://cpe.mitre.org/>
19. MITRE. Common vulnerabilities and exposures, cve (2010),  
<http://cve.mitre.org/>
20. MITRE. Common weakness enumeration, cwe (2010), <http://cwe.mitre.org/>
21. MITRE. Mitre faqs (2010), <http://cve.mitre.org/about/faqs.html>
22. Neuhaus, S., Zimmermann, T.: Security trend analysis with cve topic models. In: Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering (November 2010)
23. Newsome, J., Song, D.X.: Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software. In: NDSS. The Internet Society (2005)
24. Nguyen-Tuong, A., Guarneri, S., Greene, D., Shirley, J., Evans, D.: Automatically hardening web applications using precise tainting. In: SEC 2005, pp. 295–308. Springer, Heidelberg (2005)
25. Computer Security Division of National Institute of Standards and Technology. National vulnerability database version 2.2 (2010), <http://nvd.nist.gov/>

26. Ozment, A., Schechter, S.E.: Milk or wine: does software security improve with age? In: USENIX-SS 2006: Proceedings of the 15th Conference on USENIX Security Symposium. USENIX Association, Berkeley (2006)
27. Pietraszek, T., Berghe, C.V.: Defending Against Injection Attacks Through Context-Sensitive String Evaluation. In: Valdes, A., Zamboni, D. (eds.) RAID 2005. LNCS, vol. 3858, pp. 124–145. Springer, Heidelberg (2006)
28. The Open Web Application Security Project. Owasp top 10 - 2010, the ten most critical web application security risks (2010)
29. Robertson, W., Vigna, G.: Static enforcement of web application integrity through strong typing. In: Proceedings of the 18th Conference on USENIX Security Symposium, pp. 283–298. USENIX Association (2009)
30. RSnake. XSS (cross site scripting) cheat sheet esp: for filter evasion (2009), <http://ha.ckers.org/xss.html>
31. Vikram, K., Prateek, A., Livshits, B.: Ripley: automatically securing web 2.0 applications through replicated execution. In: CCS 2009: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 173–186. ACM, New York (2009)
32. Vogt, P., Nentwich, F., Jovanovic, N., Kruegel, C., Kirda, E., Vigna, G.: Cross site scripting prevention with dynamic data tainting and static analysis. In: In Proceedings of 14th Annual Network and Distributed System Security Symposium, NDSS 2007 (2007)
33. Wassermann, G., Su, Z.: Sound and Precise Analysis of Web Applications for Injection Vulnerabilities. In: Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation. ACM Press, New York (2007)
34. Wassermann, G., Su, Z.: Static Detection of Cross-Site Scripting Vulnerabilities. In: Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany. ACM, New York (2008) (in press)
35. 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. USENIX Association, Berkeley (2006)
36. Yu, D., Chander, A., Inamura, H., Serikov, I.: Better abstractions for secure server-side scripting. In: WWW 2008: Proceeding of the 17th International Conference on World Wide Web, pp. 507–516. ACM, New York (2008)

# Re-evaluating the Wisdom of Crowds in Assessing Web Security

Pern Hui Chia and Svein Johan Knapskog

Q2S\* NTNU, 7491 Trondheim, Norway

**Abstract.** We examine the outcomes of the Web of Trust (WOT), a user-based system for assessing web security and find that it is more comprehensive than three automated services in identifying ‘bad’ domains. Similarly to PhishTank, the participation patterns in WOT are skewed; however, WOT has implemented a number of measures to mitigate the risks of exploitation. In addition, a large percentage of its current user inputs are found to be based on objective and verifiable evaluation factors. We also confirm that users are concerned not only about malware and phishing. Online risks such as scams, illegal pharmacies and misuse of personal information are regularly brought up by the users. Such risks are not evaluated by the automated services, highlighting the potential benefits of user inputs. We also find a lack of sharing among the vendors of the automated services. We analyze the strengths and potential weaknesses of WOT and put forward suggestions for improvement.

## 1 Introduction

Security on the web remains a challenging issue today. Losses due to online banking fraud in the UK alone stood at £59.7 million in 2009, with more than 51,000 phishing incidents recorded (up 16% from 2008) [1]. Provos et al. [2] found that over 3 million malicious websites initiate drive-by downloads and about 1.3% of all Google search queries get more than one malicious URL in the result page. Meanwhile, Zhuge et al. [3] found that 1.49% of Chinese websites, sampled using popular keywords on Baidu and Google search engines, are malicious.

There is also a lack of efficient services to identify sites that are not outright malicious, but are ‘bad’ in the sense that they try to trick or harm users in many aspects, such as scams, deceptive information gathering and misuse of user data. Several fraudulent activities such as money-mule recruitment and illegal online pharmacies seem to have fallen out of the specific responsibilities or interests of the authorities and security vendors. While banking-phishing sites are taken down between 4 to 96 hours, the average life-time was found to be 2 weeks for mule-recruitment and 2 months for online pharmacy sites [4]. Problems with the adult sites may also be serious; while it is a personal judgment whether adult

\* Centre for Quantifiable Quality of Service in Communication Systems (Q2S), Centre of Excellence, appointed by the Research Council of Norway, is funded by the Research Council, NTNU and UNINETT. <http://www.q2s.ntnu.no>

content in general is inappropriate, Wondracek et al. [5] confirmed that adult sites are plagued with issues such as malware and script-based attacks and they frequently use aggressive or inappropriate marketing methods.

Online certification issuers, such as BBBOnline.org and TRUSTe.com strive to distinguish ‘good’ sites from the ‘bad’ ones. This is, however, not a straightforward task. Most websites are not entirely good or bad. There is also sometimes a conflict of interest. Problems, such as adverse-selection [6] have been observed when certifiers adopt lax requirements to certify sites in the ‘gray’ category.

### 1.1 The Wisdom of Crowds for Security

A typical argument against the idea of *the wisdom of crowds for security* is on the limited ability of ordinary users in providing reliable security evaluation. There is a general uneasiness in relying on the ordinary users for this seemingly serious task. Indeed, different from the general quality assessment, an incorrect security evaluation can cause harm to the users. Yet, this should not preclude the feasibility of collating user inputs for security purposes. Surowiecki gives multiple real life examples where inputs by non-experts collectively performed better than experts’ advices when handling complex and serious tasks [7].

PhishTank[8] and Web of Trust (WOT)[9] are two of the few existing systems that employ the wisdom of crowds to improve web security. PhishTank solicits for user reporting and voting against sites suspected to be phishes, while WOT collects public opinions on the trustworthiness, vendor reliability, privacy and child-safety aspects of domains. Both services operate on the principle that a collective decision by ordinary users, when harnessed wisely, can yield good outcomes as errors made by individuals cancel out each other. There is also the advantage of scale to cater for a large volume of items needing an evaluation.

In this work, we measure the reliability of WOT compared to 3 automated services by well known vendors, namely, McAfee’s SiteAdvisor[10], Norton’s Safe Web[11] and Google’s Safe Browsing Diagnostic Page[12]. We also investigate the participation pattern in WOT. Our findings can be summarized as follows:

- Only a few sites are commonly classified as **bad** by the prominent security vendors, indicating a lack of data sharing.
- WOT’s coverage for general sites is low compared to the automated services.
- WOT’s coverage increases when considering only domains registered in regions where active user participation is currently observed.
- WOT is more comprehensive in identifying the ‘bad’ domains.
- False negatives in identifying ‘bad’ domains are more often labeled as **unknown** by WOT, while they are often wrongly labeled as **good** by the other services.
- Contribution ratios in WOT are skewed with the comment contribution following a power law distribution.
- WOT has built a number of mitigation measures against manipulation.
- A majority of the current user inputs in WOT is based on objective evaluation criteria, and hence verifiable.
- User concerns on web security are not limited to malware and phishing.

## 2 Related Work

Surowiecki [7] outlines 4 conditions for a wise crowd to outperform a few experts. Firstly, the crowd members should be *diverse* (not homogenous). They should also have *independent thought processes* to avoid mere information cascade. The crowds should be *decentralized* to tap into local knowledge and specialization, which should be collated wisely with *a good aggregation strategy*.

In [13], Moore and Clayton evaluated the reliability and contribution patterns in PhishTank. They found that the participation ratio in PhishTank was highly skewed (following a power-law distribution), making it particularly susceptible to manipulation. Compared to a commercial phishing report, they also found that PhishTank was slightly less comprehensive and slower in reaching a decision. Our work is inspired by theirs, combined with the curiosity of why PhishTank has become widely adopted despite the initial criticisms.

While a number of studies look at the efficiency of various blacklists or tools for the issue of phishing (e.g.,[14,15]), there is little effort in evaluating the tools for web security as a whole. To our knowledge, our study is the first to evaluate the reliability of WOT, comparing it with three automated alternatives.

## 3 The Web of Trust (WOT)

WOT is a reputation system that collates user inputs into global ratings about sites under evaluation. It takes the form of an open-source browser add-on and a centralized database [9]. User inputs and the evaluation outcomes are structured around 4 aspects with ratings ranging from very poor (0-19), poor (20-39), unsatisfactory (40-59) to good (60-79) and excellent (80-100%). WOT describes the 4 aspects as follows:

- **Trustworthiness (Tr):** whether a site can be trusted, is safe to use, and delivers what it promises. A ‘poor’ rating may indicate scams or risks (e.g., identity theft, credit card fraud, phishing, viruses, adware or spyware).
- **Vendor Reliability (Vr):** whether a site is safe for business transactions. A ‘poor’ rating indicates a possible scam or a bad shopping experience.
- **Privacy (Pr):** whether a site has a privacy policy that protects sensitive information (e.g., whether it has opt-in privacy options or allows users to determine what can be made public and what should remain private). A ‘poor’ rating indicates concern that user data may be sold to third parties, be stored indefinitely or given to law enforcement without a warrant.
- **Child-Safety (Cs):** whether a site contains adult content, violence, vulgar or hateful language, or content that encourages dangerous or illegal activities.

WOT applies Bayesian inference to weigh user inputs differently based on the reliability of individual contributors, judging from their past rating behaviors. Individual user ratings are kept private to the contributors. Neither is the actual formula used in the computation publicly available. WOT argues that the hidden formula and individual inputs, plus the Bayesian inference rule, help to mitigate

typical threats facing reputation and recommender systems such as a Sybil attack in which dishonest users register multiple identities to attempt influencing the outcomes. The aggregate rating is accompanied by a confidence level (0-100%) rather than the count of the individual ratings. The developers argue that the confidence level is more appropriate as it takes into account both the number of inputs and the probable reliability of the contributors. WOT requires a minimal confidence level before publishing the aggregate rating.

Besides numerical ratings, users can also comment about the sites under evaluation. To give a comment, they must first register themselves on WOT's website. Non-registered users can only rate a site via the add-on, which gives a unique pseudonym to every WOT user. Users select one out of 17 categories which best describes their comment. Comments do not count towards the aggregate ratings. Unlike the individual ratings, they are publicly accessible.

WOT has built a number of community features on its website, including a personal page per registered user, a scorecard for each evaluated site, messaging tools, a discussion forum, a wiki, and mechanisms to call for public evaluation on specific domains. Meanwhile, the browser add-on allows a user to conveniently rate the sites he visits, in addition to signaling the reputation of different URI links, and warning the user as he navigates to sites that have been given a 'poor' rating. The child-safety rating is ignored by default, but the settings (for risk signaling and warning in general) are configurable to suit different users.

Besides user ratings and comments, WOT also receives inputs from trusted third parties. For example, it receives blacklists of phishes, spammers and illegal pharmacies from PhishTank[8], SpamCop[16] and LegitScript[17], respectively.

## 4 Data Collection

To evaluate the reliability of WOT, we compared its aggregate ratings with the outcomes provided in the querying pages of the three automated services, as identified in Section 1.1. We collected the outcomes on 20,000 sites randomly selected from the top million frequently visited sites as published by Alexa[18]. This gives us a realistic evaluation scenario in which we measure the reliability of WOT for sites that users normally visit. For each site, our program queried the assessment report from each service, parsed and stored the result (referred to as dataset-I). The querying process took place from the end of July to mid of August 2010. We have confirmed with the developers that WOT does not take inputs from any of the three automated services.

In addition, we have requested and obtained two more datasets (hereafter referred to as dataset-II and dataset-III) from the developers. Dataset-II contains the contribution level of 50,000 randomly selected users, out of >1.5 million registered users at the time of data collection. It describes the total numbers of ratings and comments which have been contributed by a user, as well as his date of registration. Dataset-III consists of 485,478 comments randomly selected from >8 million at that time. Besides the comment text, it includes the date of writing and a category chosen by the contributor to best describe the comment. The comments evaluate a total of 412,357 unique sites. To study the users'

commenting behavior, we downloaded also the aggregate ratings of the 412k sites using the public query API of WOT [9]. Both dataset-II and III contain only information that are publicly accessible for all who login to WOT's website.

## 5 Analysis

We started by studying the characteristics of the automated services:

- **McAfee’s SiteAdvisor (SA)**[10] evaluates a site based on various proprietary and automated tests on aspects such as downloads, browser exploits, email, phishing, annoyance factors (e.g., pop-ups) and affiliations with other sites. SiteAdvisor also receives inputs from TrustedSource[19] which evaluates aspects such as site behavior, traffic and linking patterns, as well as site registration and hosting. Among others, it helps SiteAdvisor to identify spamming and phishing sites. SiteAdvisor allows users to comment on a particular site, but comments are not factored into the evaluation outcomes.
- **Norton’s Safe Web (SW)**[11] tests if a site imposes threats e.g., drive-by download, phishing, spyware, Trojan, worm, virus, suspicious browser change, joke program and identity theft. It collects also user ratings and comments, but like SiteAdvisor, they do not count towards the overall rating.
- **Google’s Safe Browsing Diagnostic Page (SBDP)**[12] warns about sites that have been the hosts or intermediaries which download and install (malicious) software on a user’s device without consent. It should be noted that warnings about phishing activities are not included in the diagnostic page. Phishing reports may only be accessible via the Safe Browsing API. We note that this should not affect our results as we do not expect the frequently visited sites (used in our study) to be potential phishes.

To enable comparison, we mapped the evaluation outcomes of the respective services into 4 classes: **good**, **caution**, **bad** and **unknown**, as shown in Table 1. We classified WOT’s ratings based on its default risk signaling strategy, which regards Trustworthiness (Tr) as the most important evaluation aspect, given that it covers the scopes of Vendor Reliability (Vr) and Privacy (Pr). A site is considered **good** if its Tr rating is  $\geq 60$  without any credible warning (i.e., rating  $< 40$  and confidence level  $\geq 8$ ) in Vr or Pr.<sup>1</sup> We did not consider child-safety in the classification as it is ignored by the browser add-on in the default settings. Neither is content-appropriateness evaluated by the automated services.

### 5.1 The Reliability of WOT

We first evaluated the coverage of individual services (see Table 2). Coverage measures the fraction of evaluated sites (i.e., those with an assessment outcome ≠ **unknown**). SiteAdvisor has the highest coverage while WOT scores the lowest.

---

<sup>1</sup> We evaluated also the case which weighs all aspects equally (i.e., a site is classified as **bad** if either Tr, Vr or Pr is  $< 40$ ), but found no significant changes in results.

**Table 1.** Aligning the evaluation outcomes of different services

	WOT	SiteAdvisor	Safe Browsing DP	Safe Web
Good	Tr $\geq$ 60, and no credible warning in Vr or Pr	Green: very low or no risk	Site not currently listed as suspicious, and Google has visited it in the past 90 days.	Safe
Caution	60 > Tr $\geq$ 40, and no credible warning in Vr or Pr	Yellow: minor risk	Site not currently listed as suspicious, but part of the site was listed for suspicious activity in the past 90 days.	Caution
Bad	Tr < 40, or there is a credible warning in Vr or Pr	Red: serious risk	Site is listed as suspicious.	Warning
Unknown	No Tr rating, and no credible warning in Vr or Pr	Gray: not rated	Site not listed as suspicious, and Google has not visited it the past 90 days.	Untested

This can be attributed to the fact that decisions in WOT depend on manual user contribution. It may be also due to that the popularity of WOT is still limited to in Europe and North America, as shown by the breakdown of user activity by region on WOT’s statistics page. Considering only sites registered in the North America, the EU plus Norway and Switzerland, the coverage of WOT increases from 51.23% to 67.46%, while the coverage of SiteAdvisor increases to 94.98%.

The breakdown of the evaluated outcomes is included in Table 2. SiteAdvisor classifies 1.48% sites as **bad**. This is interestingly close to the result in [3] which found 1.49% of Chinese sites, sampled using popular keywords on Baidu and Google (different from our sampling method), are malicious. WOT classifies 3.16% sites to be **bad**. This larger value is likely due to the broader evaluation scope of WOT, which is not limited to the malicious sites only. In comparison, results by Safe Web and Safe Browsing Diagnostic Page may be too optimistic.

The Venn diagram in Figure 1 shows that out of 296 and 102 **bad** sites that SiteAdvisor and Safe Web find respectively, only 8 are on their common black-list. The small percentage of the common findings about **bad** sites indicates the different testing methodologies employed and a lack of sharing between the two vendors. The lack of data sharing is also notable in the anti-phishing industry [20]. Previously this was a problem also in the anti-virus industry, but security vendors were found to have learned the lesson and are now sharing virus samples [20]. On the other hand, WOT finds 21 **bad** sites in common with Safe Web and 73 with SiteAdvisor. This hints on the better ability of WOT in identifying ‘bad’ sites that have been found by the others.

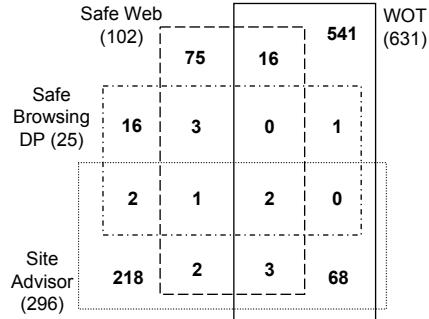
We measured Recall ( $R = T_p / (T_p + F_n)$ ), Precision ( $P = T_p / (T_p + F_p)$ ) and F-Score ( $FS = 2PR / (P + R)$ ) to quantify the reliability in identifying ‘bad’ sites. A challenge here is to determine the count of true positives ( $T_p$ ), false positives ( $F_p$ ) and false negatives ( $F_n$ ) given that we do not know the ‘correct’

**Table 2.** Coverage and the percentage of assessment outcomes

	Cov. (%)	Outcomes (%)		
		Bad	Caution	Good
WOT	51.23 <sup>a</sup>	3.16	2.15	45.93
SA	87.84	1.48	0.47	85.90
SBDP	55.65 <sup>b</sup>	0.13	1.63	53.90
SW	68.09	0.51	0.38	67.21

<sup>a</sup> Based on the default risk signaling strategy of WOT, and not including the child-safety aspect, as shown in Table 1.

<sup>b</sup> We regard a site that is not currently blacklisted and that has not been visited by Google’s web bot in the past 90 days as ‘not tested’.

**Fig. 1.** Venn diagram shows the divergence in the classification of ‘bad’ sites. Out of 948 that have been marked as **bad** by any services, only 2 receive the same verdict from all, while only 98 sites are classified as **bad** by >1 services.**Table 3.** True positives, false positives, false negatives in Optimistic or Conservative [in brackets] consensus.  $F_{n,c}$ ,  $F_{n,g}$ ,  $F_{n,u}$  denote the false negative cases where a ‘bad’ site is wrongly labelled as **caution**, **good** and **unknown** respectively.

Finding of this service	Findings of other services				
	Bad w/o any good	Mixed of good & bad	Caution only	Good w/o any bad	All unknown
Bad	$T_p$ [ $T_p$ ]	$F_{p,m}$ [ $T_p$ ]	$F_{p,c}$ [ $F_{p,c}$ ]	$F_{p,g}$ [ $F_{p,g}$ ]	$F_{p,u}$ [ $F_{p,u}$ ]
Caution	$F_{n,c}$ [ $F_{n,c}$ ]	-	$[F_{n,c}]$	-	-
Good	$F_{n,g}$ [ $F_{n,g}$ ]	-	$[F_{n,g}]$	-	-
Unknown	$F_{n,u}$ [ $F_{n,u}$ ]	-	$[F_{n,u}]$	-	-

**Table 4.** Recall (R), Precision (P) and F-Score (FS) of individual services. R,  $F_{n,c}$ ,  $F_{n,g}$  and  $F_{n,u}$  add up to 100%. The 5th row considers **bad** sites to include only those with a credible warning (such that the add-on will prompt an explicit warning dialog to the user). The last 3 rows compare only among the automated services, excluding the assessment by WOT in the consensus outcomes.

	Optimistic consensus (%)						Conservative consensus (%)					
	R	P	FS	$F_{n,g}$	$F_{n,u}$	$F_{n,c}$	R	P	FS	$F_{n,g}$	$F_{n,u}$	$F_{n,c}$
WOT	15.3	1.7	3.1	11.1	72.2	1.4	22.1	14.3	17.3	22.6	49.4	5.9
SA	8.3	3.4	4.8	57.5	27.5	6.7	10.7	26.4	15.2	69.7	15.6	4.0
SW	4.1	8.8	5.6	59.0	34.2	2.7	3.1	26.5	5.5	71.6	23.6	1.7
SBDP	2.5	16.0	4.3	40.0	55.6	1.9	1.0	36.0	1.9	47.6	46.2	5.2
WOT [cw]	13.9	2.5	4.3	-	-	-	17.2	17.8	17.5	-	-	-
SA [auto]	10.0	2.0	3.4	68.3	18.3	3.3	8.3	3.4	4.8	68.6	20.7	2.5
SW [auto]	2.9	4.9	3.6	65.7	28.0	3.4	3.5	10.8	5.3	66.1	26.9	3.5
SBDP [auto]	3.3	16.0	5.4	42.3	51.2	3.3	2.1	32.0	3.9	43.6	44.6	9.7

assessment outcomes or truth values. We approached this by comparing the outcomes of a particular service with the consensus result of the three others. Thus, in this context, Recall (R) describes the success rate of a service in recognizing all **consensus-bad** sites, while Precision (P) measures the fraction of **bad** sites identified by a service matching the consensus of the others. We define two types of consensus: optimistic and conservative. In the optimistic case, the **consensus-bad** sites are the ones classified as **bad** by other services without any contradictory classification of **good**. In the conservative case, the **consensus-bad** sites include those that have mixed **bad** and **good** verdicts by individual services. We note that the conservative case may depict a more realistic scenario given the divergence in the classification of **bad** sites. Table 3 shows the definitions of  $T_p$ ,  $F_p$  and  $F_n$ . Table 4 presents the R, P and FS values of different services.

Having the highest R in both optimistic and conservative cases, we find that WOT renders a more comprehensive protection against ‘bad’ sites in comparison to the three automated services. On a closer look, we also find that in the event that WOT fails to warn against sites having a **consensus-bad** rating, a higher percentage of these false-negatives are classified by WOT as **unknown** or **caution** rather than **good**, as indicated by the  $F_{n,u}$ ,  $F_{n,c}$ ,  $F_{n,g}$  values in Table 4. Conversely, most of the false-negatives by SiteAdvisor and Safe Web are classified as **good** rather than **unknown** or **caution**. This adds on to the reliability of WOT. Meanwhile, users may have to remain cautious even when a site has received a **good** rating from SiteAdvisor or Safe Web.

However, WOT has a low Precision (P) value in comparison to the others. As we learned from the developers that the browser add-on will only prompt the user a warning dialog when a ‘poor’ or ‘very poor’ rating (in either aspect of Tr, Vr or Pr) has a confidence level  $\geq 8$  (i.e., credible), we measured the precision of WOT considering ‘bad’ sites to be only those with such a credible warning (i.e., those that will be explicitly warned against). As shown in the 5th row of Table 4, the Precision of WOT increases, but only slightly. The low P value may reflect that that WOT covers a broader evaluation scope than the others. Yet, a very low P value may result in a situation where users habitually regard all warnings as false positives as they do not observe similar warnings from the other services. It is thus important for WOT to inform the users about the differences.

If we weigh false-positives and false-negatives equally, the tradeoff between Recall and Precision can be measured by FS – the harmonic mean of R and P. In the optimistic case, all FS values are small (3.1% to 5.6%) with Safe Web having the highest FS (despite a low R). In the conservative case, the difference in FS values becomes evident. WOT has the highest FS of 17.3%. SiteAdvisor has a FS of 15.2% and interestingly, the FS of Safe Web remains at 5.5%.

One may reason that the low R and high P values of the automated services could be an artifact of comparing them with WOT which has a broader evaluation scope. As a robustness check, we measured the reliability of the automated services using only the outputs of the other two automated services to determine

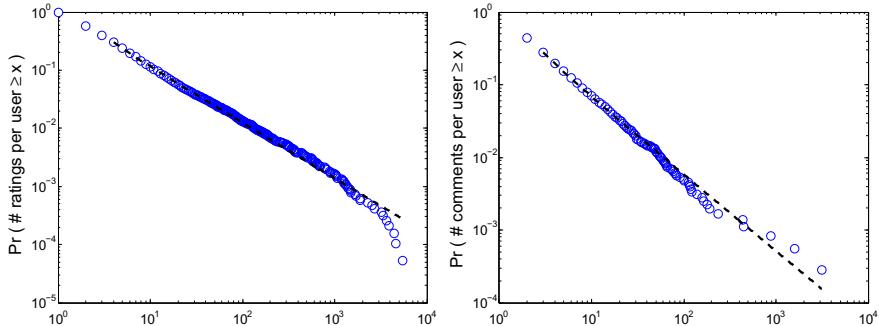
the consensus outcomes. As shown in the last three rows of Table 4, the P values drop without an evident improvement in R. All FS values are low (3.4% to 5.4%) even in the conservative case.

The above shows that WOT is reliable in comparison to the three automated services, especially when users should be cautious about web security as captured in the case of conservative consensus. Overall, WOT has shown a better ability in recognizing ‘bad’ sites among the popular online destinations. Some of its warnings may concern risks that are not currently evaluated by the others.

## 5.2 The Few Dominating Contributors

Moore and Clayton argue that the highly skewed participation ratio in Phish-Tank increases the risks of manipulation; the corruption of a few highly active users can completely undermine its validity and availability [13]. It is also not difficult for a highly active user to disrupt the system under cover of a large body of innocuous behavior [13]. We investigated if similar problems exist in WOT.

We analyzed dataset-II which describes the contribution level of 50k randomly selected users. Of these users, the total rating and comments posted are 214,872 and 20,420 respectively. However, only 38.34% of them have rated and 7.16% have commented about a site at least once.

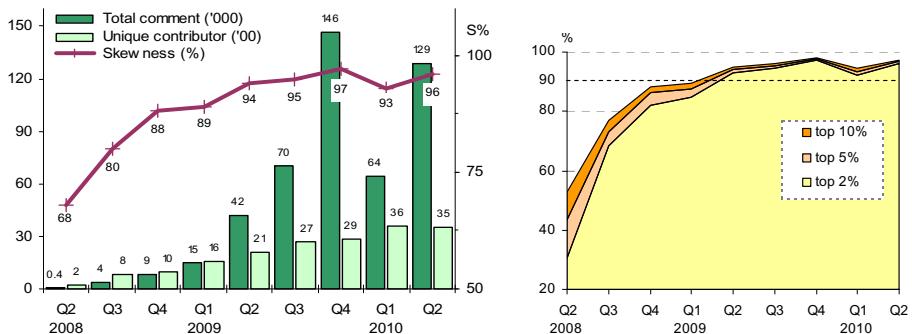


**Fig. 2.** The complementary CDF of ratings and comments. Dashed lines depict the best fitted power law distribution with  $\alpha=1.95$ ,  $x_{min}=4$  (rating, left) and  $\alpha=2.05$ ,  $x_{min}=3$  (comment, right).

The seemingly straight lines in the log-log graphs (in Figure 2) suggest that the contribution of ratings and comments could be following the power law distribution. We computed the best-fit of power-law scaling exponent  $\alpha$  and the lower cutoff  $x_{min}$  using maximum-likelihood estimation, based on the approach in [21]. We obtained the best fitted scaling exponent  $\alpha=1.95$  and lower cut-off  $x_{min}=4$  for rating contribution, and  $\alpha=2.05$  and  $x_{min}=3$  for comment contribution. The goodness-of-fit of these values were evaluated using the Kolmogorov-Smirnov (KS) test. We obtained a high  $p$ -value (0.76) for the parameters of comment

contribution, indicating that it is likely to follow a power law distribution. This is, however, not the case for rating contribution where we rejected the null hypothesis that it is power-law at the 5% significance level.

We did not proceed to test if the rating contribution follows other types of heavy-tailed distributions (e.g., log-normal, Weibull) given that it is visually intuitive that a large percentage of the contribution comes from a small group of users. We observed that the complementary cumulative distribution function (CDF) of rating contribution begins to curve-in among the top contributors<sup>2</sup> (Figure 2, bottom left). Adapting from the 80:20 rule of the Pareto principle, we measured the Skewness ( $S$ ) such that  $S$  is the largest  $k\%$  of the total inputs coming from  $(100-k)\%$  of the contributors. We found that  $S$  is 89 for rating and 95 for comment contribution. Put in words, 89% of the total ratings are provided by 11% of the rating contributors while the top 5% comment contributors gave 95% of the total comments. Both contribution ratios are skewed.



**Fig. 3.** (Left) Total comment ('000), Unique contributor ('00) and Skewness (%). (Right) Percentage of comment provision by the top 2, 5 and 10% contributors.

We then studied the evolution of user participation using dataset-III, which contains 485,478 comments contributed by 16,030 unique users. Figure 3 (left) shows an increasing number of comments and unique contributors per quarter. Unfortunately, the contribution ratio has become more skewed as WOT evolves, as shown by the  $S$  values. Since 2009 Q2, more than 90% of the total comments are actually provided by the top 2% active users as shown in Figure 3 (right). The increasing trend of skewness is likely to be caused by the mass rating tool which allows one to rate and comment 100 sites at once. The privilege to use the tool was initially given to both the Gold and Platinum users since Sep 2008 (according to WOT's wiki). As cases of misuse were detected, WOT began to award new privileges only to the Platinum users, from 28 Dec 2009. Revoking the privilege from those who have misused the tool might be the reason that has caused the dip in  $S$  and total comment during 2010 Q1.

<sup>2</sup> Excluding users who have contributed >3000 ratings, the KS test gives a  $p$ -value of 0.36, indicating that it may be a power law distribution only with an upper cut-off.

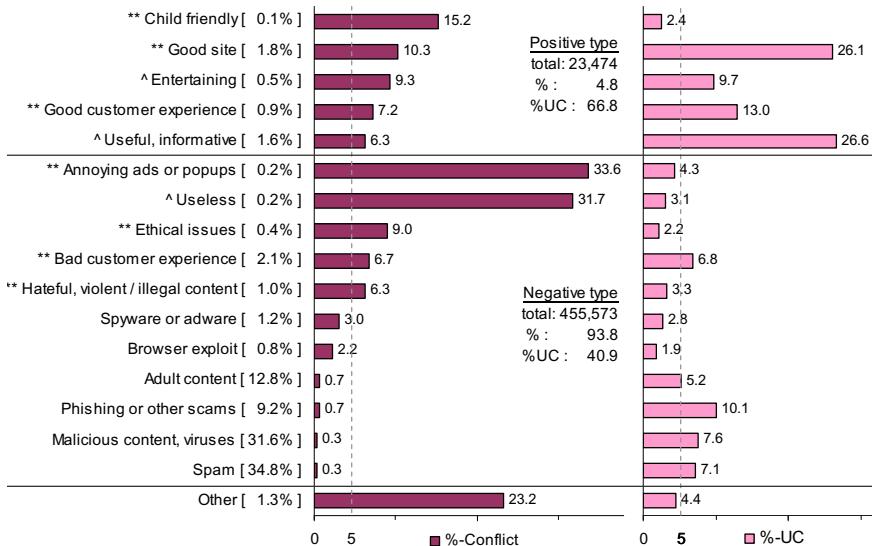
We cannot inspect the evolution of rating contribution as individual ratings are kept private in WOT. Our guess is that rating contribution evolves similarly but not as skewed given that it does not fit well as a Power Law distribution and that it has a smaller S value than that of comment. In addition, WOT has made the rating process more convenient than commenting. Using the browser add-on, users neither need to first register themselves, nor visit the WOT's website in order to give a rating.

Skewed participation patterns are not entirely unexpected; some users are naturally more inclined to contribute than the others. WOT also has put in place a number of features to mitigate the risks of exploitation. First, in its current form, security decisions in WOT are not easily guessable due to the hidden nature of the aggregation formula and individual ratings. WOT also states that it does not weigh the user inputs based on the activity level of individual contributors; the weights are computed from the reliability of their past rating behavior. These measures make the repeated cheating by a single highly active user difficult. One may be able to cast biased ratings unnoticed amidst a large number of innocuous inputs, but this is only valid if it is cost-efficient for the attacker to build up a reputation in order to rate up or down a few targeted sites. An attack may be more easily done with the help of several accomplices, or through a ‘pseudo reliability’ built by automatic rating with reference to some public blacklists. We learned from the developers that there are automatic mechanisms in WOT which monitor for suspicious user behavior. Yet, to the root of the challenges, WOT should work towards diversifying the user contribution so that it does not become a centralized/homogenous system overwhelmed with the inputs of a few. The mass rating privilege should be handled with care.

### 5.3 Exploitability, Disagreement and Subjectivity

Grouping the comments according to their respective category, we observed that there are many more comments of negative type than positive (see Figure 4). We measured the percentages of conflict (%-conflict) and unique contributors (%-UC) of each comment category. A ‘conflict’ is defined to arise when a comment of positive type is given to a site that has a poor rating ( $<40$  in either Tr, Vr, Pr or Cs aspect), or when a comment of negative type is given to a site that has a good rating ( $\geq 60$  for all aspects). A conflict can happen due to several reasons. Firstly, it can be due to the difference in scope between the comment and rating. Specifically, whether a site is useful or not, and whether it is entertaining are factors not evaluated by the four rating aspects. Secondly, assuming that the ratings reflect the true state of a site, a conflict can be due to user attempts to cheat (e.g., to defame or lie about a site of interests) or simply divergent views. We could not easily differentiate between exploitation and disagreement, but underlying the two are common factors of subjectivity and non-verifiability.

Excluding categories that are not in the scope of rating (i.e., Entertaining, Useful and informative, and Useless), we found that categories that concern user experience and content (except for ‘adult content’) have a %-conflict value of  $>5$ . In comparison, there is little conflict resulting from comments which



**Fig. 4.** %-Conflict, % of Unique Contributor and %-count [in brackets] of different comment categories. ^ denotes not in rating scope, \*\* denotes %-conflict >5.

warn about browser exploits, phishing sites or adult content. We attribute this to the different levels of objectivity. For example, feedback on whether a site has annoying ads and whether a site provides good customer experience are subjective. Meanwhile, one cannot believably allege a site for phishing, browser exploit or adult content without verifiable evidence.

In addition, we found no association between a low %-conflict value and a small group of contributors. Comments with categories such as ‘Adult content’, ‘Malicious content, viruses’ and ‘Spam’ are provided by >5% of total contributors but have a low level of conflict. Conversely, comments about ‘Child friendliness’ and ‘Ethical issues’ are given by fewer users but result a higher level of conflict.

The above observations have several implications. First, signaled by the low %-conflict, identifying a phishing site is an objective process. Given that an objective evaluation is verifiable, there is a reduced chance for successful manipulation going unnoticed, even by the highly active users. This may have served to mitigate the risks and incentives of exploitation in PhishTank. Indeed, despite the early criticisms on its highly skewed (power law) participation ratio [13], PhishTank is now adopted by multiple vendors including Yahoo!, Mozilla, Kaspersky Lab, Opera and McAfee [8].

Risks of exploitation can, however, be a real issue for WOT since several of its evaluation aspects, such as trustworthiness and vendor reliability are subjective in nature. Fortunately, in its current state, we found that a large majority of the user comments actually come under categories that have a low level of conflict e.g., ‘adult content’, ‘malicious content, viruses’, ‘spam’ and ‘phishing or other scams’. Although we cannot know for sure, the pattern exhibited here

does imply that the existing user ratings are largely based on objective criteria. While evaluation based on objective criteria do not equate honest assessment, for example one can accuse an innocent site to be malicious, such manipulation can be discovered and punished with an appropriate level of monitoring. This reduces the incentives of such an attack.

Yet, it is not unreasonable to expect an increase of subjective user inputs in the long run. 7 of the 13 comment categories in the scope of rating actually have a %-conflict value of more than 5. Comments that come under these categories were also in fact contributed by more than half of the unique contributors. Subjective opinions, if provided honestly, are valuable to a user-based system as they mark the diversity of the participants. The challenge lies in that we cannot assume the honesty of users. Subjective and non-verifiable evaluation criteria can be exploited more easily.

#### 5.4 User Concerns on Web Security

We also looked at popular words used in user comments and how the trend may have changed over time. As we discovered that a large number of comments are made with exactly the same description (likely to be caused by the mass rating tool), we used only unique comments in our analysis. We parsed for nouns and transformed them into the singular form. Table 5 shows the most frequently used words ranked in popularity. We observe that ‘spam’ and ‘scam’ are among the most common issues discussed in user comments. The word ‘information’ is also frequently used in conjunction with ‘personal’ and ‘sensitive’ describing privacy concerns. Another popular word is ‘pharmacy’ which is found in warnings against fake or illegal online pharmacy sites. The use of the word ‘phishing’ becomes dominant since late 2008. Meanwhile, concern about malware on the web, virus and Trojan included, is increasing.

**Table 5.** Popular words used in user comments per year quarter. Similar words e.g., domain, website, page (~site), scammer (~scam), program (~software) were omitted.

08'Q2	08'Q3	08'Q4	09'Q1	09'Q2	09'Q3	09'Q4	10'Q1	10'Q2
site	site	site	site	site	site	site	site	site
info	spam	spam	spam	spam	spam	spam	malware	spam
spam	criminal	info	scam	scam	malware	scam	Trojan	scam
email	email	phishing	phishing	phishing	Trojan	phishing	virus	phishing
link	info	software	software	malware	scam	info	spam	malware
people	trade	scam	pharmacy	software	phishing	malware	threat	info
pharmacy	scam	criminal	virus	info	info	pharmacy	exploit	pharmacy
software	gang	security	info	pharmacy	software	software	scam	credit card
service	porn	service	download	virus	exploit	email	phishing	abuse
child	pharmacy	warning	porn	registrar	virus	virus	info	software
privacy	brand	content	link	exploit	content	link	pharmacy	risk
product	child	email	malware	Trojan	download	Trojan	software	virus

This analysis indicates that user concerns on web security are not limited to only phishing and malware. This brings up the limitation of the automated services in catering for user concerns on online risks such as scams, illegal pharmacies, information protection and inappropriate content in general.

## 6 Discussion

The strengths of WOT lie in a number of its characteristics. First, it caters for different user concerns about web security and does so reliably. Its overall ratings are not easily guessable and hence there is little chance of manipulation. The browser add-on has also made the process of rating a site very easy. Sub-domains are designed to inherit the reputation of the parent domain unless there are sufficient ratings for the sub-domain itself, avoiding redundant user effort. WOT also encourages users to contribute responsibly by weighing the inputs according to the reliability of individual contributors through statistical analysis. In a private communication with the developers, we were told that WOT has also factored in the dynamics of aggregate ratings as the weight of individual ratings is set to decay (until the respective contributors re-visit the sites). The system is also capable of ignoring spammers and suspicious ratings as WOT monitors for unusual rating behavior automatically. Finally, the community features such as discussion forum, messaging tools and the ability to call for public evaluation for specific sites, have all contributed to a reliable reviewing process.

Yet, WOT is not without several potential weaknesses. We discuss several of them and suggest the potential mitigating strategies in the following:

- **Skewed contribution patterns.** The contribution patterns of rating and comment are skewed, most likely due to the mass rating tool. A highly skewed contribution pattern can cause WOT to be overwhelmed by the inputs of a few, violating the diversity and decentralization conditions of the wisdom of crowds. While the risks of exploitation due to a skewed participation is expected to be limited given the measures taken in WOT and the observation that a majority of the current user inputs are based on objective evaluation factors, we suggest to handle the mass rating tool with a greater care. It may be wise to restrict the highly active users to use the tool only for evaluation aspects that are objective and verifiable. At the time of writing, it is also not mandatory for them to provide the evidence of their mass ratings, although they are required to submit a comment in which it is recommended to include the relevant references and that they must be contactable by anyone who disagrees with the rating. Attention must also be given to potential gaming behavior such as building up a ‘pseudo reputation’ by simply referencing the publicly available blacklists. Essentially, WOT should work on diversifying the sources of bulk contribution.
- **A hidden approach.** While the hidden aggregation formula and user ratings may have played a part in making the assessment outcomes in WOT less easily guessable and less vulnerable to manipulation, a hidden approach

may in general result in a lack of user confidence. The situation can be more tricky given that warnings by WOT may not be frequently supported by the automated services (as characterized by the low precision value). Users unaware of the broader evaluation scope of WOT may doubt the reliability of the black-box computation and regard its warnings as mere false-positives. Neither will a hidden approach benefit from the scrutiny and suggestions for improvement from the community. It may be worth the effort for WOT to educate the users concerning its differences from the automated services. A more transparent approach capable of withstanding manipulation would be the preferred option in the long run.

- **Subjective evaluation criteria.** Subjective evaluation factors can result in contentious outcomes besides increasing the risk of manipulation. In the current state, WOT does not seem to differentiate between objective and subjective evaluation criteria. Improvement can be made in this respect. For example, the rating aggregation strategy may factor in the subjectivity level of the underlying evaluation factor. WOT may also consider tapping into the potentials of personalized communities as proposed in [22] to deal with subjective factors. Inputs from personalized communities have the advantages of being more trustworthy, relevant and impactful than those provided by unknown community members [22].

There are several limitations to our study. First, as our evaluation sample consists of sites randomly chosen from the one million most-frequently visited sites, we have not evaluated the reliability of WOT when dealing with ‘bad’ sites that are more frequently found in the long-tail of web popularity. Further, we have also not tested the timeliness of WOT’s assessment. It may appear that an assessment by WOT can take a longer time than the automated systems as it depends on user inputs and can miss out on malicious sites which are often online for a short period of time only. While these concerns are valid, we note that they are being covered in WOT by the inclusion of blacklists from trusted third party sources. Future investigation on these concerns would be interesting.

## 7 Conclusions

We have found that the Web of Trust (WOT) is more comprehensive than three popular automated services in identifying ‘bad’ domains among the frequently visited sites. Contribution patterns in WOT are found to be skewed with the comment contribution following a power-law distribution. However, WOT has implemented a number of measures to mitigate the risks of exploitation. In addition, a large majority of its current user inputs is found to be based on objective evaluation factors and hence verifiable. This may have also helped to reduce the risks and incentives of exploitation in PhishTank. We find that user concerns on web security are not limited to malware and phishing. Scams, illegal pharmacies and lack of information protection are regular issues raised but are not evaluated by the automated services. There is also an evident lack of sharing

among the vendors of the automated services. We include a discussion on the strengths and potential weaknesses of WOT which may be helpful for designing user-based security systems in general. In short, WOT clearly exemplifies that the wisdom of crowds for assessing web security can work, given a careful design.

**Acknowledgement.** We thank N. Asokan, B. Westermann and the anonymous reviewers for their comments, and the WOT developers for dataset-II, III and details about WOT.

## References

1. The UK Card Association: New Card and Banking Fraud Figures (March 10, 2010) [http://www.theukcardsassociation.org.uk/media\\_centre/press\\_releases\\_new-page/922/](http://www.theukcardsassociation.org.uk/media_centre/press_releases_new-page/922/)
2. Provos, N., Mavrommatis, P., Rajab, M., Monroe, F.: All your iFRAMES point to Us. In: Proc. USENIX Security (2008)
3. Zhuge, J., Holz, T., Song, C., Guo, J., Han, X., Zou, W.: Studying Malicious Websites and the Underground Economy on the Chinese Web. In: Proc. WEIS (2008)
4. Moore, T., Clayton, R.: The impact of incentives on notice and takedown. In: Johnson, M. (ed.) Managing Information Risk and the Economics of Security (2008)
5. Wondracek, G., Holz, T., Platzer, C., Kirda, E., Kruegel, C.: Is the Internet for Porn? An Insight into the Online Adult Industry. In: Proc. WEIS (2010)
6. Edelman, B.: Adverse selection in online “trust” certifications and search results. Electronic Commerce Research and Applications (2010)
7. Surowiecki, J.: The wisdom of crowds. Anchor Books (2005)
8. PhishTank: Vendors using PhishTank, <http://www.phishtank.com/friends.php>
9. Web of Trust (WOT), <http://www.mywot.com> Query API, [http://api.mywot.com/0.4/public\\_query2?url=<site>](http://api.mywot.com/0.4/public_query2?url=<site>)
10. McAfee SiteAdvisor, <http://www.siteadvisor.com/sites/<site>>
11. Norton Safe Web, <http://safeweb.norton.com/report/show?url=<site>>
12. Google Safe Browsing diagnostic page, <http://www.google.com/safebrowsing/diagnostic?site=<site>>
13. Moore, T., Clayton, R.C.: Evaluating the wisdom of crowds in assessing phishing websites. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 16–30. Springer, Heidelberg (2008)
14. Sheng, S., Wardman, B., Warner, G., Cranor, L.F., Hong, J., Zhang, C.: An empirical analysis of phishing blacklists. In: Proc. CEAS (2009)
15. Zhang, Y., Egelman, S., Cranor, L., Hong, J.: Phinding Phish: An Evaluation of Anti-Phishing Toolbars. In: Proc. NDSS (2007)
16. SpamCop, <http://www.spamcop.net>
17. LegitScript, <http://www.legitscript.com>
18. Alexa top million sites, <http://www.alexa.com/topsites>
19. McAfee TrustedSource, <http://www.trustedsource.org>
20. Moore, T., Clayton, R., Anderson, R.: The economics of online crime. Journal of Economic Perspectives 23(3), 3–20 (2009)
21. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-law distributions in empirical data. SIAM Review 51(4), 661–703 (2009)
22. Chia, P.H., Heiner, A.P., Asokan, N.: Use of ratings from personalized communities for trustworthy application installation. In: Proc. NordSec (2010)

# Mercury: Recovering Forgotten Passwords Using Personal Devices<sup>\*</sup>

Mohammad Mannan<sup>1, \*\*</sup>, David Barrera<sup>2</sup>, Carson D. Brown<sup>2</sup>,  
David Lie<sup>1</sup>, and Paul C. van Oorschot<sup>2</sup>

<sup>1</sup> Dept. of Electrical and Computer Engineering  
University of Toronto, Toronto, Canada

[m.mannan@utoronto.ca](mailto:m.mannan@utoronto.ca)

<sup>2</sup> School of Computer Science  
Carleton University, Ottawa, Canada

**Abstract.** Instead of allowing the recovery of original passwords, forgotten passwords are often reset using online mechanisms such as password verification questions (PVQ methods) and password reset links in email. These mechanisms are generally weak, exploitable, and force users to choose new passwords. Emailing the original password exposes the password to third parties. To address these issues, and to allow forgotten passwords to be securely restored, we present a scheme called *Mercury*. Its primary mode employs user-level public keys and a personal mobile device (PMD) such as a smart-phone, netbook, or tablet. A user generates a key pair on her PMD; the private key remains on the PMD and the public key is shared with different sites (e.g., during account setup). For password recovery, the site sends the (public key)-encrypted password to the user's pre-registered email address, or displays the encrypted password on a webpage, e.g., as a barcode. The encrypted password is then decrypted using the PMD and revealed to the user. A prototype implementation of Mercury is available as an Android application.

## 1 Introduction and Motivation

Users often forget their login passwords. This is not news to the security research community (see [26], a 1993 user study). Forgetting a password leads to a password reset through systems such as Personal Verification Questions (PVQs) or sending the new password to a pre-registered email address. Today's Web has brought about an increase in number of password-protected sites (on average 25 accounts per user [7]), leading to more forgotten passwords. Some users choose weak passwords and reuse them across many sites to cope. Password mangers have mushroomed in most desktop and smart-phone platforms as an offered solution. Some users instead keep notes, written on a piece of paper or stored in a digital file. Others rely on online reset mechanisms offered by most password-protected websites.

\* Version: November 18, 2011. Post-proceedings of Financial Cryptography and Data Security 2011.

\*\* Corresponding author.

Weaknesses in online reset techniques pose a significant threat to password based schemes. Several studies—old and new (e.g., [19], [26])—have repeatedly identified these drawbacks: answers to reset questions are publicly available or easily guessed, and even more so by close contacts. Although users may initiate reset requests only occasionally, attackers can misuse these techniques at any time. Indeed, in several real-world incidents, PVQs have been exploited to compromise user accounts [3], [23]. Such techniques have also been exploited by social engineering attacks [15, p. 272]. Beyond security, forgotten passwords often incur significant cost to current IT-dependent organizations. According to Gartner, 20 – 50% of all help desk calls are for password resets, each costing about \$70 depending on the organization type.<sup>1</sup>

Despite the critical problem of password loss by large numbers of users, little research effort has been directed towards solutions, compared to online authentication. Several mechanisms (e.g., PVQs and their proposed variants [6], [10], [25]) mandate recalling multiple infrequently-used, text-based secrets to reset a password. The “crime” of password forgetfulness is punished by burdening users with additional cognitive loads. This is clearly not a winning strategy. Others have begun to explore ways to improve such reset techniques, e.g., using pictures (rather than text questions in normal PVQ schemes) as visual cues to prompt for answers [17].

This paper offers a new approach, based on securely restoring existing passwords rather than resetting to new ones. A recent study [24] has confirmed the suspicion that users evolve passwords in a predictable manner, when they are forced to update their passwords. Our key insight is that a secure personal mobile device (PMD), such as a smart-phone or tablet, in combination with a simple to use channel between a PC and the device can solve the difficult problem of *password recovery* (i.e., delivering the original password back to the user) and reduce the need for *password reset* (i.e., creating a new password). Thus, we propose a password recovery mechanism called *Mercury*.<sup>2</sup> The basic idea is as follows: a user creates a public/private encryption key pair on her PMD using entropy from a private object (e.g., an unshared personal image, cf. [13]), or a random source. The public key is shared (e.g., during registration) with the sites hosting the user’s accounts. For password recovery, a public-key encrypted password is sent to a registered email address, or displayed on the site’s page as a barcode. Using her PMD, the user decrypts the password which is then displayed on the PMD screen. The same public key can be used with different sites. Mercury’s focus is to restore forgotten passwords, but it can also be used for traditional password resets, e.g., by sending a (new, system-generated) reset password through Mercury. This scheme can also be used offline for local password recovery, which requires no participation from sites.

---

<sup>1</sup> See <http://www.mandylionlabs.com/PRCCalc/PRCCalc.htm>. Estimates vary.

<sup>2</sup> In Roman/Greek mythology, Mercury is a messenger of gods and a deity of merchandise and merchants, which we find to be suitable in the context of the proposed recovery method, as passwords are securely delivered from the server to the end-user.

## 2 New Approach: Mercury

### 2.1 Components and Threat Model

**Components.** Main components of the Mercury scheme are as follows: **(a)** User PC: The system that will be used to initiate password recovery. It must be capable of establishing a communication channel between itself and a personal mobile device (PMD) in close proximity. The system will relay the encrypted password received from the remote server to the PMD. **(b)** Server: The system that stores and will send a user's encrypted password back to her upon request. **(c)** PMD: A device capable of performing basic encryption/decryption of data, and capable of establishing a communication channel to the user PC. We use a smart-phone as a PMD. **(d)** Mercury software: Application that is used to (re)generate, and store cryptographic keys, transmit public keys to servers and decrypt data received from the user PC. We have implemented Mercury as an Android application. **(e)** Local communication channel: The channel used for transmitting the encrypted password from user PC to PMD. Candidate channels include Bluetooth, 2D barcodes such as QR codes,<sup>3</sup> or a USB cable. **(f)** Secure offline storage: A secure (physical/digital) location for storing user data necessary for re-generating cryptographic keys in case of PMD loss or upgrade.

**Table 1.** Notation

$U, S$	User and the website, respectively.
$ID_U$	User ID of $U$ at $S$ (unique in $S$ 's domain).
$P, A_U$	$U$ 's password and email address as shared with $S$ .
$M$	A key generating object selected by $U$ .
$pubU, privU$	$U$ 's public and private keys respectively, as generated from $M$ .
$\{data\}_{E_{pubU}}$	Asymmetric (public-key) encryption of $data$ using $pubU$ .
$PkGen(\cdot)$	A custom function to generate $(pubU, privU)$ using a specified source of randomness.
$MGen(\cdot)$	A function to convert data into Mercury-encoded text.
$MRet(\cdot)$	A function to recover data from Mercury-encoded text.

**Operational assumptions and threat model.** Assume that user  $U$  has a PMD running Mercury software (see Table 1 for notation).  $U$  generates a public-private key pair (encryption-only). Integrity of the public key  $pubU$  is important;  $pubU$  must be sent to website  $S$  (from the phone or PC) via a secure channel, such as HTTPS.  $U$  may verify  $pubU$  as received by  $S$  as follows:  $S$  encrypts a known object (e.g., the site URL, favicon, or site logo) using  $pubU$ , and transmits it to the user PC.  $U$  can then use her PMD to decrypt/verify the known object.

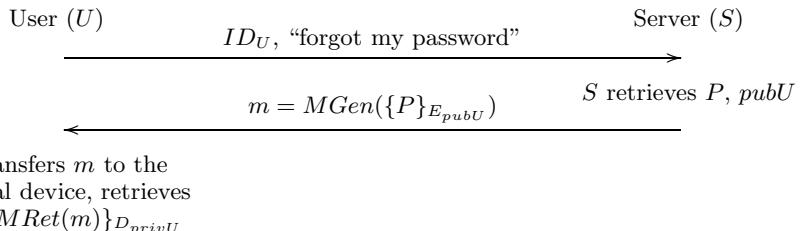
Like any password-only system, we assume that the memorized/recovered password is used only on a trustworthy machine, and the password (hash or

<sup>3</sup> A QR (Quick Response) code is a two-dimensional barcode, see developed in 1994 by a Japanese company called Denso-Wave; see <http://qrcode.com/index-e.html>.

plaintext) database on  $S$  remains beyond attackers' reach. Attackers may compromise  $U$ 's recovery email account and initiate a recovery. Email traffic is also available to attackers, and Mercury does not require email content be encrypted for password confidentiality.

## 2.2 Setup and Recovery Operation

**Account setup.** Assume that  $U$  has generated a public and private key-pair for Mercury; see Section 2.3 for key generation and backup. During account creation using a PC browser,  $U$  uploads a key file containing  $pubU$  to  $S$ .  $S$  may display the URL to upload  $pubU$  as a QR code on the browser;  $U$  then scans that URL and the phone forwards  $pubU$  to the URL. A recovery email address (as customary to many existing systems)  $A_U$  is also specified. As an additional step,  $S$  may verify both  $pubU$  (i.e., whether  $U$  can prove the possession of  $privU$  through challenge-response) and  $A_U$  (i.e., whether  $U$  can access email sent to  $A_U$ ).



**Fig. 1.** Password recovery steps

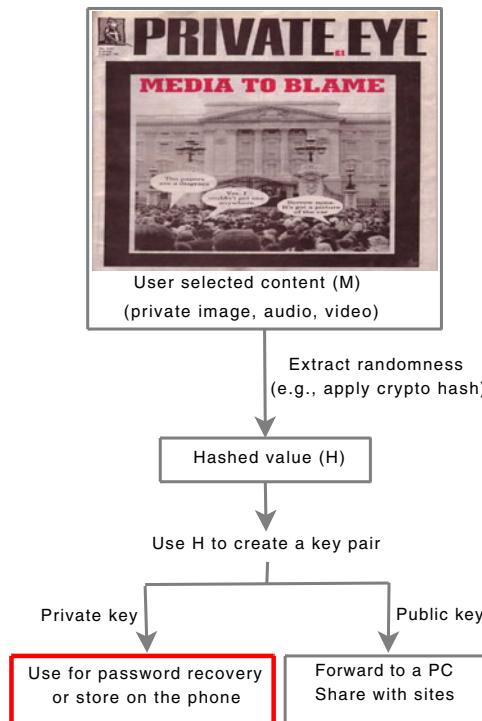
**Password Recovery Steps.** See also Fig. 1.

1.  $U$  sends her user ID  $ID_U$  and an indication to recover her password to  $S$ .
2. Using  $ID_U$  as an index,  $S$  retrieves  $U$ 's real password (or optionally assigns her a system-generated temporary password)  $P$ , and encrypts  $P$  using  $U$ 's pre-shared public key  $pubU$ . The encrypted result is then converted to a Mercury-encoded format and sent to  $A_U$  (or displayed on the PC browser).  $S$  notifies  $U$  to check her email.
3.  $U$  accesses her email and retrieves the Mercury data. The Mercury data is transferred to the PMD via the available communication channel (see Section 2). The device is now used to decrypt  $P$  (using the stored/re-generated  $privU$ ).  $P$  is then displayed on the device screen.

Note that  $U$  can use the same public key for all her services; she is not required to generate a new key-pair for each site. The key-pair is used only for encryption—no signature nor ID management are required (i.e., no need for a revocation list, or other components of a regular PKI).

### 2.3 Key Generation and Backup

We discuss here two methods for user-level public key generation, and key backup procedures. The mobile nature of today's smart-phones and other PMDs makes them more prone to loss. Losing the key pair stored on the personal device would be comparable to losing the master password in a password manager: passwords would no longer be decryptable; see also Section 3: item 3 under "Limitations." To address this issue, we have added a second-level (i.e., only used when the PMD is lost or upgraded) backup that allows the regeneration of the original key pair. Our generation and backup procedures include:



**Fig. 2.** Mercury key generation from an unshared personal file

**(a) Generate keys using entropy from a private digital object.** Mercury allows  $U$  to generate a key pair by seeding the key generator using entropy from a private object. Steps include (see also Fig. 2):

1.  $U$  selects a private object  $M$  (e.g., personal image, self-composed music) from her phone to be used for the recovery key pair generation.
2. Using a cryptographic hash function  $h$ , the Mercury program on the phone extracts randomness from  $M$ :  $H = h(M)$ .  $M$  may be truncated at some value (e.g., 1,000,000 bytes) for efficiency reasons. To gather sufficient entropy, a minimum length for  $M$  (e.g., 100,000 bytes) must be enforced.

3. The key pair is generated:  $(pubU, privU) = PkGen(H)$ .  $pubU$  is forwarded to the user PC or remote site  $S$ , but  $privU$  does not leave the phone; in fact,  $privU$  may be generated on-the-fly when needed ( $PkGen$  ensures that the same key pair is generated from the same  $H$  value, i.e., as long as  $U$  selects the same  $M$ ).

Backing up a user-selected file appears to be easier than storing keys generated from random sources. It is important to note that this method allows on-the-fly generation of keys so  $privU$  does not have to be kept on the device. The user-chosen file must also be stored offline and remain unmodified as even a single bit change may generate different keys. Using the first several hundred bytes may be avoided as some file formats store meta-data (e.g., camera-related information in an image file) in the beginning of a file; meta-data may be updated, e.g., via an image processing program. To ensure sufficient entropy for key generation, low-entropy objects must be avoided, e.g., a single-color image with all pixels having the same value. Users must also refrain from sharing  $M$  (e.g., on Flickr or Facebook) as such sharing may compromise Mercury.

**(b) Using a random seed.** Alternatively, Mercury can use a standard key generator using a random seed (i.e., without user-chosen files) and display a string to the user which contains the seed used to generate the keys (e.g., 0xB5CE69ECFFA21082430E6). If the string is given as input to the key generator, it will use that string as a seed, resulting in the same key pair being generated. The random seed in this case (as well for the object-based generation) may be manually written down or converted into a QR code, and the printed copy may serve as a physical backup.

## 2.4 Variants

**Mercury with symmetric keys only.** A symmetric-key only variant of Mercury may be called *password recovery through reverse cookies*; instead of having the server store a cookie on a client machine (as customary to the use of browser cookies), it is the client that stores a password cookie on the login server. The server sends the cookie to the client when the client requests help remembering a password.

In this mode,  $U$  generates a symmetric key ( $K = K_U$ ) with one of the methods of Section 2.3, but instead of sending the key to  $S$  during account creation,  $U$  sends an encrypted string of her password ( $\{P\}_K$ ) for the website to store (along with her email address and hashed password as usual). For password recovery,  $S$  sends  $\{P\}_K$  to  $U$ 's email, where she can use a PMD with Mercury to decode and decrypt her password using  $K$ . In this mode,  $K$  can also be generated on-the-fly from  $M$  during password recovery, or stored on the PMD. Since the same key is used for encryption and decryption,  $K$  must be kept secret.

The downside of using the symmetric key variant of Mercury is that when the user wants to update her password, a new encrypted password must be sent in parallel. Advantages of using the public key version of Mercury over this variation are discussed in Section 3.

**Alternative data channels.** While Mercury can be implemented using any type of data channel between the PC and the personal device, some channel properties are desirable. The channel should allow simple (minimal setup or need for additional software) and fast communication between devices. QR codes are a good candidate since they transfer data by simply taking a picture with the user device. The use of QR code is rapidly increasing, and is apparently more engaging for users than other channels. However, all cell-phones may not feature a camera.

Universally available audio channels on cell-phones may also be used as an alternative.  $S$  can send an encrypted password encoded as an audio file;  $U$ 's PC can play the tune and the cell-phone application can decode and decrypt the password (cf. Shazam<sup>4</sup>). As mentioned previously, Bluetooth or USB can also be used as a data channel, but require additional setup such as device pairing and carrying cables which may reduce convenience.

**Variation with PMD as primary device.** If a PMD is used as a primary email communication device (which is already common for many users) and the recovery email is accessed directly from the PMD, then neither the PC nor the PC-to-PMD channel in the basic Mercury protocol is required. Here text messages (SMS) may also be used, instead of email, to send the encrypted password directly to a smart-phone.

**Variation without a PMD.** We have also considered a version of Mercury which operates without a PMD by generating and storing keys on the PC exclusively. To avoid the loss of portability and the requirement of storing long-term keys on the PC, keys or key-generating objects may be stored on mobile storage, e.g., USB flash drives, albeit at the risk of introducing different problems (e.g., lost/stolen drives).

**Storing passwords: hashed, cleartext, or encrypted.** Mercury is transparent to how site maintainers store user passwords (cleartext, hashed, or encrypted). Security proponents recommend that only hashes of user passwords be stored to prevent direct (e.g., without running a dictionary attack) reuse of passwords in case the (hashed) password list is compromised. Password recovery requires access to the original cleartext password, but Mercury can also work without access to it. For example, encrypted temporary passwords may be sent to PMD, recovered and used to reset a password; see also the symmetric-key variant of Mercury in Section 2.4. Alternatively, a site can store the hashed password, user public key, and public-key encrypted password (instead of storing the cleartext password). The encrypted password is then forwarded when requested. Note that password update in this case is affected by whether the public-key encryption is done on the server side or at the client browser. If encryption is performed at the server side (i.e., the site receives the initial cleartext password), during password update, the site can replace the old encrypted password with the updated one, without requiring any extra step from the user. For browser-side

---

<sup>4</sup> A music discovery tool for cell-phones, see: <http://www.shazam.com/>. We acknowledge N. Asokan for proposing the use of audio channel in Mercury.

encryption (i.e., the site never receives cleartext password), the user must have the public key available during the password update.

### 3 Features and Limitations

In this section, we list several features and limitations of Mercury.

#### Advantages and features.

1. There is no need to trust third-parties in Mercury. Just as in the current practice,  $P$  is shared only between  $U$  and  $S$ .
2. If an encrypted password is compromised, e.g., by a malicious ISP or email provider, attackers cannot run an offline dictionary attack as  $P$  is encrypted with  $pubU$  (for RSA, using the EME-OAEP padding as defined in PKCS #1 v2.1 [11]).
3. Updating a password remains the same as current strategies; users resort to Mercury only when a password recovery is needed, and in the public-key variant do not require any extra step to “sync” their updated passwords (as opposed to traditional password managers).
4. Users can easily replace their personal device with a new one. However, the key-generating file must be deleted from the old device and copied to the new one; this step is simpler when the file is stored on a removable memory card. Also, the Mercury program must be installed on the new device. Note that the key regeneration technique using a user-chosen private object may simplify several user-level key management issues such as key transfer and backup. Such public keys without a PKI might enable easier adoption of other security applications including PGP, and authentication (cf. [22], [12]).
5. Mercury may be used for password recovery in a local PC. Some consumer systems, e.g., Mac OS X, Ubuntu Linux, offer users to setup a “password hint” (a text string that may help  $U$  to recall her login password). If  $pubU$  is stored on the local PC, then the login procedure may be modified to enable Mercury as follows. When  $U$  indicates that she lost her password, a temporary password is encrypted with  $pubU$ , and displayed as a QR code.  $U$  can use her smart-phone to retrieve the password. Note that most current operating systems store only password hashes, thus requiring the use of temporary password in this case. Alternatively, the original password may be stored encrypted under  $pubU$ .
6. Once the public key  $pubU$  is stored on  $S$ ,  $S$  can securely communicate additional information to  $U$ . For instance, a banking site could enable two-factor authentication by requesting the user’s password, and sending a one-time password encrypted with  $pubU$ .  $U$  decodes the one-time password and submits it back to  $S$ , proving that she is in possession of the corresponding private key  $privU$ .

#### Limitations.

1. Mercury requires service providers’ assistance for online deployment. However, some sites may be reluctant to implement Mercury as not all users

possess an additional personal device, or would be willing to use this service. To deal with such cases, Mercury can be deployed gradually, e.g., as an option so that users who want to take advantage of it, can do so without imposing any requirements on others.

2. Users must carry their device for portability (i.e., password recovery from anywhere). However, users can continue regular login without Mercury.
3. To deal with a lost or stolen personal device, users must keep a “secure” backup of the personal object or random seed used for the key generation; e.g., storing the object on a USB key and keeping it private, see Section 2.3. If there is no backup,  $U$  must generate a new key-pair, login to all Mercury-enabled sites (using recalled passwords), and then upload the new public key. Note that losing the keys or not having access to the PMD do not necessarily restrict login, as long as  $U$  has access to her primary login password.
4. Users may willingly or accidentally upload the key-generating private object to public sites such as Facebook. Making users understand the risk of such actions appears non-trivial, especially for users who want to share all their digital content with both friends and strangers. However, this risk is reduced by using a random seed for generating keys and storing the seed offline, e.g., as a printed QR code (see Section 2.3, item (b)).

**Smart-phone compromise.** If the smart-phone or other personal device is infected with malware, the user’s private key might become available to an attacker. However, the key alone may not be enough to gain access to user accounts. The attacker needs access to both the key and the encrypted password sent during password recovery. Recall that the password is sent to  $U$ ’s email address, which would require an attacker to also obtain the email account password. If the recovery email is accessible from the compromised phone (e.g., logged in email client on the phone), attackers may be able to compromise passwords for all Mercury-protected accounts of a user. Operating system security mechanisms such as built-in secure storage facilities and file system isolation might help lower the risk of this attack. For the case of physical device theft, we recommend the use of a PIN-lock<sup>5</sup> on the personal device.

## 4 Prototype Implementation on Android

Mercury has been implemented as a proof of concept application for Android. Virtually all Android phones are equipped with cameras, network connectivity, and libraries for performing the basic functions of Mercury (generating key pairs, RSA public key encryption/decryption, and data transmission over HTTP/HTTPS). Thus, we use QR Codes as the data channel to transmit encrypted passwords from the user PC to the smart-phone. The prototype consists of two parts: the client, an Android application; and the server, a demo of a normal usage scenario for Mercury. The Android application is also capable of

---

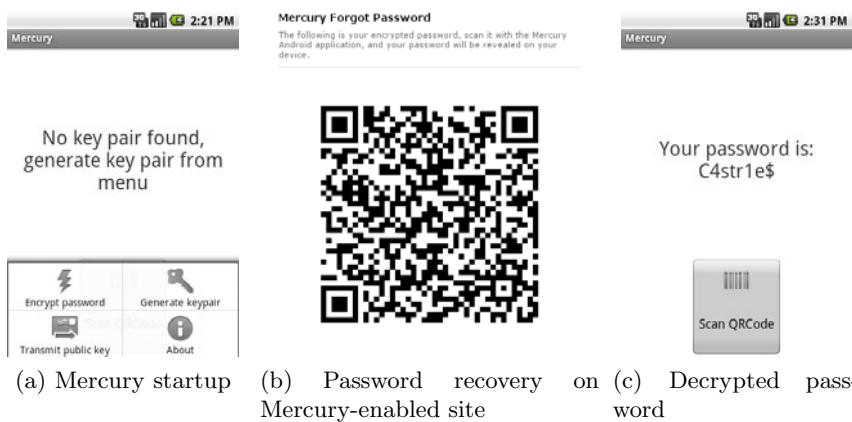
<sup>5</sup> Apple iOS and other smart-phone platforms allow the user to specify a number of password entry attempts after which to format the device memory.

operating in a standalone way without a server-side component, enabling users to immediately benefit from Mercury. We describe the implementation details below, for an Android client.

**Key Pair Creation.** The Bouncy Castle Crypto API ([bouncycastle.org](http://bouncycastle.org)) is used to generate a 1024-bit RSA key pair ( $pubU, privU$ ). Keys are generated using Java `SecureRandom` as the source of entropy, and stored with the PKCS#1 Optimal Asymmetric Encryption Padding (OAEP) option [11]. OAEP helps prevent guessing attacks on the encrypted string. The key pair is stored on the device using the Android data storage, saving files into a dedicated per application directory [2]. By default, this directory is accessible only to the given application (enforced through standard UNIX file permissions, as each application runs with a different user ID). Upon launch of the application for the first time, users are instructed to generate a key pair from the application menu.

**Generating an Encrypted Password.** Once a key pair has been created,  $U$  can proceed to either create an account on a web server that supports Mercury password recovery, or create new Mercury encrypted password for personal use. In the latter case,  $U$  selects the “Encrypt password” menu option (Fig. 3a), and inputs a password  $P$ .  $P$  is then encrypted with  $pubU$ , and the result is encoded to Base64 before being converted to a QR code ( $qr = MGen(\{P\}_{E_{pubU}})$ ). Base64 encoding is necessary to create a valid QR code of ASCII characters. The application can email  $qr$  to  $U$ , which can be printed or stored for future use.

**Scanning a Password for Recovery.** A QR code of the encrypted password is scanned using the ZXing<sup>6</sup> (“Zebra Crossing”) barcode reader library for Android. If ZXing is not installed, the Mercury app prompts  $U$  and opens the Android Market page for the Barcode Reader application that provides the ZXing service. The scanned QR code will be decoded from Base64 and then decrypted so that  $\{MRet(qr)\}_{D_{privU}} = P$ ; see Fig. 3.



**Fig. 3.** User interface for password recovery

<sup>6</sup> <http://code.google.com/p/zxing>

**Transmitting public keys to a server.** We have implemented a feature to transmit the generated public key directly from the phone to a server. This feature works by having the user scan a QR code containing the URL of the server and a nonce. The Mercury application then generates a POST request to the scanned URL that includes the public key  $pubU$ . The key is saved into  $U$ 's account on the server.

**Web server.** Our prototype server is a typical “Signup/Login/Forgot password” system using PHP and MySQL.  $U$  creates a new account; the server  $S$  stores  $(ID_U, P)$ , and then prompts  $U$  to send her public key.  $S$  verifies that the nonce is valid, and stores  $pubU$  in a database.

In a typical use scenario,  $U$  attempts to log in, and realizes she has forgotten  $P$ . Clicking the “forgot password” link after specifying  $ID_U$  emails her a QR code generated by encrypting  $P$  with  $pubU$ . Alternatively the QR code can be displayed immediately on the screen.  $U$  then uses her phone to scan the QR code and retrieve  $P$ .

**Limitations of the prototype.** Our prototype does not use a secure application data storage, such as the Android Credential Storage, available in versions 1.6 and higher. Despite this, the storage we used has UID/GID protection mechanisms offered by all versions of Android, which prevents the private key from being accessed by other applications. Future versions of Mercury will make use of the secure storage facility to reduce the risk of private key leaks.

Mercury currently uses 1024-bit RSA keys and the key length is hardcoded in this prototype. We have tested Mercury on Nexus One phones, and operations such as key generation, public key sharing and password recovery were completed without any noticeable delay from a user's perspective. During testing, we found that 2048-bit keys add a small but non-perceptible delay to Mercury operations on the Nexus One. Older devices with slower processors may exhibit a more noticeable delay with larger key sizes.

## 5 Existing Approaches, Related Work and Comparison

### 5.1 Existing Password Recovery Approaches

Below we discuss common techniques used for password recovery and managing multiple passwords.

**Password Managers.** There are two types of common password managers: local and remote/third-party-supported. In the first case, passwords (user-chosen or system generated) are stored on the user's PC (as a file managed by the password manager application, or integrated with a web browser), or on a PMD. It is generally recommended that the password file must be encrypted with a key derived from (or protected with) a password, called the master password. However, most users do not use such passwords, as with the built-in browser password managers [20]. The master password is susceptible to offline attacks if a copy

of the encrypted password file can be obtained. From a usability perspective, several issues may arise: losing the master password may result in losing access to many accounts; the local password file must be backed up; the file must also be carried with the user for portability (unless it is already on a PMD); and if a password is updated at a site, the user must also update it in the manager.

When the password file is stored online with a third-party, users must trust the service provider to be honest, even if the password file is sent encrypted from a user machine. The remote party can be actively malicious, or become compromised; strong incentives exist for attackers as such a compromise may enable access to many accounts per user.

**Email Password Recovery/Reset.** An email address is registered at the time of account creation; when a user provides her user ID and indicates that she has lost the account password, the registered email is used for sending the original password, or a temporary password/URL to reset the account password. As long as users can access the email account, this solution is portable. It is also quite instant in today's Internet, as email delivery is close to real-time. Because of low-cost deployment, and the ubiquitous use of email, email-based recovery/reset is most commonly used. One password survey [4] reports that about 92% of sampled websites offer an email-based solution, with 44% sending a reset link in the email, 32% sending a temporary (reset) password, and 24% sending the original cleartext password.

The unencrypted nature of email enables the email provider (or someone compromising the provider) to learn all reset links and passwords. Any intermediate party (e.g., ISPs, wireless access points) between the user and email server can also access the reset emails when sent via an unencrypted channel such as regular HTTP webmail as provided by Yahoo and Hotmail sites; optional HTTPS-protected sites are also not always immune to attacks, see [9]. Other pitfalls include [8]: recovery emails being classified as spam, and an exposed cleartext passwords may be exploited at other sites due to password reuse.<sup>7</sup> As email is being used for increasingly important authentication tasks (e.g., domain registration and management), compromised accounts can result in costly consequences [1].

**Other Methods.** Preregistered questions/answers are sometimes used to reset an account password. Concerns include: (i) users may forget answers to these questions; and (ii) these answers may be easier to guess than a user-chosen password (see e.g., [16], [19]). Few other solutions are deployed in practice. Google offers sending a reset code as an SMS to a pre-registered cell-phone number. eBay allows three methods for reset: answering personal profile information (e.g., postal code, phone number, and date of birth), sending reset code via an automated call-back to a phone number, and instant messaging based live help on the eBay site.

---

<sup>7</sup> One survey [21] reported that 73% of online banking users share their banking password to access at least another less sensitive account.

## 5.2 Comparison: Mercury vs. Current Approaches

Table 2 compares Mercury to other password recovery/reset approaches. For the feature comparison and analysis listed below, we assume Mercury's primary mode of operation based on a private-public key pair.

**Table 2.** Comparison of Mercury to other password recovery/reset methods

	Independence of 3 <sup>rd</sup> party trust	Portable	Recovers original password	Overhead for account creation	Transparent to password updates
Pass. managers (online)	No	Yes	Yes	High	No
Pass. managers (offline)	Yes	No	Yes	High	No
Email	No	Yes	No <sup>8</sup>	Low	N/A
PVQs	Yes	Yes	No	High	N/A
Mercury	Yes	Yes	Yes	Low	Yes

**Independence of 3<sup>rd</sup> party trust.** Online password managers and email require the user to trust the transmission channel between the server and the PC, as well as ISPs and email providers. Offline password managers require no third-party trust, but the user must manually keep track of all accounts in a password file. Mercury does not require trust in third-party sites, communication channels, or service providers.

**Portability.** Mercury offers portability if the user is in possession of the personal device at the time of password recovery. Other recovery/reset systems listed have some degree of portability, but may require the user to carry a password file at all times (e.g., offline password managers). This may reduce portability and increase inconvenience for the user.

**Restoring original password.** PVQs do not typically allow the user to obtain the original password, but rather allow the user to set a new password after successfully answering verification questions. Email based recovery/reset systems frequently send the user a link for password reset, or a one-time temporary password. Password managers, as well as Mercury allow the user to retrieve the original password.

**Overhead for account creation.** Password managers and PVQs must be updated or configured every time a new account is created. Overhead is small if there is an automatic way to keep track of new passwords (e.g., browser-based password managers). Mercury requires the user to send the public key to the server, also adding an extra step to account creation; this step can be implemented (e.g., as in our Android prototype) to have minimal time overhead.

**Transparency to password updates.** When a password is updated, the corresponding account in a password manager must be updated as well. Similar to

<sup>8</sup> Most sites email reset links/passwords as opposed to original passwords [4].

the account creation phase, some password managers might detect password updates and store them automatically. Mercury, by design, offers full transparency to password updates in the public key mode, allowing users to update their passwords as usual without additional steps.

### 5.3 Other Related Work

Here we discuss other related proposals involving user-level public keys and online authentication. Existing techniques for password recovery are discussed in Section 5.1.

Seeing is Believing (SiB) [14] uses QR codes to transfer a public key from one user device to another (equipped with a camera) as part of authenticated key establishment between devices with no previously shared context. The SiB proposal is used mainly for authenticating two devices in close proximity. Mercury relies on an existing authentication channel, and is used only when credentials are forgotten.

Snap2Pass [5] uses QR codes and a camera phone to implement a challenge-response protocol for authentication; this is envisioned as a replacement for password-based authentication. In contrast to replacing passwords, which are currently deeply entrenched into the web ecosystem, the focus of Mercury is to facilitate and simplify secure password recovery.

Some password managers, e.g., PwdHash [18] generate site-specific passwords from a single, user-chosen password. If adopted widely, these may help reduce the need for password reset as users are expected to remember only one master password. However, the master password is vulnerable to offline dictionary attack, if a site-specific password and the site URL is known (e.g., in a phishing attack). Forgetting the master password results in denied access to all PwdHash-protected accounts; access may be regained via resetting all such passwords.

Mercury does not aim to replace passwords as used. It is targeted towards password recovery, and as such, it is designed for infrequent use. It employs user-level public keys, which can be reconstructed from user-chosen digital objects. No new private or secret keys are shared between the phone and website.

## 6 Concluding Remarks

Password recovery and reset techniques are as old as passwords themselves. Before the Internet, such techniques were relatively simple and secure; e.g., users could talk to an administrator in person and reset their passwords. In current large-scale enterprise environments, and online sites with millions of users, existing password recovery techniques are inadequate and costly in terms of security, usability and expense. The proposed online password recovery technique – Mercury – avoids limitations of current password managers (both online and offline), email reset methods and PVQs.

In contrast to traditional approaches which require users to create a new password when the old one is forgotten, Mercury allows what may seem like

a small, but we believe important paradigm shift: recovering an existing password in a secure manner. Mercury's security is independent of plaintext email or the compromise of a "master password" in password managers. Mercury is also transparent to the server-side password storage method, as well as to the communication channel between the PC and the personal device. In the current implementation of Mercury, we take advantage of widely available Android smart-phones as a PMD, and increasingly-used 2D barcodes as an engaging communication channel between the PC and the phone. The Mercury Android application and a demonstration site are available for public testing at: <http://www.ccsl.carleton.ca/software/mercury/>.

**Acknowledgments.** We thank N. Asokan and anonymous referees for their helpful comments. The first author is supported by an NSERC PDF. The last author acknowledges NSERC funding under a Discovery Grant and as Canada Research Chair in Authentication and Computer Security. Partial funding from NSERC ISSNet is also acknowledged.

## References

1. Ahmad, D.: The confused deputy and the domain hijacker. *IEEE Security and Privacy* 6(1) (2008)
2. Android Open Source Project. Data storage (android developers), <http://developer.android.com/guide/topics/data/data-storage.html>
3. BBCNews.com. Obama Twitter account hacked by Frenchman(March 24, 2010), <http://news.bbc.co.uk/2/hi/8586269.stm>
4. Bonneau, J., Preibusch, S.: The password thicket: Technical and market failures in human authentication on the web. In: Workshop on the Economics of Information Security (WEIS 2010), Cambridge, MA, USA (June 2010)
5. Dodson, B., Sengupta, D., Boneh, D., Lam, M.S.: Secure, consumer-friendly web authentication and payments with a phone. In: Conference on Mobile Computing, Applications, and Services (MobiCASE 2010), Santa Clara, CA, USA (October 2010)
6. Ellison, C.M., Hall, C., Milbert, R., Schneier, B.: Protecting secret keys with personal entropy. *Future Generation Computer Systems* 16(4) (February 2000)
7. Florêncio, D., Herley, C., Coskun, B.: Do strong web passwords accomplish anything? In: USENIX Workshop on Hot Topics in Security (HotSec 2007), Boston, MA, USA (August 2007)
8. Garfinkel, S.: Email-based identification and authentication: An alternative to PKI? *IEEE Security and Privacy* 1(6) (2004)
9. Guardian.co.uk. Gmail ups security after Chinese attack. News article (January 13, 2010), <http://www.guardian.co.uk/technology/2010/jan/13/gmail-increases-security-chinese-attack>
10. Jakobsson, M., Stolterman, E., Wetzel, S., Yang, L.: Love and authentication. In: Conference on Human Factors in Computing Systems (CHI 2008), Florence, Italy (April 2008)
11. Jonsson, J., Kaliski, B.: Public-key cryptography standards (PKCS) #1: RSA cryptography specifications version 2.1. RFC 3447, Category: Informational (February 2003)

12. Lopez, J., Oppliger, R., Pernul, G.: Why have public key infrastructures failed so far. *Internet Research* 15(5) (2005)
13. Mannan, M., van Oorschot, P.: Digital objects as passwords. In: USENIX Workshop on Hot Topics in Security (HotSec 2008), San Jose, CA, USA (July 2008)
14. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-is-believing: Using camera phones for human-verifiable authentiction. *Security and Networks* 4(1–2) (2009)
15. Mitnick, K., Simon, W.L.: *The Art of Deception*. Wiley (2002)
16. Rabkin, A.: Personal knowledge questions for fallback authentication. In: Symposium on Usable Privacy and Security (SOUPS 2008), Pittsburgh, USA (July 2008)
17. Renaud, K., Just, M.: Pictures or questions? Examining user responses to association-based authentication. In: British HCI Conference, Dundee, Scotland (September 2010)
18. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J.C.: Stronger password authentication using browser extensions. In: USENIX Security Symposium, Baltimore, MD, USA (2005)
19. Schechter, S., Brush, A.J.B., Egelman, S.: It's no secret. Measuring the security and reliability of authentication via 'secret' questions. In: IEEE Symposium on Security and Privacy (May 2009)
20. Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., Vigna, G.: Your botnet is my botnet: Analysis of a botnet takeover. In: ACM Computer and Communications Security (CCS 2009), Chicago, IL, USA (November 2009)
21. Trusteer.com. Reused login credentials. Security advisory (February 2, 2010), <http://www.trusteer.com/sites/default/files/cross-logins-advisory.pdf>
22. Whitten, A., Tygar, J.D.: Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In: USENIX Security Symposium, Washington, D.C, USA (1999)
23. Wired.com. Palin e-mail hacker says it was easy (September 18, 2008), <http://www.wired.com/threatlevel/2008/09/palin-e-mail-ha/>
24. Zhang, Y., Monroe, F., Reiter, M.: The security of modern password expiration: An algorithmic framework and empirical analysis. In: ACM Computer and Communications Security (CCS 2010), Chicago, IL, USA (October 2010)
25. Zviran, M., Haga, W.J.: Cognitive passwords: The key to easy access control. *Computers & Security* 9(8) (1990)
26. Zviran, M., Haga, W.J.: A comparison of password techniques for multilevel authentication mechanisms. *Computer Journal* 36(3) (1993)

# Author Index

- Ahmad, Junaid Jameel 235  
Algasinger, Michael 190  
Anderson, Ross 220  
  
Balzarotti, Davide 284  
Barrera, David 315  
Böhme, Rainer 1  
Bond, Mike 220  
Brown, Carson D. 315  
Bubendorfer, Kris 85  
  
Camp, L. Jean 102  
Carbunar, Bogdan 127  
Chia, Pern Hui 299  
Choudary, Omar 220  
Christin, Nicolas 16  
Clark, Jeremy 47  
  
Dagon, David 93  
Deiseroth, Björn 276  
Duan, Shanshan 113  
  
Egelman, Serge 16  
  
Franz, Martin 127, 276  
Freudiger, Julien 31  
  
Goldberg, Ian 158  
Grossklags, Jens 16  
  
Haenni, Rolf 182  
Halderman, J. Alex 77  
Hamacher, Kay 276  
Heisrath, Sören 235  
Hengartner, Urs 47  
Hopper, Nicholas 268  
Hubaux, Jean-Pierre 31  
  
Jaeger, Christian 190  
Jha, Somesh 276  
Johnson, Rob 141  
  
Katzenbeisser, Stefan 127, 276  
Kesdogan, Dogan 62  
Kirda, Engin 284  
Knapskog, Svein Johan 299  
Koenig, Reto 182  
  
Lamb, Michael 141  
Levchenko, Kirill 260  
Li, Ninghui 102  
Li, Shujun 235  
Lie, David 315  
Liu, Debin 102  
Lofgren, Peter 268  
  
Mannan, Mohammad 315  
Marquardt, Philip 93  
McCoy, Damon 260  
Meagher, Kenneth 77  
Morales, Jose Andre 260  
Murdoch, Steven J. 220  
  
Nohl, Karsten 205  
Novak, Jay 77  
  
Olumofin, Femi 158  
  
Palmer, Ben 85  
Peter, Andreas 127  
Plötz, Henryk 205  
Pötzsch, Stefanie 1  
  
Rührmair, Ulrich 190  
  
Sadeghi, Ahmad-Reza 235  
Safavi-Naini, Reihaneh 173  
Schläpfer, Michael 182  
Schmitz, Roland 235  
Scholte, Theodoor 284  
Schröder, Heike 276  
Shokri, Reza 31  
Sion, Radu 127  
Soghoian, Christopher 250  
Sotakova, Miroslava 127

- Spycher, Oliver 182  
Stajano, Frank 220  
Stamm, Sid 250  
Stribley, Jonathan 77  
Tan, Chik How 113  
Traynor, Patrick 93  
Tuhin, Mohammed Ashraful Alam 173  
van Oorschot, Paul C. 315  
Vidas, Timothy 16
- Walsh, Leif 141  
Wang, Huaxiong 113  
Wang, XiaoFeng 102  
Welch, Ian 85  
Westermann, Benedikt 62  
Williams, Peter 127  
Wong, Duncan S. 113  
Yang, Guomin 113