Ahmad-Reza Sadeghi (Ed.)

LNCS 7859

Financial Cryptography and Data Security

17th International Conference, FC 2013
Okinawa, Japan, April 2013
Revised Selected Papers

Springer

Lecture Notes in Computer Science 7859

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Ahmad-Reza Sadeghi (Ed.)

Financial Cryptography and Data Security

17th International Conference, FC 2013
Okinawa, Japan, April 1-5, 2013
Revised Selected Papers

Springer

Volume Editor

Ahmad-Reza Sadeghi
Technische Universität Darmstadt/CASED
Mornewegstraße 30
64293 Darmstadt, Germany
E-mail: ahmad.sadeghi@trust.cased.de

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-39883-4 e-ISBN 978-3-642-39884-1
DOI 10.1007/978-3-642-39884-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013943690

CR Subject Classification (1998): E.3, K.6.5, K.4.4, C.3, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

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

Printed on acid-free paper

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

Preface

This volume contains the proceedings of the of the 17th International Conference on Financial Cryptography and Data Security (FC), held at Bankoku Shinryokan Busena Terrace Beach Resort in Okinawa, Japan, during April 1–5, 2013. For more than 17 years, FC has been the primary forum where international experts from academia, industry, and government present, debate, discuss, and advance the security and privacy aspects of commercial and financial systems. In the past, many successful companies have presented their very early ideas at FC.

This year, we assembled a diverse program of 14 regular papers and 17 short papers, selected by the Program Committee from the 125 submissions by authors from 32 countries, resulting in an acceptance rate of 12.5% for regular papers. All submissions received at least three reviews from the 49 Program Committee members chosen by the Program Chair or the 70 external reviewers. The conference was opened with a keynote by William Saito, founder of InTecur, council member on national strategy and policy for the National Policy Unit, and Chief Technology Officer of the Fukushima Nuclear Accident Independent Investigation Commission (NAIIC). In his talk "Can Nature Help Us Solve Risk Management Issues?" he presented intriguing insights into what we can learn from nature and evolution to protect IT systems. The second keynote was given by N. Asokan, Distinguished Researcher, formerly at the Nokia Research Center and now professor at the University of Helsinki. His exciting talk "The Untapped Potential of Trusted Execution Environments on Mobile Devices" showed the rich potential of security hardware (trusted execution environments) in mobile phones and the next-generation of mobile technology. The panel discussion was on "The State of the Art in e-Banking Security and Usability." In the co-located workshops on usable security and homomorphic encryption, a variety of new approaches and studies were presented and discussed.

The highlights of this year's conference were timely topics such as Bitcoin, which is a recently proposed virtual currency concept allowing P2P transactions, and the current and the next-generation of e-banking systems as well as usable security and privacy. A hot topic of several talks concerned the analysis of various aspects of Bitcoin, which seems to have become very popular and is the most successful electronic payment system to date with more than 300 million USD in electronic coins. Bitcoins continue to be purchased and their value is rapidly growing.

I thank the General Chair Kazue Sako from NEC, Japan, for her dedicated work and the excellent local organization of the conference, all the authors for the numerous submissions, and all the Program Committee members and the external reviewers for contributing their expertise to the selection of the papers for the program. Without their service and contribution, setting up such a conference would have been impossible. Further, I would like to acknowledge the

members of the International Financial Cryptography Association (IFCA) board of directors for their continuous effort. Finally, I thank all sponsors (NEC, ONR Global, AIST, IPA, NICT, Google and WorldPay) for supporting the conference.

May 2013 Ahmad-Reza Sadeghi

Organization

Program Committee

Alessandro Acquisti	Carnegie Mellon University, USA
Ross Anderson	Cambridge University, UK
Rainer Boehme	University of Münster, Germany
Jens Bohli	NEC Laboritories Europe, Germany
Colin Boyd	Queensland University of Technology, Australia
Liqun Chen	Hewlett-Packard Laboratories, UK
Sherman Chow	New York University, USA
Nicolas Christin	Carnegie Mellon University, USA
Reza Curtmola	New Jersey Institute of Technology, USA
George Danezis	Microsoft Research Cambridge, UK
Emiliano De Cristofaro	PARC, USA
Loic Duflot	French Central Directorate for Information Systems Security, France
William Enck	North Carolina State University, USA
Bao Feng	Institute for Infocomm Research, Singapore
Jens Grossklags	Penn State University, USA
Xuxian Jiang	North Carolina State University, USA
Ari Juels	RSA Laboratories, USA
Stefan Katzenbeisser	TU Darmstadt, Germany
Angelos Keromytis	Columbia University, USA
Florian Kerschbaum	SAP Research, Germany
Aggelos Kiayias	University of Connecticut, USA
Yuichi Komano	Toshiba Corporation, Japan
Kari Kostiainen	ETH Zrich, Switzerland
Farinaz Koushanfar	Rice University, USA
Xuejia Lai	Shanghai Jiaotong University, China
Jiangtao Li	Intel Corporation, USA
Benoit Libert	Universite Catholique de Louvain, Belgium
Di Ma	University of Michigan-Dearborn, USA
Mark Manulis	University of Surrey, UK
Kanta Matsuura	University of Tokyo, Japan
Atsuko Miyaji	Japan Advanced Institute of Science and Technology, Japan
Refik Molva	EURECOM, France
Toru Nakanishi	Okayama University, Japan
Satoshi Obana	Hosei University, Japan
Claudio Orlandi	Bar-Ilan University, Israel
Josef Pieprzyk	Macquarie University, Australia

Benny Pinkas	University of Haifa, Israel
Bart Preneel	Katholieke Universiteit Leuven (COSIC), Belgium
Ahmad-Reza Sadeghi	TU Darmstadt, Germany
Thomas Schneider	TU Darmstadt, Germany
Jamshid Shokrollahi	Robert Bosch GmbH, Germany
Matthew Smith	Leibniz Universität Hannover, Germany
Keiji Takeda	Keio University, Japan
Isamu Teranishi	NEC Corporation, Japan
Patrick Traynor	Georgia Institute of Technology, USA
Ersin Uzun	PARC, USA
Michael Wiener	Irdeto Canada, Canada
Akira Yamada	KDDI R&D Labs, Japan

Additional Reviewers

Amrutkar, Chaitrali
Androulaki, Elli
Argyros, George
Asghar, Hassan
Athanasopoulos, Elias
Azraoui, Monir
Bangerter, Endre
Boggs, Nathaniel
Breuker, Dominic
Carter, Henry
Choi, Seung Geol
Chow, Richard
Chow, Sherman
Chu, Cheng-Kang
Davi, Luca
De Cristofaro, Emiliano
Diaz, Jesus
Ding, Yi
Dmitrienko, Alexandra
Duplys, Paul
Elkhiyaoui, Kaoutar
Emura, Keita
Faber, Sky
Fischlin, Marc
Furukawa, Jun
Griffin, Robert
Hang, Isabelle
Hayashi, Ryotaro
Hazay, Carmit
Huang, Yun
Jawurek, Marek
Johnson, Benjamin
Katzenbeisser, Stefan
Kemerlis, Vasileios P.
Kontaxis, Georgios
Laguillaumie, Fabien
Leontiadis, Iraklis
Li, Xiangxue
Luhn, Sebastian
Manabe, Yoshifumi
Matsuda, Takahiro
Nikova, Svetla
Nochenson, Alan
Omote, Kazumasa
Önen, Melek
Papamanthou, Charalampos
Pashalidis, Andreas
Peeters, Roel
Perl, Henning
Peters, Thomas
Polychronakis, Michalis
Portokalidis, Georgios
Reaves, Brad
Rudolph, Larry
Samari, Katerina
Schroepfer, Axel

Seonghan, Shin
Sepehrdad, Pouyan
Seuken, Sven
Seurin, Yannick
Shao, Jun
Smith, Matthew
Suzuki, Kotaro
Tang, Qiang
Tselekounis, Yiannis
Wu, Yongdong
Xu, Hong
Yavuz, Attila Altay
Yoneyama, Kazuki
Yuen, Tsz Hon
Zhang, Haibin
Zohner, Michael

Table of Contents

Passwords

Privacy Primitives and Non-repudiation

Anonymity

Hardware Security

Secure Computation and Secret Sharing

Invited Talk

Authentication Attacks and Countermeasures

Privacy of Data and Communciation

Private Data Retrieval

Posters

Can Nature Help Us Solve Risk Management Issues?

William H. Saito

William@saitohome.com

Abstract. As a member of the commission that investigated the Fukushima (Japan) nuclear disaster and studying other catastrophes over the past century, it was discovered that all were man made and preventable; all resulted from a lack of understanding of risk and/or a refusal to accept numerous warnings and risk assessments. The lessons of Fukushima show clearly that true security planning is not a quest for absolutes (100 percent reliability), but a flexible response to the inevitability of system failures. One of the best approaches to understanding and modeling IT security is to begin with a deep understanding of biological processes in Nature. Because many contemporary security problems have analogues in the natural world, effective solutions to these problems may already exist. By ignoring them we are trying to reinvent the wheel.

Keywords: Nature, Risk, Resilience, Evolution, Biology, Systems, Fukushima.

A long time ago, towards the end of the 20th century, the software company that I founded in California developed its own suite of security applications and solutions, so it was only natural for us to study all the commercially available security tools on the market. Soon we were digging deeper into the science behind authentication, encryption and more. As our business grew, we developed a set of biometric interface standards that Microsoft ultimately adopted as part of the Windows operating system. Along this journey my interest in and knowledge about information security deepened, and I found myself advising both public- and private-sector organizations. But it was only a few years ago that I sat down and began writing about encryption, authentication, digital signatures and fairly technical aspects of cryptography as well as other esoteric aspects of security.

Then came 3.11. No, not 9.11, but March 11, 2011, the day a massive earthquake and tsunami ripped across the northeastern coast of Japan. It was only then that the life-and-death importance of risk management and its profound implications for all types of security became apparent. Terrible as the natural devastation was, the tsunami precipitated an even more terrifying event, leading to the near-destruction of the Fukushima Dai-ichi Nuclear Power Plant.

Later that year, I was appointed the Chief Technology Officer for the Fukushima Nuclear Accident Independent Investigation Commission[1], an ad hoc body reporting to the national legislature (it was, in fact, the first independent investigation ever commissioned by the National Diet of Japan). That position provided a unique opportunity to examine this catastrophe in detail and to see its multiple causes. Looking at

[1] For more information about the disaster, its causes and consequences, see The National Diet of Japan – Fukushima Nuclear Accident Investigation Commission (NAIIC) home page: http://naiic.go.jp/

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 1–5, 2013.

the long chain of errors and misjudgments that led up to Fukushima naturally brought my thinking around to the idea of security and risk management. How could the risks not have been more widely foreseen? How could the management of that risk have been so inept? What steps were taken after the accident to limit risk and what steps were put in place subsequently to prevent a similar situation from occurring?

One of my tasks as CTO was to develop a secure IT system that would allow the members of the new Commission to share and store information, ideas, notes and comments freely and flexibly without the danger of outsiders gaining access to those private discussions. How could over a hundred participants from various regions of Japan, and with widely varying levels of computing proficiency, securely conduct a sensitive investigation without accidentally or intentionally leaking information by, say, losing a personal computer or being subject to a hacker attack? Thankfully, because the IT system was designed from the ground up with security planning as a primary directive, the Commission was able to conduct its work over a period of months and publish its findings without any information being manipulated, destroyed or leaked.

When the post-3.11 world finally began to settle down, my perspective had changed and I now viewed the whole field of IT security through the broader lens of risk management. I had never liked talking about security from a purely technical perspective, which by definition misses the big picture, and after 3.11 my feelings grew even stronger. My experiences studying the Fukushima disaster and managing security for the Commission further reinforced my belief that "doing IT security" simply by designing sophisticated new authentication systems or cryptography algorithms was not the right approach. It misses the critical component of risk management. As the old adage says, security is only as strong as the weakest link in the chain. Real security, and not only in the world of IT, lies in maximizing your field of view, expanding your thinking, for example, by canvassing the natural world for useful examples, and even expanding your imagination to encompass what has never yet existed.

The catalyst for my change in perspective began a month before the start of the commission. Not being a nuclear safety or risk management expert, I took it upon myself to study all the historically significant disasters. Thus, I spent most of my yearend holiday reading over 4,000 pages of reports on disasters as varied as the Titanic, Challenger, Three Mile Island, Chernobyl, Concord, BP, Katrina and many others. What I realized was that all these catastrophes had one major factor in common: they were all preventable. That is, in each case the relevant engineers saw the potential for problems and warned their superiors, but in each case senior managers dismissed those warnings, often due to a kind of hubris I call the "It can't happen here" syndrome. More importantly, they did not comprehend the risks.

The Fukushima disaster was no different; in fact, it was totally preventable. Warnings had been issued for years, warnings that would have been red flags to any risk management officer, but those warnings were ignored. The Commission said as much

in their conclusion: "What must be admitted — very painfully — is that this was a disaster 'Made in Japan'... Its fundamental causes are to be found in the ingrained conventions of Japanese culture: our reflexive obedience; our reluctance to question authority; our devotion to 'sticking with the program'; our groupism; and our insularity."[2] They might also have echoed Dr. Richard Feynman, who concluded his addendum to the official report on the Challenger disaster with these famous words: "For a successful technology, reality must take precedence over public relations, for nature cannot be fooled."[3]

In the end, to do any kind of security, we must take to heart the a priori precepts of risk management: a) people make mistakes, b) machines eventually break and c) accidents inevitably happen. True risk management is not a quest for absolutes (100% fail-proof operation over the life of a system), but a practice in resilience, in predicating one's thinking on the assumption that *all systems will sooner or later fail in some way*. If that is your starting assumption, then real "security" becomes a challenge in how best to *recover* from all types of accidents, break-downs and system failures, both foreseeable and as yet unimagined.

We Are All Risk Management Creatures

The most important thing to understand about "security" and risk management in general is that these are not new additions to human thinking, but an intrinsic part of our oldest, most basic brain structures. In fact, the most fundamental security systems are "built in" – they're literally part of our DNA. Nor are we unique in this aspect; Nature has risk management in its genes. Which means we don't "learn" security-type thinking — we adapt, develop and (occasionally) improve on aspects of our natural heritage. This is what Einstein was getting at when he said, "Look deep into Nature, and then you will understand everything better."

We are all risk management creatures. When we wake up in the morning, without even thinking about it, we smell the air and listen for certain sounds. If we're at home and there's no perceived threat, we will naturally be less attentive to our surroundings than if we awoke, say, in an unfamiliar hotel room. Yet even in the comfortable, "risk-free" environment of our own homes, we're still safety-checking constantly, from instinctively scanning the stairs for objects that don't belong there to smelling the milk before it goes in our coffee. Getting in the shower, getting dressed, sipping that hot coffee, eating breakfast, watching the weather report — by the time we go out the door in the morning we have done a hundred routine risk management checks. Most of them require little or no conscious attention. We'd say they're habits, but that's not entirely correct; they've become habits precisely because the fundamental templates were biologically coded. We do these things instinctively because we are security-oriented, risk-management animals to begin with. All larger, more

[2] http://warp.da.ndl.go.jp/info:ndljp/pid/3856371/naiic.go.jp/wp-content/uploads/2012/09/NAIIC_report_lo_res10.pdf

[3] Feynman, R.P., "Personal observations on the reliability of the Shuttle"; Report of the Presidential Commission on the Space Shuttle Challenger Accident, Appendix F (1986).

sophisticated approaches to controlling risk are variations on or amplifications of Nature's basic instincts of self-preservation. Humans, like all biological creatures, cannot eliminate risk; but by instinctively performing many small risk-mitigating actions, we usually manage to avoid the more serious negative outcomes, like dying.

First, Look to Nature

It is this background that causes me to take a step back when I hear discussions of the newest security technologies. "Why re-invent the wheel?" I say. "If you're faced with a security problem, you should start by looking at how similar problems are dealt with in Nature." I'm not sure why this attitude seems so radical, because to me it seems pretty obvious. Before you start to tell me about the newest, coolest, "unbreakable" code, first show me that you've thought about, or at least that you're aware of analogous situations in the natural world.

Here's a simple, elegant example: think about how a human egg is fertilized. One and only one sperm is allowed to get inside that egg, and once it does, the vault doors are closed and locked. This *non-repudiation* "system" prevents polyspermia through electrical or chemical means while authenticating the spermatozoa itself[4]. I argue that such examples should be our starting point when we talk about, e.g., financial systems that need to specify and authenticate one and only one transaction.

Look at how white blood cells (lymphocytes), penicillin and other antibiotics *authenticate* and attack certain bacteria but leave others untouched. Or how DNA replication includes a regular checkpoint mechanism[5] that verifies the *integrity* of the copying process, thus ensuring that replication occurs perfectly every time. Our immune system has spent millions of years evolving into a decentralized and distributed control system that consists of billions of cells working together to manage a huge variety of threats in a robust, scalable and flexible manner. *Resilience* manifests itself in many ways, including organisms that switch between sexual and asexual reproduction depending on environmental conditions. The mechanism for cell signaling comprises at least six different communication methods, including hormones that transmodulate and *encode* different signals for different pathways.

In a truly "eat or be eaten" world, viral, bacterial and animal species have developed both offensive and defensive mechanisms to protect themselves, including cloaking, stigmergy and mimcry, which clearly have IT security analogues, such as polymorphic, APT, botnet, DDoS, phishing and pharming attacks. While the security industry discovers and responds to a seemingly endless profusion of threats, our bodies and Nature in general have been constantly fighting a far greater war just to stay alive.

---

[4] Gilbert SF. Developmental Biology. 6th edition. Sunderland (MA): Sinauer Associates; 2000. Gamete Fusion and the Prevention of Polyspermy.

[5] Noguchi, E. The DNA Replication Checkpoint and Preserving Genomic Integrity During DNA Synthesis; 2010 Nature Education 3(9):46.

Not Short-Term Solutions, But Sustainable Success

There are thousands of other examples of efficient, effective security solutions, both "out there" in the natural world and within our own bodies. Today's IT systems are still at a fairly primitive stage of mimicking existing natural systems. Natural systems work exceedingly well because of a single key difference between their design and modern, man-made systems — their risk control "protocols" are an integral part of their being. Their security "code" is written in DNA; it's designed-in from the start, not added on later as an afterthought.

The other overwhelming characteristic of the natural world, and the one that has produced so many risk management responses in animals and plants, is the process of evolution. Remember that basic fact: Nature *evolves* — it changes; it learns; it gets better – it is resilient. Yes, a whole species may suffer, and in extreme cases it might nearly die out, but the "system" learns. It responds, it discovers ways to neutralize threats, and eventually it triumphs over those threats. That "flexible response" is another key to long-term, sustainable success. It is interesting to note that the same tradeoffs in terms of cost, speed, and security vs. convenience that concern us in modern system design can also be studied in the natural world. The latter systems have evolved over hundreds of millennia to provide the optimum mix of security and efficiency.

Our security needs will continue to increase as fast as new technologies are commercialized to make our lives faster and easier. Ironically, we will need to fundamentally rethink our approach to security in order to remain up to date in this changing environment. The problem will always be: How do we keep ourselves and our data safe in an increasingly interconnected world? The answers may be closer than we think, based on hints we discover in the natural world.

Quantitative Analysis of the Full Bitcoin Transaction Graph

Dorit Ron and Adi Shamir

Department of Computer Science and Applied Mathematics,
The Weizmann Institute of Science, Israel
{dorit.ron,adi.shamir}@weizmann.ac.il

Abstract. The Bitcoin scheme is a rare example of a large scale global payment system in which all the transactions are publicly accessible (but in an anonymous way). We downloaded the full history of this scheme, and analyzed many statistical properties of its associated transaction graph. In this paper we answer for the first time a variety of interesting questions about the typical behavior of users, how they acquire and how they spend their bitcoins, the balance of bitcoins they keep in their accounts, and how they move bitcoins between their various accounts in order to better protect their privacy. In addition, we isolated all the large transactions in the system, and discovered that almost all of them are closely related to a single large transaction that took place in November 2010, even though the associated users apparently tried to hide this fact with many strange looking long chains and fork-merge structures in the transaction graph.

Keywords: bitcoin, digital coins, electronic cash, payment systems, transaction graphs, quantitative analysis.

1 Introduction

Bitcoins are digital coins which are not issued by any government, bank, or organization, and rely on cryptographic protocols and a distributed network of users to mint, store, and transfer. The scheme was first suggested in 2008 by Satoshi Nakamoto [1], and became fully operational in January 2009. It had attracted a large number of users and a lot of media attention [2] [3] [4], but so far it was difficult to get precise answers to simple questions such as: How many different users are there in the system? How many bitcoins are typically kept in each account, and how does this balance vary over time? Are most bitcoins kept by a few large users? Do they keep their bitcoins in "saving accounts" or do they spend them immediately? How many users had large balances at some point in time? What is the size distribution of bitcoin transactions, and how many of them are micropayments?

In this paper we answer some of these questions. We use the fact that all the transactions ever carried out in the Bitcoin system are available on the internet (in an anonymous way). On May 13^{th} 2012 we downloaded the full public

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 6–24, 2013.

record of this system [1], which consisted of about 180,000 HTML files. After parsing and processing these files, we built a graph of all the Bitcoin addresses and transactions up to that date. We then used the methodology described in the next section in order to try to identify which addresses are likely to belong to the same entity, and used this information to contract the transaction graph by merging such addresses, in order to get a more accurate picture of the full financial activity of each user. We then analyzed many statistical properties of both the original and the contracted transaction graphs (most of our statistical results were very similar for the two graphs, within a factor of 2). The most interesting and informative distributions we found are described in a series of tables. In addition, we isolated all the large ($\geq 50,000$ bitcoins) transactions which were ever recorded in the system, and analyzed how these amounts were accumulated and then spent. We discovered that almost all these large transactions were the descendants of a single large transaction involving 90,000 bitcoins which took place on November 8^{th} 2010, and that the subgraph of these transactions contains many strange looking chains and fork-merge structures, in which a large balance is either transferred within a few hours through hundreds of temporary intermediate accounts, or split into many small amounts which are sent to different accounts only in order to be recombined shortly afterwards into essentially the same amount in a new account.

There was one previous reported attempt [5] to download and analyze the full Bitcoin history, which also used the same methodology to try to contract all the addresses which are believed to belong to the same user. They created the graph of transactions on July 12^{th} 2011, which was before the scheme really caught on. Thus, the total number of Bitcoins participating in all the transactions in our graph is about three times larger than in their graph. In addition, we expect the transactions in our more mature graph to better represent typical use of the system, whereas their graph represents primarily the experiments run by early adopters. However, the biggest difference between our papers is that they were primarily interested in privacy issues, whereas we are primarily interested in the statistical properties of the bitcoin transaction graph. Another analysis of the Bitcoin transaction graph was presented at the Chaos Computer Club Conference in Germany in December 2011 [6]. Again, they were primarily interested in how to defeat the anonymity of the network, but also included some interesting comments about the economic principles behind the scheme, the effect of lost coins on its operation, weaknesses in its protocols, and the general topological properties of this transaction graph.

The paper is organized as follows. In Section 2 we describe the Bitcoin scheme. In Section 3 we summarize the main statistical distributions we extracted from the downloaded transactions, which describe many interesting and even surprising

[1] It is believed (but we could not fully verify) that the data from `http://blockexplorer.com/` *should* be exactly the same as what one could get as a Bitcoin client. Even if there are tiny differences they are likely to have only a negligible effect on our statistical results.

properties of the scheme. Finally, in Section 4 we present the graph of the largest transactions and analyze its strange structure.

2 The Bitcoin Scheme

Bitcoin is a decentralized electronic cash system using peer-to-peer networking to enable payments between parties without relying on mutual trust. It was first described in a paper by Satoshi Nakamoto (widely presumed to be a pseudonym) in 2008. Payments are made in bitcoins (BTC's), which are digital coins issued and transferred by the Bitcoin network. The data of all these transactions, after being validated with a proof-of-work system, is collected into what is called the block chain.

Participants begin using bitcoin by first acquiring a program called a Bitcoin wallet and one or more Bitcoin addresses. Bitcoin addresses are used for receiving bitcoins, in the same way that e-mail addresses are used for receiving e-mails. Even though Bitcoin is considered to be an experimental payment system, it is already deployed on a large scale (in the sense that the current value of all the coins issued so far exceeds 100,000,000 USD) and attracts a lot of media attention. Its proponents claim that it is the first truly global currency which does not discriminate its users based on citizenship or location, it is always running with no holidays, it is easy to secure with very low usage fees, it has no chargebacks, etc. On the other hand, its detractors claim that it is widely misused to buy illegal items [7] and to launder large sums of money, and that it is too easy to steal bitcoins from wallets via cyber attacks.

Unlike fiat currency, which has been declared to be legal tender by a government despite the fact that it has no intrinsic value and is not backed by reserves, the Bitcoin scheme has no centralized issuing authority. The network is programmed to increase the money supply in a slowly increasing geometric series until the total number of bitcoins reaches an upper limit of about 21 million BTC's. Bitcoins are awarded to Bitcoin "miners" for solving increasingly difficult proof-of-work problems which confirm transactions and prevent double-spending. The network currently requires over one million times more work for confirming a block and receiving an award (currently 50 BTC's) than when the first blocks were confirmed.

The exchange rate of bitcoins has fluctuated widely over the years, from merely \$0.01 to over \$30 per BTC. Today (October 2012) it is worth a little over \$12 per BTC. The entire activity in the Bitcoin network is publicly available through the internet in two major forms, and the one we decided to download appears as a block chain, starting at block 0 [8] (created back on the 3^{rd} of January 2009). Each block reports on as little as a single transaction to as much as over a thousand transactions, and provides hyperlinks to other blocks and to other activities of each address.

Many users adopt the Bitcoin payment system for political and philosophical reasons. Each user can have an unbounded number of addresses (which are characterized by their public/private key pairs) owned by him. A transaction

in bitcoins is a generalization of a regular bank transaction in the sense that it allows multiple sending addresses and multiple receiving addresses in the same transaction. It specifies how many bitcoins were taken from each sending address and how many bitcoins were credited to each receiving address, without the details of who gave how much to whom. An address may receive bitcoins which are either newly minted or have a specific sending address. Another important difference between bitcoin transactions and regular bank transactions is the notion of *change*, which is related to the fact that bitcoins are kept in (possibly fractional sized) chunks which have to be transferred in an all or nothing way. For example, a user can have three chunks of 10 bitcoins each. A transaction can spend 12.5 bitcoins by transferring the first full chunk plus 2.5 bitcoins from the second chunk, and then the 7.5 bitcoin change should be sent to a new address owned by the same user with new public and private keys. The user then has the option of either transferring the third chunk to the new address, or leaving it in the old address. In fact, it is considered good practice for a user to generate a new address, i.e., public-private key-pair, for every transaction even if this is not necessary. To better protect their identity, users are advised to take the following steps: they do not have to reveal any identifying information in connection with their addresses; they can repeatedly send varying fractions of their BTC's to themselves using multiple (newly generated) addresses; and/or they can use a trusted third-party in the form of a shared e-wallet to mix their transactions with those of other owners.

These operational and privacy policies of the Bitcoin scheme make it desirable for us to try to contract the transaction graph in order to get a more informative picture of the total assets and financial activities of users which are associated with many addresses, and to try to distinguish between "internal" and "external" transfers of bitcoins in it. Performing this contraction in a completely accurate way seems to be extremely difficult, but we can use the available data in order to try to find a good first approximation. Since many transactions have multiple sending addresses, we can make the reasonable assumption that all these addresses have the same owner. We then compute the transitive closure of this property over all the transactions. For example, if there is one transaction in which 1 and 2 are used as sending addresses, and another transaction in which 2 and 3 are used as sending addresses, we conclude that all three addresses are jointly owned. This can lead to two types of errors: We can underestimate the common ownership of some addresses because there was no evidence for it in the available data, and we can overestimate it if several users decided to pool their activities and to send a single transaction to which each one of them contributes some of the sending addresses. Discussions with several members of the Bitcoin community lead us to believe that at the moment there are likely to be very few overestimation errors of this type, but quite a few underestimation errors. For example, when we tried to use all the available transactions to merge the addresses of a particular large user, we were told that we managed to identify with our methodology only about one quarter of his real addresses. Note that the linkability of the addresses does not imply that the identity of the user becomes

known. However, if we have any external information about the real ownership of any one of the merged addresses, we can get a fuller picture of the Bitcoin activity of that particular individual or organization. For example, since WikiLeaks publicly advertised one of its addresses when it asked for donations, we can estimate with our methodology that WikiLeaks owns at least 83 addresses, that it was involved in at least 1088 transactions, and that it had an accumulated income in all these addresses of 2605.25 BTC's.

We acquired the complete state of the Bitcoin transaction system on May 13^{th} 2012, which contained all the transactions carried out in the system since its inception on January 3^{rd} 2009 until that date. This required downloading 180,001 separate but linked HTML files, starting from block number 180,000 [9] and following the links backwards to the zeroth block initiating the system in January 2009. Each file was parsed in order to extract all the multisender/multireceiver transactions in it, and then the collection of transactions was encoded as a standard database on our local machine. We then ran a variant of a Union-Find graph algorithm [10] in order to find sets of addresses which are expected to belong to the same user. We merged all the nodes and combined all the transactions which can be associated with him (without eliminating the internal transfers, which become self loops in the new graph). We call the original transaction graph the *address graph*, and the contracted transaction graph the *entity graph* (we avoid using the word "owner" with its complex legal connotations since we do not really know who owns each address, and instead use the neutral word "entity" as our best approximation to the common owner of multiple addresses). All the statistics described in Section 3 are derived from both the address graph and the entity graph, as indicated in the tables. In most (but not all) cases, we expect the statistics to change monotonically as we move from the address graph to the entity graph and then to the (unknown) owner graph, since each entity is typically the union of several addresses which we managed to merge, and each real owner is typically the union of several entities that we failed to merge. For example, since the average balance of an address is 2.4 BTC's and the average balance of an entity is 3.7, we can argue that the average balance of an owner is likely to be larger than 3.7 BTC's. This monotonicity can thus be used to provide plausible upper or lower bounds for the statistical properties of the real ownership graph, even though we do not know it.

3 Statistics Calculated over the Bitcoin Transaction Graph

At the time we downloaded the graph there were 3,730,218 different public keys, each associated with a different address: 3,120,948 of them were involved as senders in at least one transaction, while the additional 609,270 appear in the network only as receivers of BTC's. By running the Union-Find algorithm, we were able to associate the 3,120,948 addresses with 1,851,544 different entities. Since the other 609,270 addresses were never used as senders, they could not be merged with any other addresses by the Union-Find algorithm, and thus they

all remained as entities with a single address. By adding these singletons, we get a total of 2,460,814 entities, which implies that each one of them has on average about 1.5 addresses. However, there is a huge variance in this statistics, and in fact one entity is associated with 156,722 different addresses. By analyzing some of these addresses and following their transactions, it is easy to determine that this entity is Mt.Gox, which is the most popular Bitcoin Exchange site (responsible for almost 80% of all the exchange operations in the network). The full distribution of the number of addresses per entity is given in Table 1.

In our reduced entity graph, each m-to-n transaction has a single sender (since the m sending addresses necessarily belong to the same entity) and at most n receivers. It can thus be decomposed into at most n different transactions from the single entity associated with the m senders to the entities associated with the n receivers. In case some of the receiving addresses are identified as belonging to the same entity, their amounts are accumulated to create a single common transaction, and if some of the receivers are identified with the single sender, we create a single self loop with the combined amounts. The resulting entity graph has 7,134,836 single sender and single receiver transactions, out of which 814,044 (about 11%) involve Deepbit (the largest Bitcoin mining pool), and 477,526 (about 7%) involve Mt.Gox. About 10% of the transactions are self loops. The entity graph is not connected as it is composed of 133,742 different connected components, many of size one. For instance, there are as many as 43,710 components (about 33%) consisting of a single address which are used only for accepting (one or several batches of) freshly minted bitcoins, and which have never participated in any incoming or outgoing transactions. Note that the address graph has a larger number of 13,734,847 transactions of lower values, since a single transaction with 2 sending addresses and 3 receiving addresses is represented in the address graph as 6 single-sender and single-receiver transactions.

There are many types of statistics and graphs about the Bitcoin network which can be readily downloaded from the internet [11] [12]. However, these types of statistics tend to describe some global property of the network over time, such as the number of daily transactions, their total volume, the number of bitcoins minted so far, and the exchange rate between bitcoins and US dollars. We can go much further than that, since the entire transaction graph can be used to determine the financial history of each entity including all of its sending/receiving activities along with the daily balance of bitcoins in its various addresses and how they vary over time. Having this entity graph at hand enables us to study various statistical properties of the network, which are not easy to determine by following a small number of online links in the Blockexplorer representation of the Bitcoin network. In the rest of this section, we describe some of our findings so far.

Here is our first surprising discovery, which is related to the question of whether most bitcoins are stored or spent. The total number of BTC's in the system is linear in the number of blocks. Each block is associated with the generation of 50 new BTC's and thus there are 9,000,050 BTC's in our address graph (generated from the 180,001 blocks between block number zero and block

number 180,000). If we sum up the amounts accumulated at the 609,270 addresses which only receive and never send any BTC's, we see that they contain 7,019,100 BTC's, which are almost 78% of all existing BTC's. Due to the way bitcoins can be repeatedly moved to fresh addresses, some of which can be very recent, we can not claim that all these bitcoins are out of circulation. However, 76.5% of these 78% (i.e., 59.7% of all the coins in the system) are "old coins", defined as bitcoins received at some address more than three months before the cut off date (May 13^{th} 2012), which were not followed by any outgoing transactions from that address after they were received. One can also argue that very old dormant bitcoins were simply abandoned or lost by users who experimented with the system in its early days, when it was very difficult to buy anything or to exchange bitcoins into dollars. To be even more cautious with our estimation of dormant bitcoins, we decided to ignore all the transactions which took place prior to July 18^{th} 2010, when Mt.Gox started its exchange and price quoting services. The sum of the balances of all the addresses which have not been active since that date is 1,657,480 bitcoins. Clearly, by considering all these bitcoins as "lost" rather than "hoarded" we are underestimating the number of bitcoins which are kept dormant in "saving accounts". By ignoring these very old bitcoins and repeating the same calculation, we found that 73% of all the remaining BTC's were accumulated at addresses which only receive and never send bitcoins, and that 70% of these 73% (i.e., 51%) are dormant bitcoins in the sense that they were received more than three months before our cutoff date but after it became easy to exchange them. If instead of summing the *transaction values* we sum the *final balances* of all the addresses that were active after July 18^{th} 2010 but became inactive in the last three months, we get that 55% of all coins in the system are dormant in this sense. This is strong evidence that the majority of bitcoins are not circulating in the system, and since it is based on the address rather than the entity graph, this conclusion is not affected by possible inaccuracies in the way we associate addresses with users. Note that the total number of bitcoins participating in all the transactions since the establishment of the system (except for the actual minting operations) is 423,287,950 BTC's, and thus each coin which is in circulation had to be moved a large number of times to account for this total flow.

A previously proposed measure of the level of activity in Bitcoin was the idea of "bitcoin days destroyed" [13], which gives more weight to coins which haven't been spent in a while. To do this, one multiplies the amount of each transaction by the number of days since those coins were last spent. This is believed to give a better indication of how much real economic activity is occurring on the Bitcoin network, rather than just looking at the total transaction volume per day. The measure we use is incomparable to and fundamentally different from the "bitcoin days destroyed" as it accumulates bitcoins *left untouched* (for at least three months) in addresses, without adding any contribution from those which have been recently moved: What we focus on are those coins that are kept completely out of circulation.

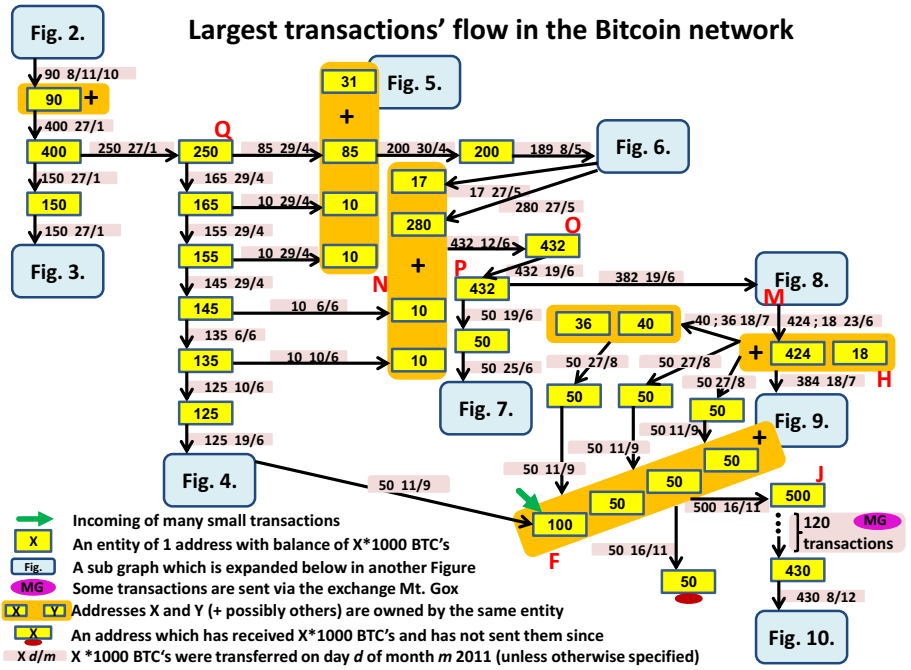


Fig. 1. The backbone of the graph of almost all largest transactions in the Bitcoin scheme (those which are larger than 50,000 BTC's). The red letters refer to some of the most active entities in Bitcoin as listed in Table 7.

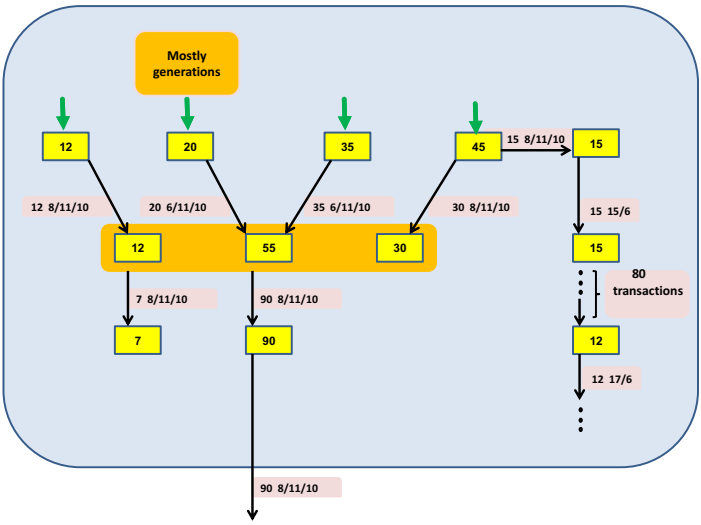


Fig. 2. A Sub graph of Fig. 1: A trace back of some flows of BTC's leading to the first large transaction of 90,000 BTC's on November 8^{th} 2010

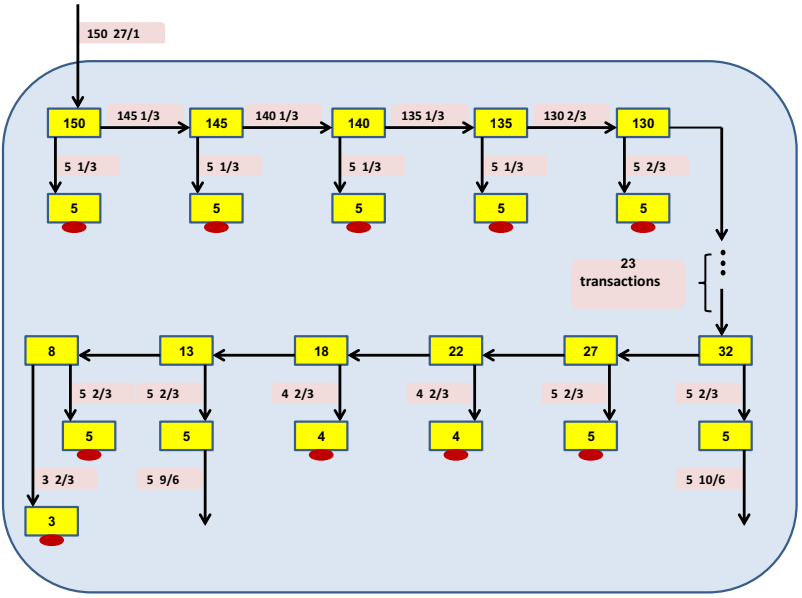


Fig. 3. A Sub graph of Fig. 1: A long chain of transactions where each address puts aside a small amount of BTC's. Those amounts sum up to 140,000 BTC's.

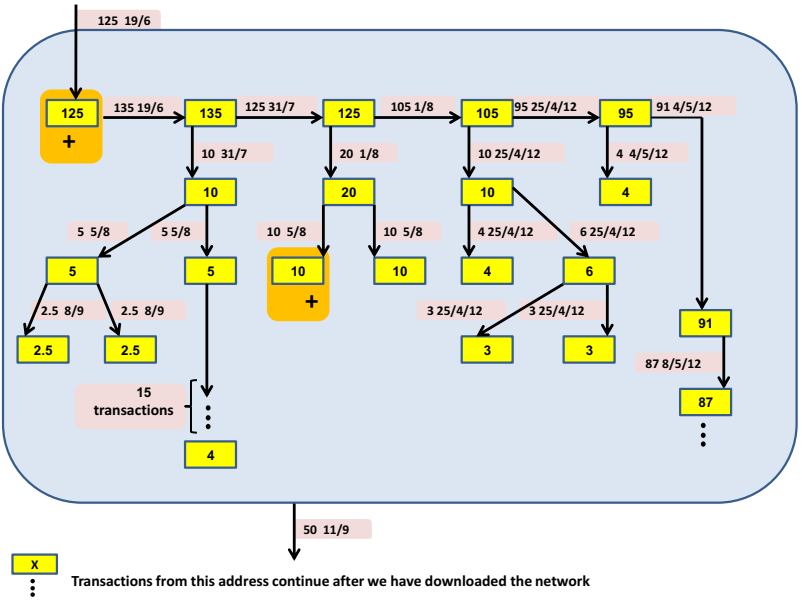


Fig. 4. A Sub graph of Fig. 1: A long chain of transactions where each address transfers most of its BTC's forward. The rest is distributed in a binary tree-like structure.

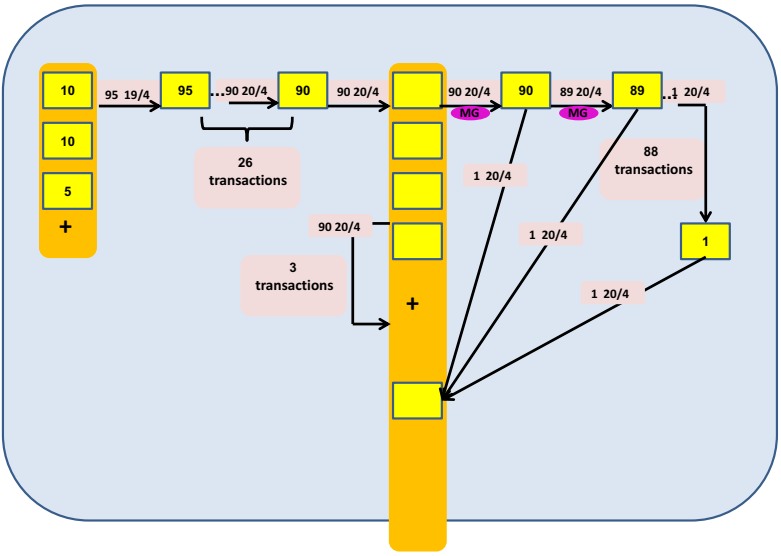


Fig. 5. A Sub graph of Fig. 1: An entity is sending 90,000 BTC's to itself in a self loop, then transfers it forward but gets it back via 90 transfers of 1,000 BTC's each, all carried out on the same day. 31,000 of it is then transferred forward.

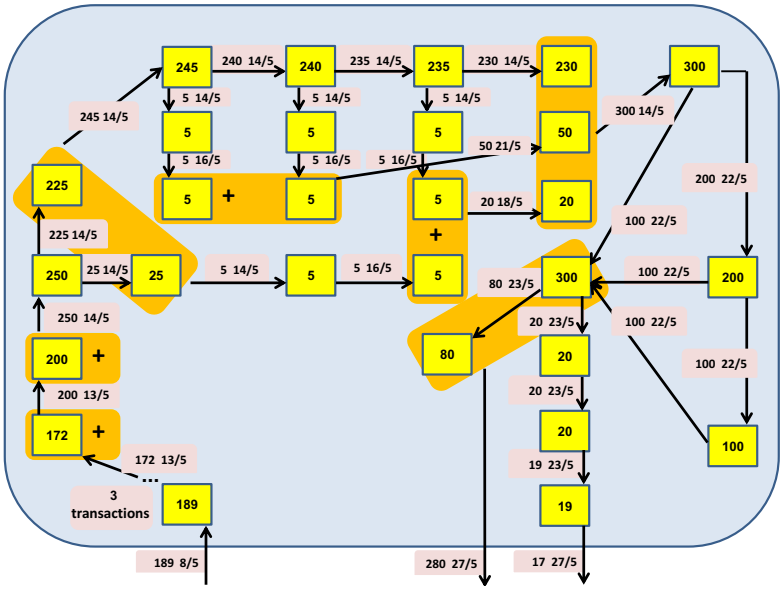


Fig. 6. A Sub graph of Fig. 1: Large amounts of BTC's are transferred from one address to another by sending parts of it to intermediate addresses, which are then being merged into the same destination

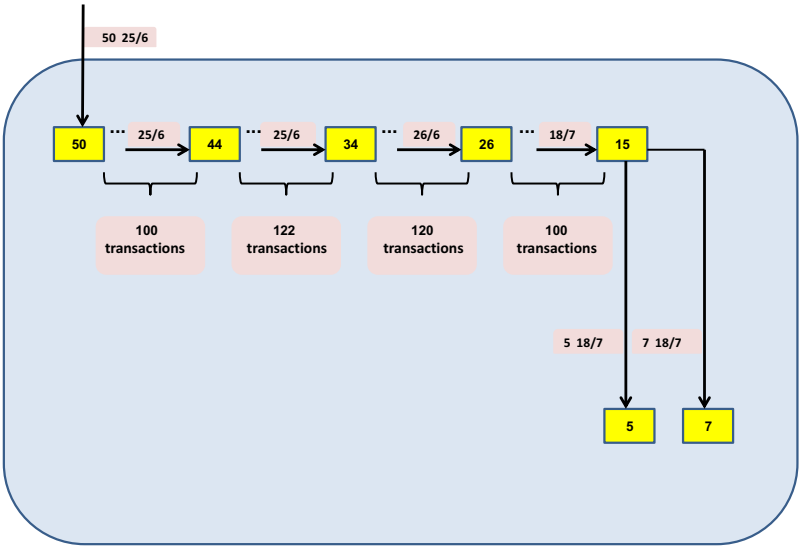


Fig. 7. A Sub graph of Fig. 1: Large amounts of BTC's are rapidly transferred in a very long chain of hundreds of transactions in a very short period of time

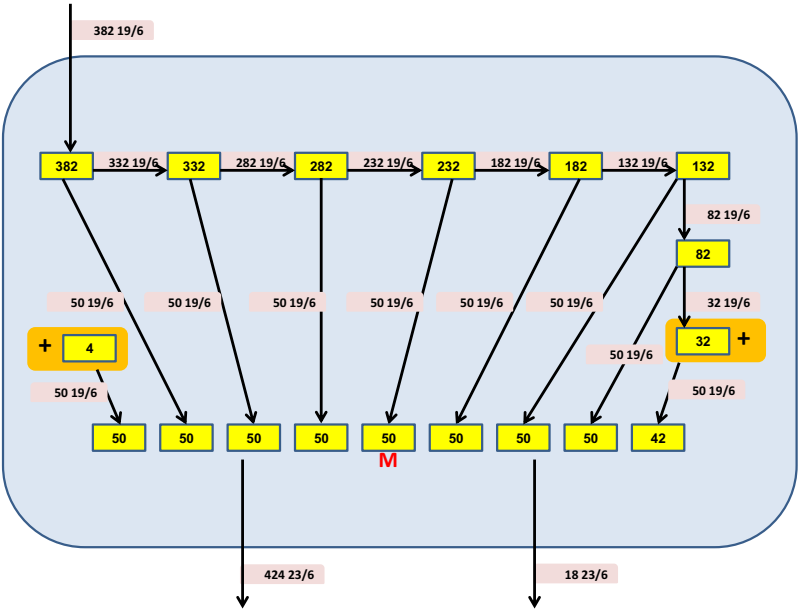


Fig. 8. A Sub graph of Fig. 1: A very large amount of BTC's is transferred by splitting it into equal amounts each directed to a different address belonging to the same entity, then most of the accumulated sums are transferred to a single receiver

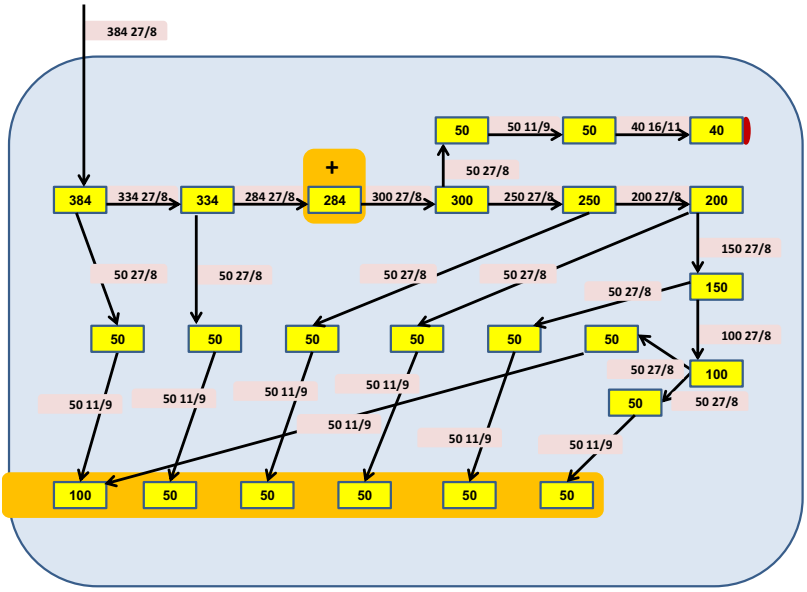


Fig. 9. A Sub graph of Fig. 1: A similar scenario as described in Fig. 8 but with more intermediate addresses

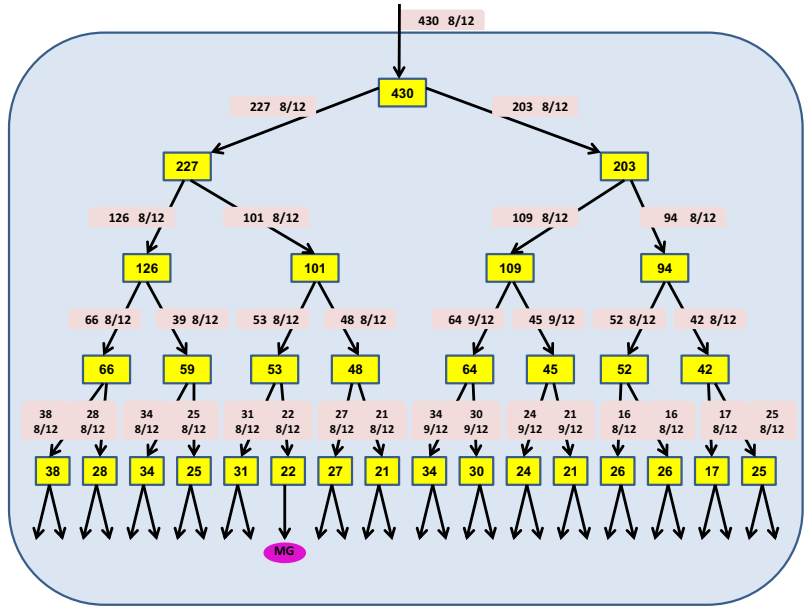


Fig. 10. A Sub graph of Fig. 1: The largest amount of transferred BTC's is finally distributed among many addresses via a binary tree-like structure

Another interesting finding is that the total number of bitcoins received by most entities and addresses is negligible. In the rest of this section, we use unparenthesized numbers to indicate values derived from the entity graph, and parenthesized numbers to indicate values derived from the address graph. For example, as can be seen from Table 2, 36% of all entities (and 40% of all addresses) received fewer than one BTC, currently worth about 12 USD, throughout their lifetime, 52% (59%) received fewer than 10 BTC's and 88% (91%) fewer than 100. At the other end of the distribution there are only four entities (and one address) which received over 800,000 BTC's, and 80 entities (129 addresses) which received over 400,000.

Similarly, as can be seen in Table 3 the current (on May 13^{th} 2012) balance of almost 97% (98%) of all entities (addresses) was less than 10 BTC's. This number decreases to 88% (91%) if instead of looking at one specific moment, we look at the *maximal balance ever seen* throughout an entity's (address'es) lifetime. This statistics is summarized in Table 4. In addition, it can be seen that there are only 78 entities (70 addresses) with current balance larger than 10,000 BTC's. This number grows to 3,812 (3,876) when looking at the maximal balance ever seen.

Another measure that may indicate the level of activity of an entity (address) is the number of transactions it has been involved with. Its distribution is presented in Table 5. It is remarkable that 97% (93%) of all entities (addresses) had fewer than 10 transactions each, while 75 entities (80 addresses) use the network very often and are affiliated with at least 5,000 transactions.

We have also calculated the distribution of the size of the transactions in the two graphs as summarized in Table 6. Again, it is evident that many transactions are very small, and 28% (47%) are smaller than 0.1 BTC each. The Bitcoin scheme actually enables sending *micro* transactions, which are of the order of 10^{-8} BTC (this is the smallest fraction into which a BTC can be broken, and is called a satoshi). When we also consider midsize amounts, we see that 73% (84%) of the transactions involve fewer than 10 BTC's. On the other hand, large transactions are rare at Bitcoin: there are only 364 (340) transactions larger than 50,000 BTC's. We have carefully inspected all these large transactions and describe our findings in the next section.

It is interesting to investigate the most active entities in the Bitcoin system, those who have either maximal incoming BTC's or maximal number of transactions. 19 such entities are shown in Table 7 sorted in descending order of the number of accumulated incoming BTC's shown in the third column. The leftmost column associates the entities with letters between A to S out of which three are identified: B is Mt.Gox, G is Instawallet and L is Deepbit. Eight additional entities: F, H, J, M, N, O, P, and Q are pointed out in the graph of the largest transactions (Fig. 1) which is presented in the next section. The second column gives the number of addresses merged into each entity. The fourth column presents the number of transactions the entity is involved with.

Table 7 shows that Mt.Gox has the maximal number of addresses, but not the largest accumulated incoming BTC's nor the largest number of transactions.

Entity A in the first row of Table 7 owns the next largest number of addresses, about 50% of those of Mt.Gox's, but received 31% more BTC's than Mt.Gox. Deepbit had sent 70% more transactions than Mt.Gox. It is interesting to realize that the number of addresses of 13 of these entities is a fifth or more of the number of transactions they have executed, which may indicate that each address is indeed used for just a few transactions. It is also clear that six out of the 19 entities in the table have each sent fewer than 30 transactions with a total volume of more than 400,000 BTC's. Since these entities were using large transactions, we were able to isolate them and to follow the flow of their transactions, see Section 4 below. On the other hand, entity A had never sent any large transactions and thus it has not been included in our graph of the largest transactions.

4 The Graph of the Largest Transactions in Bitcoin

We have identified and analyzed all the largest ($\geq 50,000$ BTC's) transactions in the entity graph, (there were 364 such transactions as described in the last column of Table 6), and followed their flow. We started with the *earliest* such large transaction, the one of 90,000 BTC's made on November 8^{th} 2010. By tracing each of the other 363 large transactions in this category, we were able to show that 348 were actual successors of this initial transaction. The resulting directed graph is depicted in Fig. 1. This graph reveals several characteristic behaviors of the flow in the Bitcoin transaction graph: long consecutive chains of transactions, fork-merge patterns that may include self loops, setting aside BTC's and final distribution of large sums via a binary tree-like structure.

Long Chains. A common prominent practice of Bitcoin users is to create chains of consecutive transactions. Some of these chains can be explained by the change mechanism in which small payments are accompanied by the creation of a new address, into which the user transfers the difference. Such chains can be found in Fig. 2, Fig. 4, Fig. 5 and Fig. 7, with lengths of 3, 15, 26, 80, 88 and 350 transactions. However, the behavior seen in Fig. 3 deviates significantly from this pattern, since the same amount of 5,000 bitcoins is repeatedly split off the main sum and put into accounts which have no additional transactions associated with them.

Fork-Merge Patterns and Self Loops. Another frequent scenario in Bitcoin is transferring a large number of BTC's from one address to another via several intermediate addresses, each receiving part of the entire amount and then sending it, mostly in full, to the same destination whether directly or via other mediators. Examples can be seen in Fig. 6, Fig. 8 and Fig. 9. A harder to follow fork-merge pattern is presented in Fig. 5: An entity is sending 90,000 BTC's to itself three times in self loops. Each time it splits it into different amounts, 76+14, 72+18 and 69+21. It uses the same address for the small amounts and different addresses for the large amounts. Then it exchanges the entire 90,000 BTC's at Mt.Gox. Finally, the 90,000 BTC's are being transferred via a chain of 90

transactions using 90 different addresses (which may or may not belong to the same owner), where at each one of them 1,000 BTC's are sent back to the first entity, recombined into essentially the very first amount of 90,000 BTC's.

Keeping Bitcoins in "Saving Accounts". Another long chain of transactions from the beginning of March 2011 can be seen in Fig. 3. This chain is different from the above ones, since at 28 out of its 30 steps, it puts aside 5,000 BTC's in what seems to be "saving accounts". The accumulated sum of 140,000 BTC's has never been sent since. These bitcoins are an example of our discovery that most of the bitcoins are not circulating in the system.

Binary Tree-Like Distributions. Often amounts of BTC's are distributed among many addresses by splitting it into two similar amounts at each step. This results in a binary tree-like structure as depicted in Fig. 10 and in Fig. 4.

5 Conclusions

The Bitcoin system is the best known and most widely used alternative payment scheme, but so far it was very difficult to get accurate information about how it is used in practice. In this paper we describe a large number of statistical properties of the Bitcoin transaction graph, which contains all the transactions which were carried out by all the users until May 13^{th} 2012. We discovered that most of the minted bitcoins remain dormant in addresses which had never participated in any outgoing transactions. We found out that there is a huge number of tiny transactions which move only a small fraction of a single bitcoin, but there are also hundreds of transactions which move more than 50,000 bitcoins. We analyzed all these large transactions by following in detail the way these sums were accumulated and the way they were dispersed, and realized that almost all these large transactions were descendants of a single transaction which was carried out in November 2010. Finally, we noted that the subgraph which contains these large transactions along with their neighborhood has many strange looking structures which could be an attempt to conceal the existence and relationship between these transactions, but such an attempt can be foiled by following the money trail in a sufficiently persistent way.

Acknowledgments. This research was supported by the Citi Foundation. We would like to thank Ronen Basri, Uriel Feige, Michal Irani, Robert Krauthgamer, Boaz Nadler, Moni Naor and David Peleg from the Computer Science and Applied Mathematics Department of the Weizmann Institute of Science for many interesting and informative discussions. We would also like to thank Aharon Friedman for his major help in acquiring and processing the Bitcoin data base. Finally, we would like to thank all the members of the Bitcoin community, and in particular Meni Rosenfeld, Stefan Richter and Peter Todd, who sent us excellent comments, criticisms and suggestions. We revised the original version of the paper in order to respond to their input.

References

1. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
2. Wallace, B.: The Rise and Fall of Bitcoin, Wired Magazine (November 23, 2011), http://www.wired.com/magazine/2011/11/mf_bitcoin/all/
3. NPR Staff: Silk Road: Not Your Father's Amazon.com (June 12, 2011), http://www.npr.org/2011/06/12/137138008/silk-road-not-your-fathers-amazon-com
4. Brett, W.: Senators seek crackdown on "Bitcoin currency", Reuters (June 8, 2011), http://www.reuters.com/article/2011/06/08/us-financial-bitcoins-idUSTRE7573T320110608
5. Reid, F., Harrigan, M.: An Analysis of Anonymity in the Bitcoin System, arXiv:1107.4524v2 [physics.soc-ph] (May 7, 2012)
6. Hamacher, K., Katzenbeisser, S.: Bitcoin - An Analysis (December 29, 2011), http://www.youtube.com/watch?v=hlWyTqL1hFA
7. Christin, N.: Traveling the Silk Road: A measurement analysis of a large anonymous online, arXiv:1207.7139v1 [cs.CY] (July 31, 2012)
8. Bitcoin's block number 0, http://blockexplorer.com/b/0
9. Bitcoin's block number 180,000, http://blockexplorer.com/b/180000
10. Cormen, T.H., Leiserson, C.H., Rivest, R.L., Stein, C.: Data structures for Disjoint Sets. In: Introduction to Algorithms, 2nd edn., ch. 21, pp. 498–524. MIT Press, McGraw Hill (2001)
11. Forbes: Top 10 Bitcoin Statistics, http://www.forbes.com/sites/jonmatonis/2012/07/31/top-10-bitcoin-statistics/
12. Block chain: Bitcoin charts, http://blockchain.info/charts
13. Bitcoin Days Destroyed, https://en.bitcoin.it/wiki/Bitcoin_Days_Destroyed

Appendix: The Distributions and the List of the Active Entities

Table 1. The distribution of the number of addresses per entity

Larger or equal to	Smaller than	Number of entities
1	2	2,214,186
2	10	234,015
10	100	12,026
100	500	499
500	1,000	35
1,000	5,000	41
5,000	10,000	5
10,000	50,000	5
50,000	100,000	1
100,000		1

Table 2. The distribution of the accumulated incoming BTC's per entity and per address

Larger or equal to	Smaller than	Number of entities	Number of addresses
0	1	893,763	1,497,451
1	10	389,302	698,132
10	100	881,273	1,206,209
100	1,000	255,826	285,820
1,000	10,000	36,713	38,484
10,000	50,000	3,593	3,723
50,000	100,000	181	190
100,000	200,000	55	50
200,000	400,000	30	29
400,000	800,000	76	129
800,000		4	1

Table 3. The distribution of the current (on May 13^{th} 2012) balance of BTC's per entity and per address

Larger or equal to	Smaller than	Number of entities	Number of addresses
0	0.01	2,097,245	3,399,539
0.01	0.1	192,931	152,890
0.1	10	95,396	101,186
10	100	67,579	68,907
100	1,000	6,746	6,778
1,000	10,000	841	848
10,000	50,000	71	65
50,000	100,000	5	3
100,000	200,000	1	1
200,000	400,000	1	1
400,000		0	0

Table 4. The distribution of the maximal balance of BTC's ever seen per entity and per address

Larger or equal to	Smaller than	Number of entities	Number of addresses
0	0.1	547,763	1,063,876
0.1	10	668,247	1,160,170
10	100	945,083	1,188,596
100	1,000	259,142	276,613
1,000	10,000	36,769	37,087
10,000	50,000	3,513	3,521
50,000	100,000	163	159
100,000	200,000	40	41
200,000	400,000	26	26
400,000	500,000	68	129
500,000		2	0

Table 5. The distribution of the number of transactions per entity and per address

Larger or equal to	Smaller than	Number of entities	Number of addresses
1	2	557,783	495,773
2	4	1,615,899	2,197,836
4	10	222,433	780,433
10	100	55,875	228,275
100	1,000	8,464	26,789
1,000	5,000	287	1,032
5,000	10,000	35	51
10,000	100,000	32	24
100,000	500,000	7	3
500,000		1	2

Table 6. The distribution of the size of the transactions in the Bitcoin scheme

Larger or equal to	Smaller than	Number of transactions in the graph of entities	Number of transactions in the graph of addresses
0	0.001	381,846	2,315,582
0.001	0.1	1,647,087	4,127,192
0.1	1	1,553,766	2,930,867
1	10	1,628,485	2,230,077
10	50	1,071,199	1,219,401
50	100	490,392	574,003
100	500	283,152	262,251
500	5,000	70,427	67,338
5,000	20,000	6,309	6,000
20,000	50,000	1,809	1,796
50,000		364	340

Table 7. The list of most active entities in Bitcoin, which have either maximal incoming BTC's or maximal number of transactions. Some of the letters in the leftmost column: F, H, J, M, N, O, P and Q refer to the red letters in Fig. 1 pointing these entities out.

Entity ID	Number of Addresses	Accumulated Incoming BTC's	Number of Transactions
A	78,251	2,886,650	246,012
B (Mt.Gox)	156,722	2,206,170	477,526
C	13,289	941,013	77,525
D	12,520	867,996	48,347
E	191	692,864	1,353
F	12	660,000	23
G (Instawallet)	23,649	633,606	92,593
H	9	580,000	59
I	10,561	514,066	49,550
J	4	500,021	6
K	134	479,254	1,039
L (Deepbit)	2	452,929	814,044
M	9	442,000	10
N	128	432,161	137
O	10	432,286	14
P	1	432,078	3
Q	14	430,490	23
R	2,124	321,866	300,486
S	1,037	20,308	197,334

Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk

Tyler Moore[1] and Nicolas Christin[2]

[1] Computer Science & Engineering, Southern Methodist University, USA
tylerm@smu.edu
[2] INI & CyLab, Carnegie Mellon University, USA
nicolasc@cmu.edu

Abstract. Bitcoin has enjoyed wider adoption than any previous cryptocurrency; yet its success has also attracted the attention of fraudsters who have taken advantage of operational insecurity and transaction irreversibility. We study the risk investors face from Bitcoin exchanges, which convert between Bitcoins and hard currency. We examine the track record of 40 Bitcoin exchanges established over the past three years, and find that 18 have since closed, with customer account balances often wiped out. Fraudsters are sometimes to blame, but not always. Using a proportional hazards model, we find that an exchange's transaction volume indicates whether or not it is likely to close. Less popular exchanges are more likely to be shut than popular ones. We also present a logistic regression showing that popular exchanges are more likely to suffer a security breach.

Keywords: Bitcoin, currency exchanges, security economics, cybercrime.

1 Introduction

Despite added benefits such as enhanced revenue [1] or anonymity [2], and often elegant designs, digital currencies have until recently failed to gain widespread adoption. As such, the success of Bitcoin [3] came as a surprise. Bitcoin's key comparative advantages over existing currencies lie in its entirely decentralized nature and in the use of proof-of-work mechanisms to constrain the money supply. Bitcoin also benefited from strongly negative reactions against the banking system, following the 2008 financial crisis: Similar in spirit to hard commodities such as gold, Bitcoin offers an alternative to those who fear that "quantitative easing" policies might trigger runaway inflation.

As of January 2013, Bitcoin's market capitalization is approximately US$187 million [4]. However, with success comes scrutiny, and Bitcoin has been repeatedly targeted by fraudsters. For instance, over 43,000 Bitcoins were stolen from the Bitcoinica trading platform in March 2012 [5]; in September 2012, $250,000 worth of Bitcoins were pilfered from the Bitfloor currency exchange [6]. Interestingly, experience from previous breaches does not suggest that failures necessarily trigger an exodus from the currency. In fact, with two possible exceptions—a June 2011 hack into the largest Bitcoin currency exchange, which coincided with the USD-Bitcoin exchange rate peaking, and the August 2012 downfall of the largest Bitcoin Ponzi scheme [8]—the (volatile) Bitcoin exchange rate has fluctuated independently from disclosed hacks and scams.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 25–33, 2013.

While Bitcoin's design principles espouse decentralization, an extensive ecosystem of third-party intermediaries supporting Bitcoin transactions has emerged. Intermediaries include currency exchanges used to convert between hard currency and Bitcoin; marketplace escrow services [7]; online wallets; mixing services; mining pools; or even investment services, be they legitimate or Ponzi schemes [8]. Ironically, most of the risk Bitcoin holders face stems from interacting with these intermediaries, which operate as de facto centralized authorities. For instance, one Bitcoin feature prone to abuse is that transactions are irrevocable, unlike most payment mechanisms such as credit cards and electronic fund transfers. Fraudsters prefer irrevocable payments, since victims usually only identify fraud after transactions take place [9, 10]. Irrevocability makes any Bitcoin transaction involving one or more intermediaries subject to added risk, such as if the intermediary becomes insolvent or absconds with customer deposits.

In this paper, we focus on one type of intermediary, currency exchanges, and empirically examine the risk Bitcoin holders face from exchange failures. Section 2 explains our data collection and measurement methodology. Section 3 presents a survival analysis of Bitcoin exchanges, and shows that an exchange probability of closure is inversely correlated to its trade volumes. Section 4 complements this analysis with a logistic regression that indicates that popular exchanges are more likely to suffer security breaches. Section 5 reviews related work and Section 6 discusses follow-up research.

2 Data on Bitcoin-Exchange Closures

2.1 Data Collection Methodology

We begin by collecting historical data on the Bitcoin exchange rates maintained by the website `bitcoincharts.com`. This includes the daily trade volumes and average weighted daily price for 40 Bitcoin exchanges converting into 33 currencies until January 16, 2013, when the data collection was made. We calculated the average daily trade volume for each exchange by tallying the total number of Bitcoins converted into all currencies handled by the exchange for the days the exchange was operational.

We also calculate the "lifetime" of each exchange, that is, the number of days the exchange is operational, denoted by the difference between the first and last observed trade. We deem an exchange to have closed if it had not made a trade in at least two weeks before the end of data collection. We further inspected the existence of a report on the Bitcoin Wiki [11] or on Bitcoin forums [12] to confirm closure, and determine whether closure was caused by a security breach (e.g., hack or fraud). We also checked for reports on whether or not investors were repaid following the exchange's closure.

Finally, to assess regulatory impact, we attempted to identify the country where each exchange is based. We then used an index (ranging between 0 and 49) computed by World Bank economists [13] to identify each country's compliance with "Anti-Money-Laundering and Combating the Financing of Terrorism" (AML-CFT) regulations [13].

2.2 Summary Statistics

Table 1 lists all 40 known Bitcoin currency exchanges, along with relevant facts about whether the exchange later closed. Nine exchanges experienced security breaches,

Table 1. Bitcoin exchange indicators. "Origin" denotes the jurisdiction under which the exchange operates, "AML," the extent to which the exchange's jurisdiction has implemented "Anti-Money Laundering and Combating the Financing of Terrorism" international standards [13]. "Risk ratio" is the relative risk of exchange failure based on the Cox proportional hazards model (Section 3).

Exchange	Origin	Dates Active	Daily vol.	Closed?	Breached?	Repaid?	AML	Risk Ratio
BitcoinMarket	US	4/10 – 6/11	2454	yes	yes	–	34.3	1.12
Bitomat	PL	4/11 – 8/11	758	yes	yes	yes	21.7	1.28
FreshBTC	PL	8/11 – 9/11	3	yes	no	–	21.7	2.01
Bitcoin7	US/BG	6/11 – 10/11	528	yes	yes	no	33.3	1.59
ExchangeBitCoins.com	US	6/11 – 10/11	551	yes	no	–	34.3	0.65
Bitchange.pl	PL	8/11 – 10/11	380	yes	no	–	21.7	0.61
Brasil Bitcoin Market	BR	9/11 – 11/11	0	yes	no	–	24.3	3.85
Aqoin	ES	9/11 – 11/11	11	yes	no	–	30.7	1.57
Global Bitcoin Exchange	?	9/11 – 1/12	14	yes	no	–	27.9	1.45
Bitcoin2Cash	US	4/11 - 1/12	18	yes	no	–	34.3	1.47
TradeHill	US	6/11 - 2/12	5082	yes	yes	yes	34.3	0.94
World Bitcoin Exchange	AU	8/11 – 2/12	220	yes	yes	no	25.7	1.80
Ruxum	US	6/11 – 4/12	37	yes	no	yes	34.3	1.24
btctree	US/CN	5/12 – 7/12	75	yes	no	yes	29.2	0.98
btcex.com	RU	9/10 – 7/12	528	yes	no	no	27.7	0.61
IMCEX.com	SC	7/11 – 10/12	2	yes	no	–	11.9	1.88
Crypto X Change	AU	11/11 – 11/12	874	yes	no	–	25.7	0.53
Bitmarket.eu	PL	4/11 – 12/12	33	yes	no	no	21.7	1.09
bitNZ	NZ	9/11 – pres.	27	no	no	–	21.3	1.14
ICBIT Stock Exchange	SE	3/12 – pres.	3	no	no	–	27.0	2.15
WeExchange	US/AU	10/11 – pres.	2	no	no	–	30.0	2.23
Vircurex	US?	12/11 – pres.	6	no	yes	–	27.9	4.41
btc-e.com	BG	8/11 – pres.	2604	no	yes	yes	32.3	1.08
Mercado Bitcoin	BR	7/11 – pres.	67	no	no	–	24.3	0.95
Canadian Virtual Exchange	CA	6/11 – pres.	832	no	no	–	25.0	0.53
btcchina.com	CN	6/11 – pres.	473	no	no	–	24.0	0.60
bitcoin-24.com	DE	5/12 – pres.	924	no	no	–	26.0	0.52
VirWox	DE	4/11 – pres.	1668	no	no	–	26.0	0.45
Bitcoin.de	DE	8/11 – pres.	1204	no	no	–	26.0	0.49
Bitcoin Central	FR	1/11 – pres.	118	no	no	–	31.7	0.91
Mt. Gox	JP	7/10 – pres.	43230	no	yes	yes	22.7	0.49
Bitcurex	PL	7/12 – pres.	157	no	no	–	21.7	0.76
Kapiton	SE	4/12 – pres.	160	no	no	–	27.0	0.80
bitstamp	SL	9/11 – pres.	1274	no	no	–	35.3	0.54
InterSango	UK	7/11 – pres.	2741	no	no	–	35.3	0.45
Bitfloor	US	5/12 – pres.	816	no	yes	no	34.3	1.45
Camp BX	US	7/11 – pres.	622	no	no	–	34.3	0.63
The Rock Trading Company	US	6/11 – pres.	52	no	no	–	34.3	1.14
bitme	US	7/12 – pres.	77	no	no	–	34.3	1.04
FYB-SG	SG	1/13 – pres.	3	no	no	–	33.7	2.23

caused either by hackers or other criminal activity. Five of these exchanges subsequently closed, but four have survived so far (Mt. Gox, `btc-e.com`, Bitfloor, and Vircurex). Another 13 closed without experiencing a publicly-announced breach.

The popularity of exchanges varied greatly, with 25% of exchanges processing under 25 Bitcoins each day on average, while the most popular exchange, Mt. Gox, has averaged daily transactions exceeding 40 000 BTC. The median daily transactions carried out by exchanges is 290, while the mean is 1 716.

One key factor affecting the risk posed by exchanges is whether or not its customers are reimbursed following closure. We must usually rely on claims by the operator and investors if they are made public. Of the 18 exchanges that closed, we have found evidence on whether customers were reimbursed in 11 cases. Five exchanges have not

reimbursed affected customers, while six claim to have done so. Thus, the risk of losing funds stored at exchanges is real but uncertain.

As a first approximation, the failure rate of Bitcoin exchanges is 45%. The median lifetime of exchanges is 381 days. These summary statistics obscure two key facts: exchanges are opened at different times and so their maximum potential lifetimes vary, and a majority of exchanges remain viable at the end of our observation period. Survival analysis can properly account for this.

3 Survival Analysis of Exchange Closure

We use survival analysis to estimate the time it takes for Bitcoin exchanges to close and to identify factors that can trigger or stave off closure. Robust estimation requires considering that some exchanges remain open at the end of our measurement interval ("censored" data points). Two mathematical functions are commonly used. First, a survival function $S(t)$ measures the probability that an exchange will continue to operate longer than for t days. Second, a hazard function $h(t)$ measures the instantaneous risk of closure at time t. To identify factors affecting an exchange's survival time, we use a Cox proportional hazards model [14], rather than traditional linear regression. We can also estimate the survival function using the best-fit Cox model.

3.1 Statistical Model

We hypothesize that three variables affect the survival time of a Bitcoin exchange:

Average daily transaction volume: an exchange can only continue to operate if it is profitable, and profitability usually requires achieving scale in the number of fee-generating transactions performed. We expect that exchanges with low transaction volume are more likely to shut down. We use a log-transformation of the transaction volume given how skewed transaction volumes are.

Experiencing a security breach: suffering a security breach can erase profits, reduce cash flow, and scare away existing and prospective customers. We thus expect breached exchanges to be more likely to subsequently close.

AML/CFT compliance: some Bitcoin exchanges complain of being hassled by financial regulators. Thus, exchanges operating in countries with greater emphasis on anti-money laundering efforts may be pressured into shutting down.

We then construct a corresponding proportional hazards model [14]:

$$h_i(t) = h_0(t) \exp(\beta_1 \log(\text{Daily vol.})_i + \beta_2 \text{Breached}_i + \beta_3 \text{AML}_i).$$

Here, $h_i(t)$ is the hazard rate for exchange i, $\log(\text{Daily vol.})_i$ is the transaction volume at exchange i, Breached_i indicates whether exchange i suffered a security breach, and AML_i denotes the AML/CFT compliance score for the exchange's country of incorporation. $\beta_1, \beta_2, \beta_3$ are best-fit constants, and $h_0(t)$ is the unspecified baseline hazard.

3.2 Results

The best-fit Cox model is:

		coef.	exp(coef.)	Std. Err.	Significance
$\log(\text{Daily vol.})_i$	β_1	-0.173	0.840	0.072	$p = 0.0156$
Breached_i	β_2	0.857	2.36	0.572	$p = 0.1338$
AML_i	β_3	0.004	1.004	0.042	$p = 0.9221$
log-rank test: Q=7.01 ($p = 0.0715$), $R^2 = 0.145$					

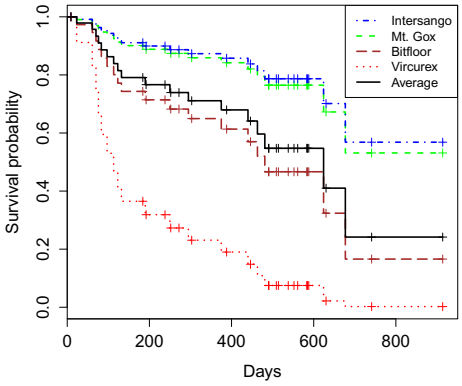


Fig. 1. Empirically-derived survival probability function for Bitcoin exchanges

The daily volume is negatively associated with the hazard rate ($\beta_1 = -0.173$): doubling the daily volume rate corresponds to a 16% reduction in the hazard rate ($\exp(\beta_1) = 0.84$) . Thus, exchanges that process more transactions are less likely to shut down.

Suffering a breach is positively correlated with hazard, but with a p-value of 0.1338, this correlation falls just short of being statistically significant at this time. Given that just nine exchanges publicly reported suffering breaches and only five later closed, it is not surprising that the association is not yet robust.

Finally, the anti-money laundering indicator has no measurable correlation with exchange closure. This could suggest that regulatory oversight is not triggering closures, but it could also reflect that the indicator itself does not accurately convey differences in attitudes the world's financial regulators have taken towards Bitcoin.

Figure 1 plots the best-fit survival function according to the Cox model. The survival function precisely quantifies the probability of failure within a given amount of time. This can help Bitcoin investors weigh their risks before putting money into an exchange-managed account. The black solid line plots the estimated survival function for the best fit parameters outlined above for the mean values of exchange volume,

whether a site has been hacked, and AML score. For instance, $S(365) = 0.711$ with 95% confidence interval $(0.576, 0.878)$: there is a 29.9% chance a new Bitcoin exchange will close within a year of opening (12.2%–42.4% with 95% confidence).

Figure 1 also includes survival functions for several Bitcoin exchanges. These are calculated based on the exchange's values for parameters in the Cox model (e.g., transaction volume). For instance, Mt. Gox and Intersango are less likely to close than other exchanges. Meanwhile, Vircurex (established in December 2011 and breached in January 2013) continues to operate despite low transaction volumes and a survival function that estimates one-year survival at only 20%.

The right-most column in Table 1 presents relative risk ratios for all exchanges. These indicate how the hazard function for each exchange compares to the baseline hazard. Values less than 1 indicate that the exchange is at below-average risk for closure; values greater than 1 denote above-average risk. Of course, any exchange may close, but those with lower risk ratios have a better chance of remaining operational. For instance, while 6 of the 18 closed exchanges have risk ratios below 1, 12 of the 22 open ones do.

4 Regression Analysis of Exchange Breaches

While we cannot conclude that security breaches trigger exchanges to close, we can examine whether any other factors affect the likelihood an exchange will suffer a breach.

4.1 Statistical Model

We use a logistic regression model with a dependent variable denoting whether or not an exchange experiences a breach. We hypothesize that two explanatory variables influence whether a breach occurs:

Average daily transaction volume: bigger exchanges make richer targets. As an exchange processes more transactions, more wealth flows into its accounts. Consequently, we expect that profit-motivated criminals are naturally drawn to exchanges with higher average daily transaction volumes.

Months operational: every day an exchange is operational is another day that it could be hacked. Longer-lived exchanges, therefore, are more exposed to breaches.

The model takes the following form:

$$\log\left(p_b/(1-p_b)\right) = c_0 \; + c_1 \; \log(\text{Daily vol.}) + c_2 \; \text{months operational} + \varepsilon.$$

The dependent variable p_b is the probability that an exchange experiences a security breach, c_0, c_1, c_2 are best-fit constants, $\log(\text{daily vol.})$ is the log-transformed daily transaction volume at the exchange, # months operational is the time (in months) that the exchange has been operational, and ε is an error term.

4.2 Results

The logistic regression yields the following results:

	coef.	Odds-ratio	95% conf. int.	Significance
Intercept	-4.304	0.014	(0.0002,0.211)	$p = 0.0131$
log(Daily vol.)	0.514	1.672	(1.183,2.854)	$p = 0.0176$
Months operational	-0.104	0.901	(0.771,1.025)	$p = 0.1400$
Model fit: $\chi^2 = 10.3, p = 0.00579$				

Transaction volume is positively correlated with experiencing a breach. Months operational, meanwhile, is negatively correlated with being breached, but the association just falls short of statistical significance ($p = 0.14$). Thus, we face a conundrum: according to the results of Section 3, high-volume exchanges are less likely to close but *more likely* to experience a breach. Bitcoin holders can choose to do business with less popular exchanges to reduce the risk of losing money due to a breach, or with more popular exchanges that may be breached, but are less likely to shut down without warning.

Figure 2 takes the coefficients for a best-fit logit model and plots the probability that an exchange operational for the average duration of one year will be breached as transaction volume increases. For example, exchanges handling 275 Bitcoins' worth of transactions each day have a 20% chance of being breached, compared to a 70% chance for exchanges processing daily transactions worth 5570 Bitcoins.

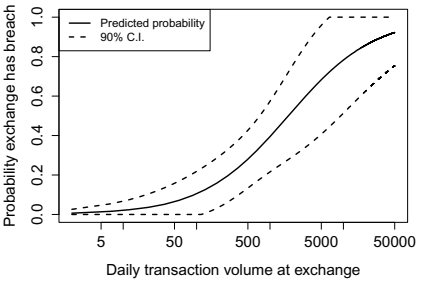


Fig. 2. Probability that an exchange will experience a breach as the average volume of Bitcoins exchanged varies, according to the best-fit logit model.

5 Related Work

Bitcoin's recent success has piqued the interest of a number of researchers in studying it. A couple of works looked into the cryptographic aspects [15, 16, 17] and ways to either improve or build on Bitcoin. Another set of papers explored the Bitcoin network of transactions [18, 19], and documented practical uses of Bitcoin [7]. Others yet investigated economic considerations regarding, in particular, the economic costs of proof-of-work mechanisms such as Bitcoin [20]. Different from these related efforts, we believe our paper is the first to focus on Bitcoin exchanges.

6 Discussion

In this paper, we empirically investigated two risks linked to Bitcoin exchanges. We conducted a survival analysis on 40 Bitcoin exchanges, which found that an exchange's average transaction volume is negatively correlated with the probability it will close prematurely. We also presented a regression analysis which found that, in contrast to the survival analysis, transaction volume is positively correlated with experiencing a breach. Hence, the continued operation of an exchange depends on running a high transaction volume, which makes the exchange a more valuable target to thieves.

Our statistical analysis presents three notable limitations. First, there is substantial randomness affecting when an exchange closes or is breached that is not captured by our model. Future work might investigate additional explanatory variables, such as the exchange reputation. Second, some explanatory variables did not achieve statistical significance due to the data set's modest size. The analysis is worth revisiting as time passes and new exchanges are opened and old ones close. Third, some indicators may need to be changed as Bitcoin grows. For instance, rapid increases in transaction volumes may render long-term unweighted averages less meaningful.

Finally, we focused on economic considerations, such as closure risks, that a rational actor would want to estimate before investing in a given exchange. However, reducing Bitcoin to a mere speculative instrument misses an important piece of the puzzle. Most Bitcoin users are early adopters, often motivated by non-economic considerations. For instance, Silk Road users, who constitute a large share of the Bitcoin economy [7], may shy away from exchanges that require identification, and instead prefer assurances of anonymity. This may in turn lead them to use exchanges posing greater economic risk. Studying the unique characteristics of Bitcoin users and investors, compared to typical foreign exchange traders, is an avenue for future work we think well worth exploring.

Acknowledgments. We thank Rainer Böhme and our anonymous reviewers for their extensive feedback on an earlier version of this paper. This research was partially supported by the National Science Foundation under ITR award CCF-0424422 (TRUST).

References

1. Birch, D., McEvoy, N.: Electronic cash – technology will denationalise money. In: Luby, M., Rolim, J.D.P., Serna, M. (eds.) FC 1997. LNCS, vol. 1318, pp. 95–108. Springer, Heidelberg (1997)
2. Chaum, D.: Achieving electronic privacy. Scientific American, 96–101 (August 1992)
3. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009), http://www.bitcoin.org/bitcoin.pdf
4. Bitcoin Watch, http://bitcoinwatch.com/ (last accessed January 27, 2013)
5. Leyden, J.: Linode hackers escape with $70k in daring Bitcoin heist. The Register (March 2012), http://www.theregister.co.uk/2012/03/02/linode_bitcoin_heist/
6. Lee, T.: Hacker steals $250k in bitcoins from online exchange bitfloor. Ars Technica (September 2012), http://arstechnica.com/tech-policy/2012/09/hacker-steals-250k-in-bitcoins-from-online-exchange-bitfloor/

7. Christin, N.: Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. Technical Report CMU-CyLab-12-018, Carnegie Mellon University (2012)
8. Jeffries, A.: Suspected multi-million dollar Bitcoin pyramid scheme shuts down, investors revolt. The Verge (August 2012), http://www.theverge.com/2012/8/27/3271637/bitcoin-savings-trust-pyramid-scheme-shuts-down
9. Anderson, R.: Closing the phishing hole: Fraud, risk and nonbanks. In: Federal Reserve Bank of Kansas City – Payment System Research Conferences (2007)
10. Moore, T., Han, J., Clayton, R.: The postmodern Ponzi scheme: Empirical analysis of high-yield investment programs. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 41–56. Springer, Heidelberg (2012)
11. Bitcoin wiki, https://bitcointalk.org/ (last accessed January 27, 2013)
12. Bitcoin forums, https://en.bitcoin.it/ (last accessed January 27, 2013)
13. Yepes, C.: Compliance with the AML/CFT international standard: Lessons from a cross-country analysis. IMF Working Papers 11/177, International Monetary Fund (July 2011)
14. Cox, D.: Regression models and life-tables. Journal of the Royal Statistics Society, Series B 34, 187–220 (1972)
15. Clark, J., Essex, A.: CommitCoin: Carbon dating commitments with bitcoin (short paper). In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 390–398. Springer, Heidelberg (2012)
16. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to better – how to make Bitcoin a better currency. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 399–414. Springer, Heidelberg (2012)
17. Karame, G., Androulaki, E., Capkun, S.: Two Bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin. In: Proc. ACM CCS, Raleigh, NC (October 2012)
18. Ron, D., Shamir, A.: Quantitative analysis of the full Bitcoin transaction graph, Cryptology ePrint Archive, Report 2012/584 (October 2012)
19. Reid, F., Harrigan, M.: An analysis of anonymity in the Bitcoin system, arXiv:1107.452a4v2 [physics.soc-ph] (May 2012), http://arxiv.org/abs/1107.4524
20. Becker, J., Breuker, D., Heide, T., Holler, J., Rauer, H.P., Böhme, R.: Can we afford integrity by proof-of-work? Scenarios inspired by the Bitcoin currency. In: Proc. WEIS, Berlin, Germany (June 2012)

Evaluating User Privacy in Bitcoin

Elli Androulaki[1], Ghassan O. Karame[2], Marc Roeschlin[1],
Tobias Scherer[1], and Srdjan Capkun[1]

[1] ETH Zurich, 8092 Zuerich, Switzerland
{elli.androulaki,capkuns}@inf.ethz.ch,
{romarc,schereto}@student.ethz.ch
[2] NEC Laboratories Europe, 69115 Heidelberg, Germany
ghassan.karame@neclab.eu

Abstract. Bitcoin is quickly emerging as a popular digital payment system. However, in spite of its reliance on pseudonyms, Bitcoin raises a number of privacy concerns due to the fact that all of the transactions that take place are publicly announced in the system.

In this paper, we investigate the privacy provisions in Bitcoin when it is used as a primary currency to support the daily transactions of individuals in a university setting. More specifically, we evaluate the privacy that is provided by Bitcoin *(i)* by analyzing the genuine Bitcoin system and *(ii)* through a simulator that faithfully mimics the use of Bitcoin within a university. In this setting, our results show that the profiles of almost 40% of the users can be, to a large extent, recovered even when users adopt privacy measures recommended by Bitcoin. To the best of our knowledge, this is the first work that comprehensively analyzes, and evaluates the privacy implications of Bitcoin.

Keywords: Bitcoin, user privacy, privacy definitions, experimental evaluation.

1 Introduction

Bitcoin [6] is an emerging digital currency that is currently being integrated across a number of businesses [1] and exchange markets.

Bitcoin is a Proof-of-Work (PoW) based currency that allows users to generate digital coins by performing computations. Bitcoin users execute payments by digitally signing their transactions and are prevented from double-spending their coins (i.e., signing-over the same coin to two different users) through a distributed time-stamping service [6]. This service operates on top of the Bitcoin Peer-to-Peer (P2P) network and ensures that all transactions and their order of execution are available to the public. To strengthen the privacy of its users, Bitcoin users participate in transactions using pseudonyms—referred to as *Bitcoin addresses*. Generally, each user has hundreds of different Bitcoin addresses that are all stored and transparently managed by its client.

In spite of the reliance on pseudonyms, the public timestamping mechanism of Bitcoin raises serious concerns with respect to the privacy of users. In fact, given that Bitcoin transactions basically consist of a chain of digital signatures, the expenditure of individual coins can be publicly tracked [16].

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 34–51, 2013.

In this work, we evaluate the privacy that is provided by Bitcoin when it is used to support the daily transactions of individuals in a university setting. This is achieved *(i)* by investigating the behavior of Bitcoin client and exploiting its properties, and *(ii)* through a novel simulator that mimics the use of Bitcoin as the primary currency within a university setting. Finally, we discuss possible measures that can be used to enhance the privacy of users in Bitcoin. To the best of our knowledge, this is the first work that analyzes, and evaluates the privacy provisions in Bitcoin. More specifically, our contributions in this paper can be summarized as follows:

- We adapt existing privacy notions to the Bitcoin context and we investigate the privacy-enhancing measures that are used in current Bitcoin implementations.
- We design and implement a simulator that faithfully emulates the functionality of Bitcoin. Our simulator also provides us with "ground truth" information that corresponds to the use of Bitcoin in a university setting.
- We investigate the privacy provisions of Bitcoin in a realistic university setting through our simulator. Our results show that the profiles of 40% of the university users can be constructed using behavior-based clustering techniques with 80% accuracy, even in the case when users manually transfer their Bitcoins among their addresses in an attempt to enhance their privacy.
- We discuss possible measures that can be used by Bitcoin developers to enhance the privacy of users in Bitcoin.

The remainder of this paper is organized as follows. In Section 2, we present a background on Bitcoin. In Section 3, we introduce metrics that we use to measure privacy in Bitcoin. In Section 4, we present our Bitcoin simulator and our evaluation results. In Section 5, we discuss the implications of our findings and we explore possible countermeasures for enhancing privacy in Bitcoin. In Section 6, we overview the related work and we conclude the paper in Section 7.

2 Background on Bitcoin

Bitcoin is a decentralized P2P payment system [6] that relies on PoW. Payments are performed by generating *transactions* that transfer Bitcoin coins (BTCs) between Bitcoin users. Users participate in transactions using pseudonyms—referred to as *Bitcoin addresses*. Generally, each user has hundreds of different Bitcoin addresses that are all stored and managed by its (digital) *wallet*. Each address is mapped through a transformation function to a unique public/private key pair. These keys are used to authorize the transfer of the ownership of BTCs among addresses.

Transactions: Users transfer coins (BTCs) to each other by issuing a transaction. A transaction is formed by digitally signing a hash of the transaction through which a BTC was acquired. Given that in Bitcoin there is one-to-one correspondence between signature public keys and addresses, a transaction taking place between two addresses a_{S} and a_{R} has the following form: $\tau(a_{\mathsf{S}} \rightarrow a_{\mathsf{R}}) = \{\text{source}, B, a_{\mathsf{R}}, \ \mathsf{SIG}_{\text{sk}_{a_{\mathsf{S}}}}(\text{source}, B, a_{\mathsf{R}})\}$.

Here, $\mathsf{SIG}_{\mathrm{sk}_{a_S}}$ is the signature using the private key sk_{a_S} that corresponds to the public key associated with the a_S, B is the amount of BTCs transferred, and source is a reference to the most recent transaction that a_S acquired the B BTCs from. After their creation, Bitcoin transactions are released in the Bitcoin network. Once the validity of τ is *confirmed* (as described later in this section), a_R can subsequently use this transaction as a reference to spend the acquired BTCs. Consequently, Bitcoin transactions form a public record and any user can verify the authenticity of a BTC by checking the chain of signatures of the transactions in which the BTC was involved.

In the case where a_S needs to spend a value that exceeds the maximum value of a BTC that it possesses, then its Bitcoin client will automatically combine a number of its BTCs as multiple inputs of the same outgoing transaction. We analyze the impact of "multi-input" transactions on the privacy of users in Bitcoin in Section 4.1. Figure 4 in Appendix A depicts an example of multiple-input transactions.

Shadow Addresses: In the current implementation of Bitcoin, a new address—the "shadow" address [1]—is automatically created and used to collect back the "change" that results from any transaction issued by the user. Besides the reliance on pseudonyms, shadow addresses constitute the only mechanism adopted by Bitcoin to strengthen the privacy of its users.

Confirmation of Transactions: As mentioned before, transactions are broadcasted in the Bitcoin network and are subject to validity checks by users in the system. Valid transactions are included by a special type of users, the *miners*, in Bitcoin *blocks* that are also broadcasted in the network. More specifically, to generate a new block, miners must find a nonce value that, when hashed with additional fields (i.e., the Merkle hash of all valid and received transactions, the hash of the previous block, and a timestamp), results in a value below a given threshold. If such a nonce is found, miners then include it in a block thus allowing any entity to verify the PoW. Upon successful block generation, a miner is granted a number of new BTCs. This provides an incentive for miners to continuously support Bitcoin. Table 2 in Appendix B shows the information included in Bitcoin block number 80,000 as reported in the Bitcoin block explorer [2]. The resulting block is forwarded to all users in the network, who can then check its correctness by verifying the hash computation. If the block is deemed to be "valid"[1], then the users append it to their previously accepted blocks, thus growing the Bitcoin block *chain*. Bitcoin relies on this mechanism to resist double-spending attacks; for malicious users to double-spend a BTC without being detected, they would not only have to redo all the work required to compute the block where that BTC was spent, but also they need to recompute all the subsequent blocks in the chain.

3 Modelling Privacy in Bitcoin

In this section, we introduce our adversarial model and we define a number of metrics that can be used to quantify privacy in Bitcoin.

[1] That is, the block contains correctly formed transactions that have not been previously spent, and has a correct PoW.

3.1 Adversarial Model

We observe the public log of Bitcoin, denoted by pubLog, within a period of time Δt. During this period, $\mathrm{n_U}$ users, $\mathrm{U} = \{u_1, u_2, \ldots, u_{\mathrm{n_U}}\}$, participate in pubLog through a set of $\mathrm{n_A}$ addresses: $\mathsf{A} = \{a_1, a_2, \ldots, a_{\mathrm{n_A}}\}$. We assume that within Δt, $\mathrm{n_T}$ transactions have taken place as follows: $\mathrm{T} = \{\tau_1(\mathsf{S}_1 \rightarrow \mathsf{R}_1), \ldots, \tau_{\mathrm{n_T}}(\mathsf{S}_{\mathrm{n_T}} \rightarrow \mathsf{R}_{\mathrm{n_T}})\}$, where $\tau_i(\mathsf{S}_i \rightarrow \mathsf{R}_i)$ denotes a transaction with (unique) ID number i and S_i and R_i denote the sets of senders' addresses and recipients' addresses, respectively.

We assume that the adversary $\mathcal{A}$ is motivated to acquire information about the addresses/transactions pertaining to all or a subset of Bitcoin users. As such, $\mathcal{A}$ does not only have access to pubLog, but is also *part of the Bitcoin system* and can also incur one or more transactions through Bitcoin. Furthermore, we assume that $\mathcal{A}$ can have access to the (public) addresses of some vendors along with (statistical) information such as the pricing of items or the number of their clients within a specified amount of time. We, however, assume that $\mathcal{A}$ is computationally bounded and as such cannot construct ill-formed Bitcoin blocks, double-spend confirmed transactions, or forge signatures, etc.

Throughout this paper, we consider the "privacy" measures adopted by existing Bitcoin clients. Namely, we assume that *(i)* new shadow addresses are used to collect change that results from issued transactions, *(ii)* users own many Bitcoin addresses, and *(iii)* users are encouraged to frequently change their addresses (by transferring some of their BTCs to newly created addresses); this conforms with the current practices adopted in Bitcoin.

3.2 Quantifying Privacy in Bitcoin

In this section, we introduce two different notions of Bitcoin privacy, activity unlinkability and profile indistinguishability and we provide metrics to appropriately quantify these notions.

Activity unlinkability refers to the fact that an adversary $\mathcal{A}$ should not be able to link two different addresses (address unlinkability) or transactions (transaction unlinkability) pertaining to a user of *her choice*. By design, if $\mathcal{A}$ can link two Bitcoin addresses to the same user, then she can also link all the transactions that these addresses participate in. Therefore, we focus our analysis on address unlinkability. On the other hand, *profile indistinguishability* refers to the (in-)ability of $\mathcal{A}$ to reconstruct the profiles of *all the users* that participate in pubLog. Profiles, here, consist of the set of addresses (address-based profiles) or set of transactions (transaction-based profiles) of Bitcoin users. As such, profile indistinguishability property is a stronger privacy notion than the activity unlinkability as it assesses the concealment of the profiles of *all users* in Bitcoin. Unlike the activity unlinkability case, here we also account for transaction-based profiles. This is due to the fact that address-based and transaction-based profiles are not equivalent when it comes to modeling user profiles in Bitcoin[2].

In the following, we provide definitions on both privacy notions, and we rely on these definitions to provide metrics for them. In particular, we define address unlink-

[2] An adversary can perform well in address-based profiling but not in transaction-based profiling: she may correctly profile addresses of users that are involved in few transactions and "miss-classify" few addresses who participate in many transactions.

ability and profiling indistinguishability through the AddUnl and the $\mathrm{ProfInd}$ games, respectively. We quantify these notions by assessing the advantage of an adversary $\mathcal{A}$ in winning these games over an adversary who responds to all game challenges with random guesses, $\mathcal{A}^{\mathcal{R}}$. We assume that $\mathcal{A}$ has access to pubLog and that both $\mathcal{A}$ and $\mathcal{A}^{\mathcal{R}}$ have gathered (the same) a-priori knowledge $\mathcal{K}_{\mathcal{A}}$ with respect to correlations of a subset of addresses (i.e., whether these addresses belong to the same user or not).

Activity Unlinkability (Address Unlinkability): We construct the address unlinkability game in Bitcoin, AddUnl, as follows. $\mathcal{A}$ chooses an address $a_0 \in \mathsf{pubLog}$ chosen among the addresses that appear in pubLog and sends it to the challenger $\mathcal{C}$. If the owner of a_0 does not have any other Bitcoin address, then $\mathcal{A}$ wins. Otherwise, the challenger $\mathcal{C}$ randomly chooses a bit b. If $b = 1$, then $\mathcal{C}$ randomly chooses another address $a_1 \in \mathsf{pubLog}$ such that a_0, a_1 belong to the same user; otherwise, $\mathcal{C}$ randomly chooses a_1 such that the two addresses are owned by different users. The challenger sends $\langle a_0, a_1 \rangle$ to $\mathcal{A}$, who responds with her estimate b' on whether the two addresses belong to the same user. $\mathcal{A}$ wins the game if she answers correctly, i.e., $b = b'$. We say that Bitcoin satisfies address unlinkability if for all p.p.t. adversaries $\mathcal{A}$ and $\forall \langle a_0, a_1 \rangle$, $\mathcal{A}$ has only a negligible advantage over $\mathcal{A}^{\mathcal{R}}$ in winning, i.e., if:

$$\mathrm{Prob}[b' \leftarrow \mathcal{A}(\mathsf{pubLog}, \mathcal{K}_{\mathcal{A}}) : b = b'] - \mathrm{Prob}[b' \leftarrow \mathcal{A}^{R}(\mathcal{K}_{\mathcal{A}}) : b = b'] \leq \varepsilon,$$

where ε is negligible.

Quantifying Address Unlinkability: In Section 4.1, we show that due to inherent properties of the Bitcoin protocol and client, $\mathcal{A}$ can succeed with a considerable probability in winning the above AddUnl game. In what follows, we measure the *degree* to which Bitcoin addresses can be *linked* to the same user.

To do so, we express the estimate of $\mathcal{A}$ through an $\mathrm{n_A} \times \mathrm{n_A}$ matrix, $\mathrm{E_{link}}$, where $\mathrm{E_{link}}[i,j] = \{p_{i,j}\}_{i,j \in [1,\mathrm{n_A}]}$. That is, that for every address a_i, $\mathcal{A}$ assesses the probability $p_{i,j}$ with which a_i is owned by the same user as every other address a_j in pubLog. Note that $\mathrm{E_{link}}$ incorporates $\mathcal{K}_{\mathcal{A}}$, and any additional information that $\mathcal{A}$ could extract from pubLog (e.g., by means of clustering, statistical analysis, etc.). Similar to [15], we quantify the success of $\mathcal{A}$ in the AddUnl game as follows. Let $\mathrm{GT_{link}}$ denote the genuine address association matrix, i.e., $\mathrm{GT_{link}}[i,j] = 1$, if a_i and a_j are of the same user and $\mathrm{GT_{link}}[i,j] = 0$ otherwise for all $i, j \in [1, \mathrm{n_A}]$. For each address a_i we compute the error in $\mathcal{A}$'s estimate, i.e., the distance of $\mathrm{E_{link}}[i,*]$ from the genuine association of a_i with the rest of the addresses in pubLog, $||\mathrm{E_{link}}[i,*] - \mathrm{GT_{link}}[i,*]||$, where $||\cdot||$ denotes norm-L1 of the corresponding row-matrix. Thus, the success of $\mathcal{A}$ in AddUnl, $\mathrm{Succ}_{\mathcal{A}}$, can then be assessed through $\mathcal{A}$'s maximum error: $\max_{\forall a_i \notin \mathcal{K}_{\mathcal{A}}} (||\mathrm{E_{link}}[i,*] - \mathrm{GT_{link}}[i,*]||)$.

Similarly, we represent the estimate of $\mathcal{A}^{\mathcal{R}}$ in the AddUnl game for all possible pairs of addresses by the $\mathrm{n_A} \times \mathrm{n_A}$ matrix $\mathrm{E^{\mathcal{R}}_{link}}$ which is constructed as follows. $\mathrm{E}^{\mathcal{R}}[i,j] = \pi_{i,j}$ if $\langle a_i, a_j \rangle \in \mathcal{K}_{\mathcal{A}}$, and $\mathrm{E^{\mathcal{R}}_{link}}[i,j] = \rho + (1-\rho)\frac{1}{2}$ otherwise. Here, $\pi_{i,j}$ represents the probability that addresses a_i a_j correspond to the same user according to $\mathcal{K}_{\mathcal{A}}$, and ρ refers to the fraction of addresses in $\{\mathsf{pubLog}\text{-}\mathcal{K}_{\mathcal{A}}\}$ that cannot be associated to other

addresses (i.e., when their owners have only one address)[3]. For pairs of addresses that are not included in $\mathcal{K}_{\mathcal{A}}$, this probability equals to $\rho + (1-\rho)\frac{1}{2}$.

Given this, we measure the degree of address unlinkability in Bitcoin against $\mathcal{A}$ by measuring address linkability, i.e., by evaluating the additional success that $\mathcal{A}$ can achieve from pubLog, when compared to $\mathcal{A}^{\mathcal{R}}$. We call this advantage $\mathrm{Link}^{\mathrm{abs}}_{\mathcal{A}}$: $\mathrm{Link}^{\mathrm{abs}}_{\mathcal{A}} = \mathrm{Succ}_{\mathcal{A}} - \mathrm{Succ}_{\mathcal{A}^{\mathcal{R}}}$, and its normalized version $\mathrm{Link}_{\mathcal{A}}$: $\mathrm{Link}_{\mathcal{A}} = \frac{\mathrm{Succ}_{\mathcal{A}} - \mathrm{Succ}_{\mathcal{A}^{\mathcal{R}}}}{\mathrm{Succ}_{\mathcal{A}^{\mathcal{R}}}}$.[4]

User Profile Indistinguishability: The profile indistinguishability property is a stronger privacy notion than the activity unlinkability as it refers to the concealment of all Bitcoin user profiles. Here, we require that an adversary is not able to group addresses or transactions corresponding to Bitcoin users correctly. As mentioned before, we account for the indistinguishability of the profiles of users, assuming address-based and transaction-based profiles.

We construct the $\mathrm{ProfInd}$ game as follows. Here, the challenger $\mathcal{C}$ sends to $\mathcal{A}$ the number of users $\mathrm{n_U}$ in pubLog-$\mathcal{K}_{\mathcal{A}}$. $\mathcal{A}$ responds with $\mathrm{n_U}$ (non-overlapping) sets of addresses (or transactions) and with $\mathrm{E_{prof}} = \{\mathrm{g}_i\}_{i=1}^{\mathrm{n_U}}$ representing the estimate on the profile of all users in the system. Let $\mathrm{GT_{prof}} = \{\mathrm{gt}_i\}_{i=1}^{\mathrm{n_U}}$ represent the genuine grouping of addresses (or transactions) to users. That is, $\mathrm{GT_{prof}} = \{a_{u_i}\}_{i=1}^{\mathrm{n_U}}$ for address-based profiles, and $\mathrm{GT_{prof}} = \{\tau_{u_i}\}_{i=1}^{\mathrm{n_U}}$, for transaction-based profiles, where a_{u_i} and τ_{u_i} represent the sets of addresses and transactions respectively of user u_i. Clearly, $\mathcal{A}$ wins if she guesses correctly, i.e., if $\mathrm{E_{prof}} \equiv \mathrm{GT_{prof}}$.

We say that a system satisfies the profile indistinguishability property if there is no p.p.t. adversary $\mathcal{A}$ who wins the $\mathrm{ProfInd}$ game with better probability than $\mathcal{A}^{\mathcal{R}}$, i.e.:

$$\forall \text{ p.p.t. } \mathcal{A}: \quad \mathrm{Prob}[\mathrm{E_{prof}} \leftarrow \mathcal{A}(\mathsf{pubLog}, \mathrm{n_U}) : \mathrm{E_{prof}} \equiv \mathrm{GT_{prof}}] - \mathrm{Prob}[\mathrm{E^{\mathcal{R}}_{prof}} \leftarrow \mathcal{A}^{\mathcal{R}}(\mathrm{n_U}) : \mathrm{E^{\mathcal{R}}_{prof}} \equiv \mathrm{GT_{prof}}] \leq \varepsilon.$$

Quantifying User Profile Indistinguishability: We now proceed to quantify $\mathcal{A}$'s advantage in winning the $\mathrm{ProfInd}$ game by measuring the similarity of $\mathcal{A}$'s estimate $\mathrm{E_{prof}}$ from the genuine grouping of profiles $\mathrm{GT_{prof}}$, $\mathrm{Sim}(\mathrm{E_{prof}}, \mathrm{GT_{prof}})$, where the similarity function Sim ranges in $[0, 1]$. As in address unlinkability, we measure profile indistinguishability against $\mathcal{A}$ by measuring the degree of profile distinguishability that $\mathcal{A}$ can achieve, i.e., we assess the advantage of $\mathcal{A}$ in approximating $\mathrm{GT_{prof}}$ over $\mathcal{A}^{\mathcal{R}}$ by $\mathrm{Prof}_{\mathcal{A}} = \mathrm{Sim}(\mathrm{E_{prof}}, \mathrm{GT_{prof}}) - \mathrm{Sim}(\mathrm{E^{\mathcal{R}}_{prof}}, \mathrm{GT_{prof}})$.[5]

We quantify $\mathrm{Sim}(\mathrm{E_{prof}}, \mathrm{GT_{prof}})$ and $\mathrm{Prof}_{\mathcal{A}}$ by relying on two commonly used entropy based distance metrics, namely: the *Normalized Mutual Information* (NMI) and the *Adjusted Mutual Information*, (AMI). NMI assesses the similarity of two groupings of the same items (in our case, $\mathrm{E_{prof}}$ and $\mathrm{GT_{prof}}$), and takes higher values (1) the more identical the groupings are [19, 20]. On the other hand, AMI assesses the advantage of $\mathcal{A}$ in winning the $\mathrm{ProfInd}$ game. More specifically, given the two groupings

[3] In our experiments in Section 4, we assume that ρ is negligible since there are at least two addresses per user in Bitcoin (a real address and a "shadow" address).

[4] We say that Bitcoin satisfies μ-address unlinkability if $\forall$ p.p.t. algorithms $\mathcal{A}$ and the corresponding $\mathcal{A}^{\mathcal{R}}$: $\mathrm{Prob}[\mathrm{E_{link}} \leftarrow \mathcal{A}(\mathsf{pubLog}, \mathcal{K}_{\mathcal{A}}), \mathrm{E^{\mathcal{R}}_{link}} \leftarrow \mathcal{A}^{\mathcal{R}}(\mathcal{K}_{\mathcal{A}}) : \mathrm{Link}_{\mathcal{A}} \geq 1 - \mu] \leq \varepsilon$.

[5] We say that a system offers μ-profile indistinguishability, and we write μ-ProfInd, if $\forall$ p.p.t. $\mathcal{A}$: $\mathrm{Prob}[\mathrm{E_{prof}} \leftarrow \mathcal{A}(\mathsf{pubLog}, \mathrm{n_U}), \mathrm{E}^{\mathcal{R}} \leftarrow \mathcal{A}^{\mathcal{R}}(\mathrm{n_U}) : \mathrm{Prof}_{\mathcal{A}} \geq 1 - \mu] \leq \varepsilon$.

$\mathrm{E}_{\mathrm{prof}}$ and $\mathrm{GT}_{\mathrm{prof}}$ AMI approaches 0 when $\mathrm{E}_{\mathrm{prof}}$ is close to random assignment of addresses/transactions to groups, i.e., $\mathrm{E}^{\mathcal{R}}_{\mathrm{prof}}$, and is 1 when $\mathrm{E}_{\mathrm{prof}}$ matches $\mathrm{GT}_{\mathrm{prof}}$ [19,20]. Assuming address-based profiles, NMI and AMI are computed as follows:

$$\mathrm{NMI} = \frac{\mathcal{I}(\mathrm{E}_{\mathrm{prof}}, \mathrm{GT}_{\mathrm{prof}})}{\max(\mathrm{H}(\mathrm{E}_{\mathrm{prof}}), \mathrm{H}(\mathrm{GT}_{\mathrm{prof}}))}, \quad \mathrm{AMI} = \frac{\mathcal{I}(\mathrm{E}_{\mathrm{prof}}, \mathrm{GT}_{\mathrm{prof}}) - \mathcal{E}}{\max(\mathrm{H}(\mathrm{E}_{\mathrm{prof}}), \mathrm{H}(\mathrm{GT}_{\mathrm{prof}})) - \mathcal{E}},$$

where:

$\mathcal{I}(\mathrm{E}_{\mathrm{prof}}, \mathrm{GT}_{\mathrm{prof}}) = \sum_{i=1}^{\mathrm{n_U}} \sum_{j=1}^{\mathrm{n_U}} \frac{n_{(i,j)}}{\mathrm{n_A}} \log(\frac{n_{(i,j)} \cdot \mathrm{n_A}}{n_{(i,*)} n_{(*,j)}})$,

$\mathrm{H}(\mathrm{E}_{\mathrm{prof}}) = -\sum_{i=1}^{\mathrm{n_U}} \frac{n_{(i,*)}}{\mathrm{n_A}} \log(\frac{n_{(i,*)}}{\mathrm{n_A}})$, $\quad \mathrm{H}(\mathrm{GT}_{\mathrm{prof}}) = -\sum_{j=1}^{\mathrm{n_U}} \frac{n_{(*,j)}}{\mathrm{n_A}} \log(\frac{n_{(*,j)}}{\mathrm{n_A}})$,

$\mathcal{E} = \sum_{i=1}^{\mathrm{n_U}} \sum_{j=1}^{\mathrm{n_U}} \sum_{n \in \mathcal{M}} \frac{n}{\mathrm{n_A}} \log(\frac{\mathrm{n_A} n}{n_{(i,*)} n_{(*,j)}}) \frac{n_{(i,*)}! n_{(*,j)}! (\mathrm{n_A} - n_{(i,*)})! (\mathrm{n_A} - n_{(*,j)})!}{\mathrm{n_A}! (n_{(i,*)} - n)! (n_{(*,j)} - n)! (\mathrm{n_A} - n_{(i,*)} - n_{(*,j)} - n)!}$.

Here, $\mathrm{n_A}$ is the number of Bitcoin addresses, $n_{(i,j)}$ is the number of u_i's addresses, which are assigned to group g_j, $n_{(i,*)}$ and $n_{(*,j)}$ are the number of addresses of u_i and g_j respectively. $\mathcal{E}$ reflects the *expected* mutual information between $\mathrm{GT}_{\mathrm{prof}}$ and a random grouping of addresses ($\mathrm{E}^{\mathcal{R}}_{\mathrm{prof}}$). Also, $\mathcal{M} = [\max(n_{(i,*)} + n_{(*,j)} - \mathrm{n_A}, 0), \min(n_{(i,*)}, n_{*,j})]$. Similar calculations can be derived to compute NMI and AMI for transaction-based profiles.

Remark 1. Although NMI and AMI represent sufficiently well the success of $\mathcal{A}$ in profiling all the users in the system, they do not measure the success of the adversary on the profiling of a particular user u_i. In Section 4.4, we measure the success of $\mathcal{A}$ in profiling u_i by assessing the maximum similarity of the addresses (or transactions) of each user u_i with each adversarial cluster g_j i.e., $\max_{\forall j}(\mathrm{Sim}(a_{u_i}, \mathrm{g}_j))$.

4 Evaluating Privacy in Bitcoin

In what follows, we show how the adversary, given pubLog, can gather some knowledge about Bitcoin users by exploiting the properties of existing Bitcoin client implementations. We then evaluate, by means of our Bitcoin simulator, the success of the adversary in the aforementioned privacy games given this knowledge.

4.1 Exploiting Existing Bitcoin Client Implementations

Current Bitcoin client implementations enable $\mathcal{A}$ to link a fraction of Bitcoin addresses that belong to the same user.

Heuristic I—Multi-input Transactions: As mentioned earlier, multi-input transactions occur when u wishes to perform a payment, and the payment amount exceeds the value of each of the available BTCs in u's wallet. In fact, existing Bitcoin clients choose a set of BTCs from u's wallet (such that their aggregate value matches the payment) and perform the payment through multi-input transactions. It is therefore straightforward to conclude that if these BTCs are owned by different addresses, then the input addresses

belong to the same user. Note that, currently, Bitcoin clients do not provide support for different users to participate in a single transaction; to achieve this, users would have to modify the Bitcoin client implementation themselves.

Heuristic II—"Shadow" Addresses: As mentioned earlier, Bitcoin generates a new address, the "shadow" address [1], on which each sender can collect back the "change".

This mechanism suggests a distinguisher for shadow addresses. Namely, in the case when a Bitcoin transaction has two output addresses, a_{R_n}, a_{R_o}, such that a_{R_n} is a new address (i.e., an address that has never appeared in pubLog before), and a_{R_o} corresponds to an old address (an address that has appeared previously in pubLog), we can safely assume that a_{R_n} constitutes a shadow address for a_i. Note that, in the current Bitcoin implementation, users rarely issue transactions to two different users.

Evaluating Heuristics I and II: In what follows, we evaluate the implications of these heuristics on the user-privacy in Bitcoin. Given the absence of recent statistical data on the number of Bitcoin users, we rely on an estimate of the number of Bitcoin users performed in September 2011; at that time, Bitcoin users amounted to 60,000 users [1]. We then created a C++ parser that parses the first 140,000 blocks (till August 2011) in the Bitcoin block explorer [2].

Our C++ parser extracts all the addresses in each block and categorizes them in clusters of Generic Addresses, GAs, given the two aforementioned heuristics. The parser then outputs a list of addresses organized in different GAs. Our results show that there were 1,632,648 unique addresses in the first 140,000 blocks. Given Heuristic I, we could classify these addresses into 1,069,699 distinct GAs. Given Heuristic II, this number decreases to 693,051 GAs; this corresponds to grouping approximately 58% of Bitcoin addresses with an average of 11.55 addresses per GA. This clearly shows that $\mathcal{A}$'s advantage in the AddUnl game is considerable given Heuristics I and II.

4.2 Behavior-Based Analysis

Besides exploiting current Bitcoin implementations, $\mathcal{A}$ could also make use of behavior-based clustering techniques, such as K-Means (KMC), and the Hierarchical Agglomerative Clustering (HAC) algorithms. Let U be the set of users populating Bitcoin and $(\mathsf{GA}_1, \ldots, \mathsf{GA}_{\mathrm{n_{GA}}})$ denote the GAs that $\mathcal{A}$ has obtained by applying the two aforementioned heuristics on pubLog, respectively. Given this, the goal of $\mathcal{A}$ is to output a group of clusters of addresses $\mathrm{E_{prof}} = \{\mathsf{g}_1, \ldots, \mathsf{g}_{n_\mathsf{U}}\}$ such that $\mathrm{E_{prof}}$ best approximates U. Since each GA is owned by exactly one user, the estimate on the assignment of each GA_i can be modeled by a variable z_i such that $z_i = k$, if and only if, GA_i belongs to g_k.

In fact, HAC assumes that initially each GA represents a separate cluster ($\{z_i = i\}_{i=1}^{\mathrm{n_{GA}}}$) and computes similarity values for each pair of clusters. Clusters with higher similarity value are combined into a single cluster and cluster-to-cluster similarity values are re-computed. The process continues until the number of created clusters equals the number of users $\mathrm{n_U}$. KMC is then initialized using the output of HAC and assumes that each user is represented by the center of each cluster. The algorithm iterates assignments of GAs to clusters and aims at minimizing the overall distance of GAs to

the center of the cluster they have been assigned to. The centers of the clusters and the GA-to-cluster distances are re-computed in each round.

In our implementation (cf. Section 4.3), we represent each transaction that appears within a GA using: *(i)* the time at which the transaction took place, *(ii)* the indexes of the different GAs that appear within the transaction (as senders or recipients), and *(iii)* the values of the BTCs spent by the transaction. Let τ_x denote the set of transactions of GA_x. The degree of similarity between GA_i and GA_j, denoted by $\mathsf{Sim}^{\mathsf{hac}}(\mathsf{GA}_i, \mathsf{GA}_j)$, is then represented by the cosine similarity of lists τ_i and τ_j, i.e., $\mathsf{Sim}^{\mathsf{hac}}(\mathsf{GA}_i, \mathsf{GA}_j) = \frac{\sum_{\forall \tau \in \tau_i \cap \tau_j} (f_{(\tau,i)} \cdot f_{(\tau,j)})}{\|\tau_i\| \cdot \|\tau_j\|}$, where $f_{(\tau,i)}$, $f_{(\tau,j)}$ are the occurrences of item τ in lists τ_i and τ_j respectively, and $\|X\|$ denotes the norm II of vector X. Given this, the resulting distance metric in KMC is $\mathsf{Dist}^{\mathsf{kmc}}(\mathsf{GA}_i, \mathsf{g}_k) = \frac{2}{1+\mathsf{Sim}^{\mathsf{hac}}(\mathsf{GA}_i, \mathsf{g}_k)} - 1$.

Our implementation also accounts for constraints that are posed by real-world settings. Namely, since users cannot be physically located in two different places at the same time, they cannot participate in two different (physical) exchanges of goods at the same time. To account for this case (cf. Section 4.4), we apply different weighting for similarity of GAs who participate in transactions concurrently.

4.3 Simulating the Use of Bitcoin in a University

To evaluate the success of $\mathcal{A}$ in the AddUnl and the $\mathrm{ProfInd}$ games, we simulate a realistic case of using Bitcoin in the Department of Computer Science in ETH Zurich. Here, we assume that the shops located around the university also accept BTCs as a currency. Given the lack of details and statistics about the current use of Bitcoin, this was one of the few "workable" uses of Bitcoin that we could try to accurately model and through which we could evaluate the advantage of $\mathcal{A}$ in the system.

Experimental Setup: To evaluate the privacy implications of using Bitcoin in a university environment, we devised the setup shown in Figure 1. Our Bitcoin simulator takes an XML configurations file as input and outputs: *(i)* a log that details the events that were simulated, the "ground truth", as well as *(ii)* the resulting simulated public Bitcoin log, pubLog. The XML configurations file contains all the necessary parameters to run the simulator. These include the number of users, the number of miners, the simulation time, the difficulty in block generation, as well as usage configurations for creating user profiles and Bitcoin sellers/buyers. Further details about our simulator can be found in Appendix D. As shown in Figure 1, the outputs of our simulator are used to evaluate $\mathcal{A}$'s success. In fact, once the simulations terminate, a Perl-based parser uses the simulated Bitcoin block as input and pre-classifies the simulated addresses into GAs according to Heuristics I and II. The resulting "pre-filtered output" is then fed into our clustering algorithms, the HAC and the KMC algorithms (both implemented in C). The output of these algorithms is then compared using another Perl-based script with the "ground truth" generated by the simulator in order to compute the success of $\mathcal{A}$ in the AddUnl and the $\mathrm{ProfInd}$ games.

We tuned our simulator to match a real-world scenario that reflects the actual behavior of the staff and student members of a Computer Science Department of a university in the Fall 2012 semester. In our setting, we consider a variable number of users, 5.2%

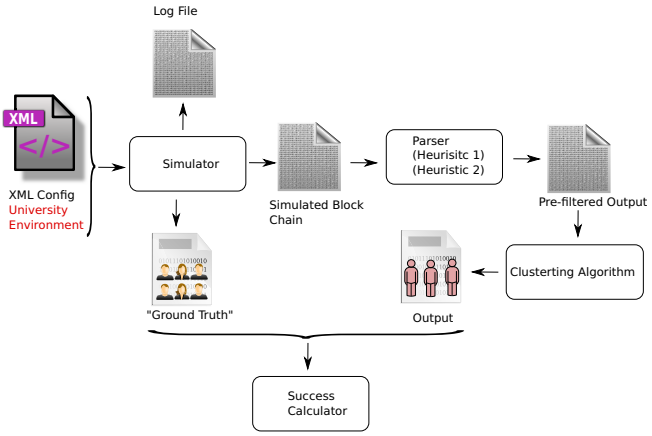


Fig. 1. Experimental setup used throughout our simulations. The outputs of our Bitcoin simulator are pre-filtered according to Heuristics I and II and then fed as input to our clustering algorithm. The clustering result is then compared with the "ground truth" that is emulated by our simulator.

of which are "Professors", 42.0% are "Staff" and the remaining 52.8% are "Students". We consider a total of 6 events, each having several options: lunching/dining (12 options), buying groceries (2 options), buying from vending machines (4 options), online shopping (5 options), purchasing books (2 options), and performing barters with other users, totalling 25 different Bitcoin vendors present in our system. For each user, we assign a probability that the user undergoes each of the possible options of each event. These probabilities are assigned according to the "category" of the user; that is, if the user is a "Professor", then it is more likely that he/she would eat lunches at more expensive restaurants, when compared to the case where the user falls in the "Student" category. For each event, we specify in the XML configuration the following parameters: the frequency of the event, and the price range per option of the event. Each option is assigned a rating that would reflect its popularity. The probability of performing an option is interpolated from the frequency of occurrence of the event per week, and from the rating of the option. To ensure a large variety of profiles in our user base, we specify in the XML configuration a minimum and maximum value for the frequency, rating and price fields. These bounds depend on the category of the user, the event and option in question (c.f. Appendix C). At the start of our experiments, users originally have few (< 10) addresses; as they issue new transactions, new (shadow) addresses are created. We also model the behavior of "privacy-aware" users; we assume that these users create new addresses and send some of their BTCs from their old to their new addresses.

4.4 Experimental Results

Throughout our experiments, we emulated two different scenarios for each simulation round. In the first scenario denoted by "Partial Knowledge", we assume that $\mathcal{A}$ is aware of the location/service of all Bitcoin vendors and as such can distinguish whether a

Table 1. Behavior-based clustering results in the "Partial Knowledge" and "No Knowledge" scenarios. A column entitled X (Y%) denotes an experiment featuring X users among which $Y\%$ are privacy-aware. Each data point in our plots is averaged over five rounds of experiments; we also present the corresponding 95% confidence intervals (shown after the "$\pm$" sign).

		100 (50%)	**200 (0%)**	**200 (50%)**	**200 (100%)**	**400 (50%)**
$\mathbf{Link}_{\mathcal{A}}$		0.91 ± 0.01	0.90 ± 0.01	0.91 ± 0.01	0.92 ± 0.01	0.93 ± 0.01
$\mathbf{Prof}^{a}_{\mathcal{A}}$	NMI	0.76 ± 0.01	0.87 ± 0.01	0.79 ± 0.01	0.70 ± 0.01	0.80 ± 0.01
	AMI	0.75 ± 0.01	0.86 ± 0.01	0.77 ± 0.01	0.68 ± 0.01	0.77 ± 0.01
$\mathbf{Prof}^{\tau}_{\mathcal{A}}$	NMI	0.68 ± 0.01	0.73 ± 0.02	0.70 ± 0.01	0.65 ± 0.01	0.72 ± 0.01
	AMI	0.67 ± 0.01	0.72 ± 0.01	0.69 ± 0.01	0.63 ± 0.01	0.70 ± 0.01

(a) Results in the "Partial Knowledge" scenario.

		100 (50%)	**200 (0%)**	**200 (50%)**	**200 (100%)**	**400 (50%)**
$\mathbf{Link}_{\mathcal{A}}$		0.90 ± 0.01	0.90 ± 0.01	0.91 ± 0.01	0.92 ± 0.01	0.93 ± 0.01
$\mathbf{Prof}^{a}_{\mathcal{A}}$	NMI	0.79 ± 0.01	0.89 ± 0.01	0.79 ± 0.01	0.71 ± 0.02	0.80 ± 0.01
	AMI	0.78 ± 0.02	0.88 ± 0.01	0.78 ± 0.02	0.69 ± 0.02	0.78 ± 0.01
$\mathbf{Prof}^{\tau}_{\mathcal{A}}$	NMI	0.69 ± 0.01	0.73 ± 0.03	0.69 ± 0.03	0.65 ± 0.01	0.72 ± 0.01
	AMI	0.68 ± 0.01	0.72 ± 0.01	0.68 ± 0.03	0.63 ± 0.01	0.70 ± 0.01

(b) Results in the "No Knowledge" scenario.

transaction was performed in exchange of a physical good. In this case, we include the vendors's addresses in the prior knowledge of $\mathcal{A}$ when computing $\mathrm{Adv}_{\mathrm{unl}}$; we also assume that $\mathcal{A}$ can tune the clustering algorithm to take into account that the same user performing this transaction cannot appear in other transactions that takes place at the same time. This case emulates the realistic setting where $\mathcal{A}$ can extract a subset of the addresses owned by geographically co-located Bitcoin users/vendors from the overall public Bitcoin log; for example, $\mathcal{A}$ can extract from the Bitcoin log all the addresses that interact with a known address of a vendor located within the university environment. In the second scenario denoted by "No Knowledge", we consider the case where $\mathcal{A}$ does not know the location or service of the vendors, and as such does not have any prior knowledge, but assumes that up to 10% of the transactions are performed in exchange of goods delivered over the Internet.

Given this setup, we evaluate the metrics $\mathrm{Link}_{\mathcal{A}}$, $\mathrm{Prof}^{a}_{\mathcal{A}}$ (for address-based profiles), and $\mathrm{Prof}^{\tau}_{\mathcal{A}}$ (for transaction-based profiles) with respect to *(i)* the fraction of "privacy-aware" users and *(ii)* the number of users $\mathrm{n_U}$. By privacy-aware users, we refer to users that manually generate new Bitcoin addresses (following a configuration in the XML file) to enhance their privacy in the system. Table 1 depicts our findings. Our results show that both the "Partial Knowledge" and the "No Knowledge" configurations exhibited comparable results.

In the first round of experiments, we evaluate the success of $\mathcal{A}$ with respect to the fraction of "privacy-aware" users. More specifically, we run our clustering and privacy evaluation algorithm in a setting featuring 200 users, among which $0\%, 50\%$, and 100% of the users are privacy-aware. Table 1 shows $\mathrm{Link}_{\mathcal{A}}$, $\mathrm{Prof}^{a}_{\mathcal{A}}$, and $\mathrm{Prof}^{\tau}_{\mathcal{A}}$ with respect to the fractions of privacy-aware users. Here, we use a normalized version on $\mathrm{Link}_{\mathcal{A}}$.

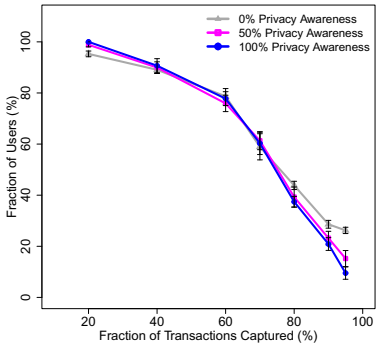


Fig. 2. Fraction of transactions captured by our clustering algorithms in the "No Knowledge" case

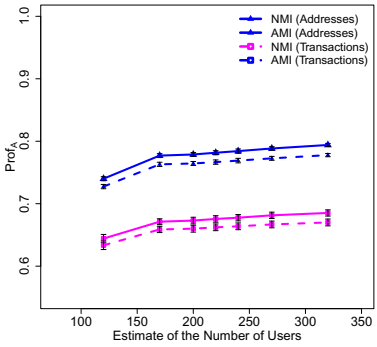


Fig. 3. The case where $\mathcal{A}$ cannot accurately estimate $\mathrm{n_U}$. We assume the 'Partial Knowledge" case where $\mathrm{n_U} = 200$.

In both configurations, the advantage of $\mathcal{A}$ in AddUnl game is only negligible affected by the fraction of the privacy aware users in the system. More specifically, we can see that our clustering algorithms outperform $\mathcal{A}^{\mathcal{R}}$ by almost 90%. On the contrary, $\mathrm{Prof}^{a}_{\mathcal{A}}$ and $\mathrm{Prof}^{\tau}_{\mathcal{A}}$ show a better dependency on the fraction of privacy aware users. When none of users in the system are privacy-aware, the performance of our clustering algorithms is high. In particular, in both configurations $\mathrm{Prof}_{\mathcal{A}^a}$ (NMI and AMI for addresses) range within $0.87 - 0.89$, while $\mathrm{Prof}^{\tau}_{\mathcal{A}}$ (NMI and AMI for transactions) are 0.73. However, as the fraction of the privacy aware users increases, the performance of $\mathcal{A}$ drops and results in $\mathrm{Prof}^{\tau}\ \mathcal{A}$ and $\mathrm{Prof}^{a}_{\mathcal{A}}$ of 0.70 and 0.63 respectively. This mischief can be explained by the fact that privacy aware users add noise to the Bitcoin log. However, the fact that AMI values remain consistently far from 0 and close to 1 indicates that $\mathcal{A}$ performs much better than $\mathcal{A}^{\mathcal{R}}$ and that the estimate chosen by $\mathcal{A}$ is close to the genuine assignment of users to clusters.

Figure 2 depicts the overall fraction of user profiles (measured by means of the similarity of transactions appearing in a user's wallet and the corresponding cluster as discussed in Section 3.2) that are captured by $\mathcal{A}$. Our results show that in the case when $\mathrm{n_U} = 200$ users with 0% privacy awareness, almost 42% of the users have their preferences captured with 80% accuracy. In the case featuring 100% privacy-aware users, this fraction drops to 35% of the users whose profile was correctly clustered with an accuracy of at least 80%. We therefore conclude that the privacy of users in Bitcoin can be compromised, even if users manually create new addresses in order to enhance their privacy in the system.

Furthermore, our results show that $\mathcal{A}$'s advantage over $\mathcal{A}^{\mathcal{R}}$ is not significantly affected by the number of participant users in the case of address unlinkability. $\mathrm{Prof}^{a}_{\mathcal{A}}$ and $\mathrm{Prof}^{\tau}_{\mathcal{A}}$ increase from 0.76 and 0.68 to 0.80 and 0.72 respectively as the number of users increases from 100 to 400. This is mostly because when the number of users increases, the assignments of addresses (or transactions) into groups of users ($\mathcal{A}^{\mathcal{R}}$) performs worst. In Figure 3, we evaluate the case where $\mathcal{A}$ does not have an accurate estimate of the number of users in the university setting. Our findings show that even

if $\mathcal{A}$'s estimate of the number of users is not accurate, the privacy of a considerable fraction of users is still compromised.

5 Discussion

So far, our analysis focused on the current implementation of Bitcoin when used in a university setting. Note that our results provide rather an upper bound to privacy in the studied university setting than an accurate assessment of privacy in Bitcoin in generic settings. In what follows, we analyze the implications of our findings in generic implementations/uses of Bitcoin.

Evading Heuristic I: We start by showing that Heuristic I cannot be easily evaded in any future implementation of Bitcoin without compromising the basic operation of Bitcoin. Indeed, the combination of multiple inputs ensures that coins with "large" values can be recreated from existing smaller BTCs; this prevents the value of coins from being continuously deprecated following every issued transaction until the value of these coins reaches the minimum amount. At that point in time, the only way for Bitcoin users to issue transactions without combining their previous coins is to perform multiple transactions with single-input, one coin at a time. This clearly does not solve the problem since this process can still be tracked by the adversary (transactions will be linked by time). Another alternative would be for Bitcoin developers to provide support for different users to transparently participate in a single transaction. While this would increase the use-cases where Bitcoin finds applicability (e.g., performing contracts [1]), the collaborative construction of transactions by different users is unlikely to be predominately used in the network.

Evading Heuristic II: Similarly, it is easy to see that evading Heuristic II can only decrease the privacy of Bitcoin users. That is, if "shadow" addresses were not utilized, and the change of coins is simply put back in the sender's address then users' activities can be traced in an easier way. We point out that Heuristic II results in the dispersion of the coins among several addresses of the users. This makes the privacy leakage due to Heuristic I even more considerable. One possible way to "evade" Heuristic II would be for Bitcoin users to *(i)* first divide the coins according to the required payment in one transaction and *(ii)* then make the payment with zero change at a random point later in time. Another possible solution to "harden" the reliance on Heuristic II would be for Bitcoin to support Mutli-Input Multi-Output (MIMO) transactions.

Implications to Generic Uses of Bitcoin: We argue that our findings are not specific to the studied university setting and apply to other generic-uses of Bitcoin. More specifically, we believe that the adversary can, to a large extent, extract from the public Bitcoin log a small set of addresses that correspond to geographically co-located users; the adversary can subsequently run our clustering algorithms on the extracted set of addresses. For instance, if Bitcoin were to be used across shops, then the adversary can extract all the addresses that interacted with a specific set of Bitcoin vendors that are located within a specific geographic region. The larger is the number of addresses of (physical)

vendors that the adversary knows, the more complete is the view of the adversary of the geographic sub-network.

The reliance on third-party trusted entities (e.g., Bitcoin banks, Bitcoin Anonymizers, FlexCoin [3]) emerges as one of the few workable solutions to increase the privacy of Bitcoin clients. These entities can hide the direct relationship between the inputs and outputs of a transaction within a sufficiently large anonymity set. However, this solution comes at odds with the main intuition behind the complete decentralization in Bitcoin.

6 Related Work

In [5], Elias investigates the legal aspects of privacy in Bitcoin. In [7], Babaioff *et al.* address the lack of incentives for Bitcoin users to include recently announced transactions in a block, while in [4], Syed *et al.* propose a user-friendly technique for managing Bitcoin wallets. In [14], Karame *et al.* thoroughly investigate double-spending attacks in Bitcoin and show that double-spending fast payments in Bitcoin can be performed in spite of the measures recommended by Bitcoin developers. Clark *et al.* [11] propose the use of the Bitcoin PoW to construct verifiable commitment schemes. Reid and Harrigan [16] analyze the flow Bitcoin transactions in a small part of Bitcoin log, and show that external information, i.e., publicly announced addresses, can be used to link identities and organizations to some transactions.

ECash [8–10] and anonymous credit cards were the first attempts to define privacy-preserving transactions. Privacy in ECash consists of user anonymity and transaction unlinkability; by relying on a set of cryptographic primitives ECash ensures that payments pertaining to the same user cannot be linked to each other or to the payer, provided that the latter does not misbehave. In [15], Pfitzmann *et al.* define unlinkability and privacy in pseudonymous systems. Dwork [13] defined differential privacy and quantified the information leakage from the query access of individuals. In Section 3.2, we adapt Dwork's generic differential privacy notion to our Bitcoin privacy notions. Finally, in [18], Shokri *et al.* quantify location privacy by assessing the error of the adversarial estimate from the ground truth. In [12, 17] the authors further introduce entropy-based metrics to assess the communication privacy in anonymous networks.

7 Conclusion

In this paper, we evaluated the privacy provisions in Bitcoin when it is used as a primary currency to support the daily transactions of individuals in a university setting.

Our findings show that the current measures adopted by Bitcoin are not enough to protect the privacy of users if Bitcoin were to be used as a digital currency in a university setting. More specifically, we rely on a simulator that mimics the use of Bitcoin in a realistic university setting. Our results show that if Bitcoin is used as a digital currency to support the daily transactions of users in a typical university environment, then behavior-based clustering techniques can unveil, to a large extent, the profiles of 40% of Bitcoin users, even if these users try to enhance their privacy by manually creating new addresses. Finally, we discussed a number of solutions that could be integrated by Bitcoin developers to enhance the privacy of users.

Acknowledgements. The authors would like to acknowledge the reviewers for their valuable comments and feedback.

References

1. Bitcoin – Wikipedia, https://en.bitcoin.it/wiki
2. Bitcoin Block Explorer, http://blockexplorer.com/
3. Flexcoin –The Bitcoin Bank, http://www.flexcoin.com/
4. Bitcoin Gateway, A Peer-to-peer Bitcoin Vault and Payment Network (2011), http://arimaa.com/bitcoin/
5. Bitcoin: Tempering the Digital Ring of Gyges or Implausible Pecuniary Privacy (2011), http://ssrn.com/abstract=1937769 or http://dx.doi.org/10.2139/ssrn.1937769
6. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System
7. Babaioff, M., Dobzinski, S., Oren, S., Zohar, A.: On Bitcoin and Red Balloons. CoRR (2011)
8. Brands, S.: Electronic Cash on the Internet. In: Proceedings of the Symposium on the Network and Distributed System Security (1995)
9. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
10. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990), http://dl.acm.org/citation.cfm?id=88314.88969
11. Clark, J., Essex, A.: CommitCoin: Carbon Dating Commitments with Bitcoin (Short Paper). In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 390–398. Springer, Heidelberg (2012)
12. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)
13. Dwork, C.: Differential privacy: a survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
14. Karame, G., Androulaki, E., Capkun, S.: Double-Spending Fast Payments in Bitcoin. In: Proceedings of ACM CCS (2012)
15. Pfitzmann, A., Hansen, M.: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology. Fachterminologie Datenschutz und Datensicherheit, 111–144 (2008)
16. Reid, F., Harrigan, M.: An Analysis of Anonymity in the Bitcoin System. CoRR (2011), http://www.bibsonomy.org/bibtex/257d6640d03ae4a5668ef8b32656461eb/dblp
17. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In: Dingledine, R., Syverson, P. (eds.) PET 2002. LNCS, vol. 2482, pp. 41–53. Springer, Heidelberg (2003)
18. Shokri, R., Theodorakopoulos, G., Boudec, J.L., Hubaux, J.P.: Quantifying location privacy. In: Proceedings of the IEEE Symposium on Security and Privacy (2011)
19. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: 26th Annual International Conference on Machine Learning, ICML (2009)
20. Vinh, N.X., Epps, J., Bailey, J.: Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. Journal of Machine Learning Research (2010)

A Multi-input Transactions

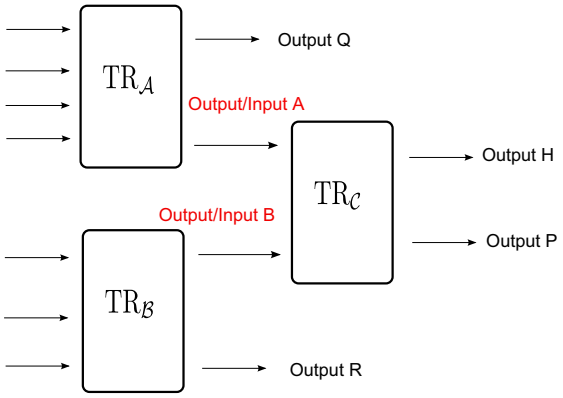


Fig. 4. Example of referencing input/output in transactions

B Bitcoin Block Explorer

Table 2. Example Block of Bitcoin. The block contains 2 transactions, one of which awards the miner with 50 BTCs.

Hash: 000000000043a8c0fd1d6f726790caa2a406010d19efd2780db27bdbbd93baf6
Previous block: 00000000001937917bd2caba204bb1aa530ec1de9d0f6736e5d85d96da9c8bba
Next block: 00000000000036312a44ab7711afa46f475913fbd9727cf508ed4af3bc933d16
Time: 2010-09-16 05:03:47
Difficulty: 712.884864
Transactions: 2
textbfMerkle root: 8fb300e3fdb6f30a4c67233b997f99fdd518b968b9a3fd65857bfe78b2600719
Nonce: 1462756097

Input/Previous Output	Source & Amount	Recipient & Amount
N/A	Generation: 50 + 0 total fees	Generation: 50 + 0 total fees
f5d8ee39a430...:0	1JBSCVF6VM6QjFZyTnbpLjoCJ...: 50	16ro3Jptwo4asSevZnsRX6vf..: 50

C Example Configuration File

Listing 1. Example of XML configuration parameters for a "lunch" event with 12 options corresponding to the profile of a "Professor"

```
<!– lunch, eventid="0" refers to event with id="0" from above –>
<ProfileEvent eventid="0" minFreqPerWeek="5" maxFreqPerWeek="5" >
<Store storeid="0" maxPref="1" minPref="0" maxPrice="10.0" minPrice="8.0" />
<Store storeid="1" maxPref="1" minPref="0" maxPrice="13.0" minPrice="10.0" />
<Store storeid="2" maxPref="1" minPref="0" maxPrice="15.0" minPrice="13.0" />
<Store storeid="3" maxPref="4" minPref="2" maxPrice="25.0" minPrice="15" />
<Store storeid="4" maxPref="2" minPref="0" maxPrice="20.0" minPrice="15.0" />
<Store storeid="5" maxPref="2" minPref="0" maxPrice="17.0" minPrice="12.0" />
<Store storeid="6" maxPref="0.5" minPref="0" maxPrice="20.0" minPrice="8.0" />
<Store storeid="7" maxPref="0.5" minPref="0" maxPrice="10.0" minPrice="7.5" />
<Store storeid="8" maxPref="4" minPref="2" maxPrice="25.0" minPrice="15" />
<Store storeid="9" maxPref="0.5" minPref="0" maxPrice="25.0" minPrice="10" />
<Store storeid="10" maxPref="0.5" minPref="0" maxPrice="10.0" minPrice="5.0" />
<Store storeid="11" maxPref="3" minPref="1" maxPrice="12.0" minPrice="9.0" />
</ProfileEvent>
```

D Bitcoin Simulator

Our simulator is round-based; in each simulation round (defined as a "weekly timestepping" interval), events are added to a priority queue with a probability dictated by the configuration file. These events correspond to one of the following operations:

- *Issue a new transaction:* Users might issue new Bitcoin transactions whose time, value, beneficiary and purpose stem from the XML configurations file. The process of transaction issuance in our simulator fully mimics its counterpart in the genuine Bitcoin system.
- *Generate a new Bitcoin address:* Here, in addition to the automatically generated addresses in Bitcoin (cf. Section 3), "privacy-aware" users might decide to generate a number of new addresses to further "obfuscate" their usage of Bitcoin.

Conforming with the current use of Bitcoin, only few users in our setting were miners (i.e., mining is currently mostly performed by dedicated mining pools).

Our Bitcoin simulator abstracts away network delays, congestion, jitter, etc.. We also assume that all transactions in the system are well-formed and we do not model transaction fees that are incurred in the network. While malformed transactions and double-spending attempts [14] can be indeed witnessed in the genuine Bitcoin system, we believe that malicious behavior in Bitcoin is orthogonal to our privacy investigation—which explains the reason why we did not model such a misbehavior in our simulator. Moreover, our simulator relies on a variant greedy algorithm that closely approximates the genuine algorithm used in Bitcoin. Note that while the distribution of generated blocks in our simulator matches that in Bitcoin [14][6], we increased the average time

[6] It was shown in [14] that the block generation in Bitcoin follows a shifted geometric distribution with parameter 0.19.

between the generation of successive blocks in the simulator to better cope with simulated network dynamics[7].

E Captured Transactions w.r.t. $\mathrm{n_U}$

Figure 5 shows the impact of the number of users $\mathrm{n_U}$ on the performance of our profiling algorithms.

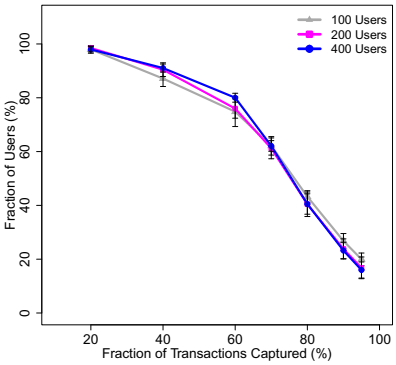


Fig. 5. Fraction of captured transactions with respect to $\mathrm{n_U}$. Here, we consider the "No Knowledge" case where 50% of the users are privacy-aware.

Our results show that the fraction of users whose transactions are correctly captured by our algorithms is not considerably affected by the number of users in the system. For instance, the fraction of users whose profiles were captured with an accuracy of at least 80% is approximately 40% when $\mathrm{n_U} = 100, 200, 400$.

Note that this conforms with our previous observations in Section 4.4; as shown in Figure 3, $\mathrm{Prof}_{\mathcal{A}}$ is only slightly affected as $\mathrm{n_U}$ increases.

[7] Throughout our experiments, we considered that new blocks were generated every 20 minutes on average.

The Importance of Being Earnest [In Security Warnings]

Serge Egelman[1] and Stuart Schechter[2]

[1] University of California, Berkeley
[2] Microsoft Research, Redmond
egelman@cs.berkeley.edu
stuart.schechter@microsoft.com

Abstract. In response to the threat of phishing, web browsers display warnings when users arrive at suspected phishing websites. Previous research has offered guidance to improve these warnings. We performed a laboratory study to investigate how the choice of background color in the warning and the text describing the recommended course of action impact a user's decision to comply with the warning. We did not reveal to participants that the subject of the study was the warning, and then we observed as they responded to a simulated phishing attack. We found that both the text and background color had a significant effect on the amount of time participants spent viewing a warning, however, we observed no significant differences with regard to their decisions to ultimately obey that warning. Despite this null result, our exit survey data suggest that misunderstandings about the threat model led participants to believe that the warnings did not apply to them. Acting out of bounded rationality, participants made conscientious decisions to ignore the warnings. We conclude that when warnings do not correctly align users' risk perceptions, users may unwittingly take avoidable risks.

Keywords: Usability, security warnings, risk communication.

1 Introduction and Background

Many web browsers use full screen warning messages that are displayed to users whenever they visit suspected phishing websites. Egelman et al. studied several of these warnings and proposed a set of recommendations for improving them [1]. These recommendations included designing warnings that get noticed by interrupting the user's primary task, recommending a clear course of action so that the user knows what to do, distinguishing them from less serious warnings to prevent habituation, and minimizing the impact that a well-designed forgery has on a user's trust. In this study, we performed a controlled experiment to examine some of these recommendations.

The first question we examined was whether clearer explanations of users' available options would result in them making better choices. Most browser-based phishing warnings present users with multiple options, usually a recommendation not to visit a suspicious website and another option to bypass the warning. We examined the options offered by Microsoft's Internet Explorer 8 phishing warning [3].

When a user visits a suspected phishing website, she is advised to "go to my homepage instead." Because this text does not conceptually help the user complete her primary task—it was unlikely that she was trying to visit her homepage—we were concerned that this text may contribute to the warning being ignored. Thus, we tested how option text

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 52–59, 2013.

impacts decisions by creating an experimental condition that appeared to be more likely to aid in completing the primary task: "search for the real website." We hypothesized that this text would be more effective because it may facilitate completion of a primary task and it underscores the threat model: the user was visiting a fraudulent website designed to look like a legitimate one and therefore following this link may help the user locate the intended website. In addition to examining the option text, we wanted to examine the recommendation to minimize habituation by designing phishing warnings differently from less-severe warnings. Thus, we also varied the background color by turning it red in some conditions, while keeping it white in another.

We contribute to the literature on security warnings by showing that altering text and color significantly increase user attention. However, we show that attention alone is insufficient for warning compliance; because many participants did not believe the warnings were relevant to them, they chose to ignore them. We conclude that a user may face moral hazard when she encounters a security warning that does not effectively communicate the risk it is trying to mitigate. We later validated this finding in subsequent work [4].

2 Methodology

We performed our experiment on the Microsoft campus, using a recruitment service to obtain a (non-university-biased) sample of 59 participants. We did not tell participants that we were studying security, as that would compromise external validity by priming them. Instead, we told them that we were performing a usability study of Hotmail and therefore only recruited Hotmail users. At the time, Hotmail was the largest webmail provider worldwide [5], and therefore we believe our sample is generalizable to a large proportion of Internet users.

We randomly assigned participants to one of three between-group conditions and then gave them a set of tasks that involved checking email. After completing the final task, participants received a simulated phishing email. We observed their reactions to a warning from one of the three treatments and then asked them to fill out an exit survey.

2.1 Recruitment

We recruited participants during September of 2008. Thirty were male and the mean age was 37.6 ($\sigma = 11.6$). We selected participants who had previously opted in to being contacted about user studies at Microsoft, and screened out participants who either did not use Hotmail for their email or IE as their primary web browser. Because we were only interested in participants who were most vulnerable to phishing attacks, we screened out participants who had technical jobs.

When a participant arrived for his individual session, we asked him to sign a consent form, and then handed him an instruction sheet. The experimenter read the instructions aloud to ensure that everyone received the same information. When ready to begin, the experimenter left the room so as to not influence participants' behaviors. The experimenter observed participants from a separate control room as they completed a series of tasks. Once complete, the experimenter returned to administer an exit survey.

2.2 Tasks

To maximize ecological validity, we wanted participants to behave as they would when seeing phishing warnings outside the laboratory. Since security is rarely a primary task (e.g., users do not sit down at the computer to "not get phished"), we needed to mask the purpose of the study. We told participants that we were examining the usability of Hotmail. As an incentive, we paid them a dollar for each message that they opened during the study, and an additional four dollars if they "interacted with that message." So as to not bias them towards our phishing messages—we did not want them to feel compelled to click links in every message—we told them that filing messages away or deleting them would also count. Thus, we created an incentive only to read every received email, not necessarily to follow its instructions; participants received just as much compensation for deleting a message as following its links.

Because we could not solely rely on them to receive real email during the study period from outside sources, we explained that the experimenter would send them a message every ten minutes, but did not specify how many times this would occur. The first message sent by the experimenter was a personal message written in plaintext that asked the participant to visit a movie website and respond with the movie they most wanted to see. The second message was an HTML-based message that came from a photo-sharing website inviting the participant to view a shared photo album that the experimenter had posted. These two messages served only to further convince participants that they were part of an email usability study, we therefore do not mention them again.

Two minutes after participants viewed the second email message, the experimenter sent a simulated phishing message. This message did not follow the ten minute interval and was intended to create some ambiguity as to whether it was part of the study or not. The domain that we used to send it was registered solely for the study, though in its body it claimed to be from Microsoft and encouraged readers to click a link and enter personal information on the resulting website. The domain used for the destination URL as well as sending the email, *microsoft-study.com*, was added to a phishing blacklist, thereby triggering a phishing warning when accessed. The message offered participants the opportunity to enter a prize drawing if they visited the included URL. Upon arriving at this URL, participants saw one of three warnings that we describe in the next section. If they chose to ignore the warning and proceed to the website, they were presented with a login form that appeared identical to the Hotmail login screen (i.e., the goal of the simulated phishing scam was to capture Hotmail credentials).

In real life, a phishing warning appears after a user has clicked a link in a scam email. For ecological validity, we needed participants to be in this same frame of mind, which is why we incentivized them to read messages received during the study. Specifically, if a participant did not read the message, she would never attempt to visit the suspicious website, she would never see one of the three warning messages, and we would not yield any data regarding whether or not she would have obeyed the warning. We were not measuring how many messages participants read or how many websites they visited. The behavior we were studying was whether, after reading a scam email and visiting its included URL, participants would dismiss the phishing warning and submit login credentials. Thus, our dependent variable was whether participants entered information into the fake Hotmail login website.

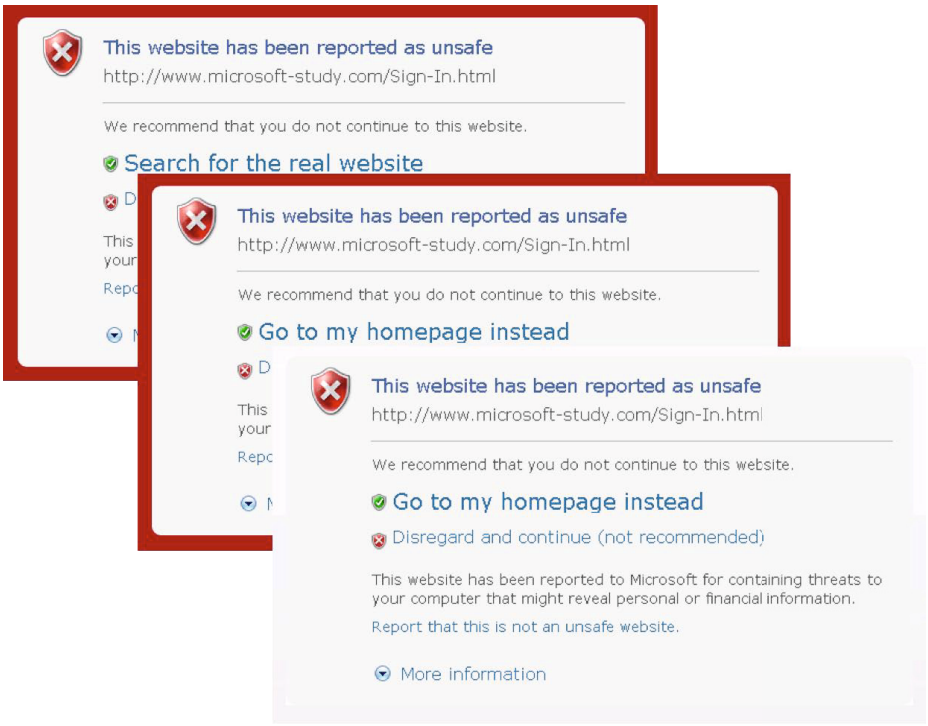


Fig. 1. Participants who clicked the link contained in the simulated phishing email were exposed to one of three possible phishing warnings. The bottom represents the *search* condition, while the middle represents the *home* condition, and the top represents the *white* condition.

2.3 Conditions

We created our initial two conditions to examine the role of the option text: one warning recommended that users "go to my homepage instead," while the warning in the other condition recommended that they "search for the real website." We refer to these as the *home* and *search* conditions, respectively. Our hypothesis was that study participants would be less likely to heed the recommendations of the phishing warnings if those recommendations appeared unlikely to help complete a primary task (i.e., they were not attempting to visit a homepage at the time that the warning appeared). We believed that the text "search for the real website" would not conflict with the primary task as well as underscore the threat model.

Previously, Egelman et al. concluded that users were habituated to ignoring the IE7 phishing warnings because these warnings were designed similarly to other IE7 security warnings, such as those used to indicate SSL errors [1]. We created a third condition to examine habituation effects by removing the red border, and replacing it with a white border, so that it would look similar to the ubiquitous IE7 warnings. We refer to this condition as the *white* condition. Our three experimental conditions are described in Table 1 and depicted in Figure 1. We intentionally did not create a fourth condition,

to separate the effects of the red border from the effects of the new text (e.g., a warning with a white border using the "search for the real website" text), over concerns about statistical power. Specifically, we designed this experiment as a first inquiry into whether an effect exists, rather than an attempt to quantify the size of that effect.

Table 1. Descriptions of the three conditions as well as summary statistics for the total viewing time, average number of views per user, and the average time per view. Participants in the *search* condition viewed the warnings significantly more frequently as well as for significantly longer periods of time in total.

Condition	Option Text	Background	Total Time	Average Views	Average Time
White	*Go to my homepage instead*	White	12.00s	1.36	9.76s
Home	*Go to my homepage instead*	Red	17.81s	1.67	10.76s
Search	*Search for the real website*	Red	30.97s	2.67	11.84s

3 Results

We observed 48 of 59 participants (81%) follow the link to the suspected phishing website. Due to technical difficulties, three of these participants saw no phishing warnings and therefore proceeded to enter their personal information (i.e., in the absence of a warning, participants believed this was a legitimate website). Throughout the rest of this paper, we focus on the 45 remaining participants who saw one of the three phishing warnings. Of these 45 participants who viewed the warnings, twelve entered personal information (27%), whereas everyone else navigated away from the website.

A chi-square test did not show that participants in any one condition were significantly more likely to divulge their credentials: five in the *white* condition (33% of 15), three in the *home* condition (20% of 15), and four in the *search* condition (26.7% of 15). We believe that this null result has more to do with low statistical power stemming from our limited sample size. However, we found a significant interaction effect based on both the red border and the text of the warnings with regard to the amount of time participants spent reading the warnings. Table 1 lists the total time participants in each condition spent viewing the phishing warnings, the number of times they revisited the phishing warnings, and the average time spent viewing the warnings.[1]

We performed a Kruskall-Wallis one-way analysis of variance and found that participants in the *search* condition viewed the phishing warnings for significantly longer time in total ($\chi^2_2 = 7.83$, $p < 0.020$). Upon performing post-hoc analysis using a Mann-Whitney U test, we found that this was due to significant differences between the 31s average viewing time in the *search* condition and the 12s average viewing time in the *white* condition ($p < 0.010$; Cohen's $d = 0.98$). Likewise, when examining the total number of times that participants viewed the warnings, we found that those in the *search* condition went back to review the warning significantly more often (i.e., they closed the warning, reread the email message, clicked the link again, etc.; $\chi^2_2 = 7.02$,

[1] We removed data from one participant in the *white* group after he—against directions—asked for help and then waited for the experimenter to respond from the observation room, therefore artificially increasing the amount of time he spent viewing the warning.

$p < 0.030$). This was also attributed to the contrast with the *white* condition ($p < 0.012$; Cohen's $d = 0.99$). This indicates an interaction effect between the red background and the new text; participants spent significantly longer analyzing the warnings only when both these features were present.

Using our exit survey, we found a significant correlation between participants ignoring warnings during the experiment and claiming to have seen them previously ($\phi = 0.497$, $p < 0.001$); nine of the twelve "victims" said they recognized the warnings (75%), whereas only seven of the thirty-three non-victims (21%) claimed to have recognized the warnings. Thus, the combination of the new text and red background decreased habituation, which may explain why participants in the *search* condition spent significantly longer viewing the warnings.

4 Discussion

Our warning manipulations increased the amount of time participants spent reading the warnings. It is not clear whether the originality of the designs simply decreased habituation, or whether the new option text caused them to think more about their choices.[2] Still, a third of our participants ultimately succumbed to the attack. We found no correlation between falling for the attack and the amount of time spent viewing the warnings. Thus, while participants in the *search* condition paid more attention to the warnings, they were just as likely to dismiss them. In this section we discuss some possible reasons for why the warnings failed and how warning effectiveness may be improved.

4.1 Bounded Rationality

Of the twelve participants who divulged credentials, all but one understood that the warnings wanted them to navigate away (i.e., "do not visit the website"). The one participant, who was in the *white* condition, responded *"check the sender or link to make sure it would not be harmful."* Thus, participants did not disregard the warnings because they did not understand what the warnings wanted them to do. Instead, we believe that participants chose to disregard the warnings because they did not believe they were at risk; none of the warnings (Figure 1) mentioned a specific threat unless the user clicked the "more information" link. The warnings only said that the website "has been reported as unsafe" and that it was reported for "containing threats to your computer." These terms are vague and do not describe one specific threat model. Thus, it is not surprising that participants who ignored the warnings did not understand the threat: ten of the participants who ignored the warnings (83% of 12) said that they did so because they were visiting a legitimate website.

Users are exposed to many varying ill-defined online threats. In research on users' mental models of computer security, Wash observed that this has resulted in widely varying conceptualizations when given vague terms like "security" and "hacker" [6]. Because the warnings used terminology like "unsafe" and "threats to your computer,"

[2] Six (40% of 15) participants in the *search* condition attempted to use the search functionality of the warning to find the "real" website. Since no real website existed, this proved futile.

without providing details, participants likely had varying mental models. When given explanations of the threat model, participants acted rationally: nine of ten participants who clicked the "more information" link, and read about phishing, complied with the warning. These participants correctly understood that the website was attempting to steal their credentials.

4.2 Moral Hazard

We examined participants' understandings of the threat model by asking them to explain the danger of ignoring the warnings. We coded correct answers as ones that said phishing scams attempt to steal personal information. We found that only 14 understood this (31% of 45). Of the remaining 31 participants, all of them mentioned other threats. Some examples included:

- *"I could potentially get a virus or spyware"*
- *"Getting a virus ruining your computer"*
- *"Will get some spyware"*

Three participants who disregarded the warnings (25% of 12) said that they did not care if our computer was infected with a virus. That is, because they believed that someone else would bear all the risk from an infected computer, they did not believe there were any incentives to obeying the warnings. While this would be a rational justification if the threat were indeed malware (see, e.g., [2]), it illustrates how bounded rationality, caused by a limited understanding of the threat model, resulted in moral hazard.

4.3 Lack of Motivation

In Wogalter's Communication-Human Information Processing Model (C-HIP) [7], people undergo several steps between warning exposure and choosing an action. Motivation is a key step: users are unwilling to comply with warnings that they do not believe apply to them. Thus, the changes we made to the warning resulted in improvements at the attention stages of the model by minimizing habituation effects (this was corroborated by the significant correlation between participants ignoring the warnings and claiming to recognize them; those in the *search* condition were least likely to recognize them). However, the warnings failed because they failed to motivate participants.

We therefore believe that our experimental results indicate that motivation problems may be preventable by designing warnings to explicitly state a threat model. In fact, we later performed a followup experiment to validate this finding [4]: participants were significantly more likely to obey SSL warnings when those warnings explicitly communicated threat models that participants found to be relevant to them.

5 Conclusion

We expected to find that by using techniques to increase attention, participants would be more likely to obey the warnings because they would spend more time reading them.

We found that we were partially correct: participants spent more time reading the warnings, but they ultimately did not behave any differently. Our exit survey data suggests participants who were unmotivated by the threat model—as they understood it—chose to disobey the warnings. We expected to observe a much greater effect size and therefore used a limited sample. In the *white* condition, ten of fifteen participants complied with the warnings. We consider this to be the baseline rate of compliance because this condition was designed to appear similar to previous phishing warnings (i.e., this condition approximated a control). Given our sample size, for there to be a significant difference between one of the other experimental conditions and this baseline rate of compliance (67%), the other conditions would need compliance rates of 100%. Thus, phishing warnings have improved to the point that much larger sample sizes are needed to quantitatively study minor design changes.

While we were unable to reject the null hypothesis, this study yielded important lessons for future security mitigations. We showed that distinguishing severe risks from other less-severe risks may aid in capturing user attention. However, warnings cannot rely on attention alone, they must also communicate risk effectively. Many participants incorrectly believed they were being warned about different irrelevant threats. In future warnings, designers should highlight the risks of ignoring the warnings so that users are more likely to understand that the warnings apply to them. This means warning less often in low risk situations, providing stronger evidence of the presence of risk, or helping users to link the risk to their immediate situations through contextual cues.

References

1. Egelman, S., Cranor, L.F., Hong, J.: You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In: CHI 2008: Proceeding of the 26th SIGCHI Conference on Human Factors in Computing Systems, pp. 1065–1074. ACM, New York (2008)
2. Herley, C.: So long, and no thanks for the externalities: the rational rejection of security advice by users. In: New Security Paradigms Workshop, pp. 133–144 (2009)
3. Lawrence, E.: IE8 Security Part III: SmartScreen Filter (July 2008), `http://blogs.msdn.com/ie/archive/2008/07/02/ie8-security-part-iii-smartscreen-filter.aspx`
4. 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, SSYM 2009, pp. 399–416. USENIX Association, Berkeley (2009)
5. Terdiman, D.: Microsoft aiming to clean up hotmail user's inboxes. CNET News (October 3, 2011), `http://news.cnet.com/8301-13772_3-20114975-52/microsoft-aiming-to-clean-up-hotmail-users-inboxes/`
6. Wash, R.: Folk models of home computer security. In: Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS 2010. ACM, New York (2010)
7. Wogalter, M.S.: Communication-Human Information Processing (C-HIP) Model. In: Wogalter, M.S. (ed.) Handbook of Warnings, pp. 51–61. Lawrence Erlbaum Associates (2006)

Exploring Extrinsic Motivation for Better Security: A Usability Study of Scoring-Enhanced Device Pairing

Alexander Gallego[1], Nitesh Saxena[2], and Jonathan Voris[3]

[1] YieldMo, Inc.
[2] University of Alabama at Birmingham
[3] Columbia University

Abstract. We explore the use of extrinsic motivation to improve the state of user-centered security mechanisms. Specifically, we study applications of scores as user incentives in the context of secure device pairing. We develop a scoring functionality that can be integrated with traditional pairing approaches. We then report on a usability study that we performed to evaluate the effect of scoring on the performance of users in comparison operations. Our results demonstrate that individuals are likely to commit fewer errors and show more acceptance when working with the scoring based pairing approach. Framing pairing as a game and providing feedback to users in the form of a score is an efficient way to improve pairing security, particularly among users such as children who may not be aware of the consequences of their decisions while performing security tasks.

Keywords: Device Pairing, Games, Usability, Entertainment, Mobility, Ubiquitous Computing.

1 Introduction

Devices which utilize short range wireless communication are commonplace and continue to proliferate. This popularity unfortunately increases the prominence of associated security risks. Wireless channels are easy to eavesdrop upon and manipulate. A fundamental security objective is therefore to secure them. In this paper, the term "pairing" refers to the process of bootstrapping secure communication between two wireless devices in a way that is resistant to eavesdropping and man-in-the middle attacks. A promising research direction towards solving the pairing dilemma is to leverage an Out-Of-Band (OOB) channel that is governed by human users. Examples of OOB channels include audio, visual, and tactile channels. Unlike classical radio channels, OOB channels are "human-perceptible," i.e., the underlying transmission and reception that drives them can be perceived by human senses. Due to this property, OOB communication provides authentication and integrity, unlike radio communication.[1]

The usability of an OOB-based pairing method is very important. Since OOB channels typically have low bandwidth, the shorter the data that a pairing method needs to transmit over these channels, the better its usability. A recent innovation to this end is

[1] Previous research has shown that it is more difficult for humans to identify the origin of sound than other forms of output [5]. Unlike traditional wireless channels, however, it is possible for users to do so.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 60–68, 2013.

the development of Short Authenticated String (SAS) based protocols (e.g.,[8]) that reduce the length of data transmitted over OOB channels. A variety of pairing methods based on visual, audio, tactile, and infrared OOB channels have been proposed based on these protocols [4].

Unfortunately, device pairing has not been addressed in a fully desirable manner. Prior work on pairing raises several fundamental usability and security related concerns and research challenges. One of the most prominent of these is that the amount of security that a SAS based pairing method provides is dependent on the size of strings that it uses; a k-bit SAS limits the probability of a successful attack to 2^{-k}. Existing pairing methods use short strings in their SAS protocols. Typically, these values are only 15 bits long. SASs of this size are not large enough to provide sufficient security for certain applications. Unfortunately, increasing the length of a pairing system's SASs causes pairing to take longer to complete. This, in turn, leads to poor usability and can also have an impact on security.

Further, even while using short OOB strings, several comparison-based pairing methods do not offer the theoretical level of security guaranteed by their underlying protocols, as demonstrated in [4]. This is due to the potential for human errors in these protocols. Such errors can be of two forms: *fatal* and *safe* [1]. Fatal errors (also known as false positives or "Type I" errors) occur when a user accepts a pairing instance, although the OOB strings on the two devices did not match, which may lead to a man-in-the-middle attack. Safe errors (or false negatives or "Type II" errors), on the other hand, occur when a user rejects a pairing instance even when the OOB strings on the two devices match. Such errors undermine the usability of pairing, but can also have an indirect impact on security; a failed pairing necessitates repetition, which may lead to user annoyance and translate into attacks eventually.

Our overall solution to these challenges is to make use of a reward system as a way of measuring and improving users' performance during the pairing process. The system draws from motivation research in human psychology. Motivation can be intrinsic or extrinsic [7]. Intrinsic motivation emanates from oneself, i.e., when one is inherently interested in a task. Clearly, human users lack intrinsic motivation for security tasks. Extrinsic motivation, on the other hand, relates to how users can be externally motivated for non-intrinsically interesting tasks [6], and is directly applicable to security tasks, such as pairing. In this paper, we consider a very simple form of extrinsic motivation, visual scores, to improve the security and usability of secure device pairing. Framing pairing as a game and providing feedback to users in the form of a score is an efficient way to improve pairing security, particularly among users such as children who may not be aware of the consequences of their decisions while performing security tasks.

Our contributions in this paper are as follows: (1) We develop a scoring functionality that can be integrated with traditional pairing approaches that involve the manual comparison of numbers displayed on two devices; (2) We report on a between-subjects study to evaluate the effect of scoring on the performance of numeric comparisons.

2 Threat Model for Device Pairing

We summarize the threat model for device pairing based on OOB communication [8]. In this model, wireless devices can establish two types of communication mediums.

The first is a traditional wireless radio channel, which is characterized by a large bandwidth capacity. We imagine a worst case scenario in which an adversary has full control over the wireless channel. The second medium comprises the set of OOB channels, which feature modest bandwidths but are physically authenticatable. That is, OOB channels are crafted from output which can be perceived by unassisted humans, which allows users to verify transmission sources themselves.

3 Design of Pairing Methods

In this section, we present the design and implementation of two pairing approaches. One of these, referred to as Plain Comparison, is a variant of the traditional approach involving numeric comparisons. The other, referred to as Scored Comparison, integrates a scoring and grading functionality with plain numeric comparisons. Traditional pairing approaches involving numeric comparisons employ the comparison of a single 5 digit number displayed on the screen of two devices. For better security, one possibility is to employ longer numbers; however, longer numbers become harder for the users to compare. In our Plain Comparison method, we display four 5 digit numbers on four seperate quadrants of the screen, implying four times the security provided by the traditional method.

In order to add a scoring functionality to the Plain Comparison method, we needed to incorporate some comparison instances that could be used to calculate the score based on the performance of the user. Note that the "pairing instances," that is, the numbers resulting from the OOB strings generated by the pairing protocol, cannot be used for this purpose because the devices themselves are not aware whether these instances are matching or non-matching. To this end, we create some dummy "scoring instances"; whether these are matching or non-matching is pre-determined and known to the devices. A score is calculated based on how accurately users compare the scoring instances. The scoring instances are required to calculate a user's score, since unlike the pairing instances, the device pair shares knowledge regarding whether or not they match. As an alternative to scoring instances, the secure channel can be used to calculate scores based on pairing instances if score calculation is postponed until after devices are successfully paired. While this is more efficient, it further delays user feedback.

3.1 Scored Comparison Method

The scored pairing GUI consists of colored quadrants which are used as a visual mnemonic technique to help users associate the numbers displayed within. We also used another type of mnemonic technique called chunking, where information is broken up into more manageable sizes to allow short-term memory to operate more efficiently. This pairing method consisted of 2 rounds, and therefore 8 comparison instances. 4 instances were shown per round; this was done to keep the design similar to the Plain Comparison method. 4 instances were used for pairing and 4 instances were used for scoring.

Initialization Step. In order for our pairing game to calculate a score, the two devices that are being paired must have a mechanism for determining which displayed values

match and which differ. In order to accomplish this, the computers can first agree upon a seed value. There are several ways in which the two devices can settle on a seed. Two possibilities are the use of a publicly known value that is controlled by a service provider or synchronized system times. We employed the latter approach in our implementation. After a seed has been agreed upon, it is used to populate two binary arrays. These arrays are used to generate the numeric values that will be displayed to users during the pairing process. Note that our prototype implementation also utilizes the same seed in order to generate pairing instances. In a real world implementation, however, pairing instances would be generated from OOB strings that are created as a result of the underlying SAS pairing protocol.

Pairing Step. The pairing step is the iterative game loop of our system. To start, all four color quadrants of the pairing game's display will be empty. The basic idea of this portion of the pairing procedure is to populate these quadrants with random five digit integer values. First, two of the quadrants are filled with pairing instances; these will be used to determine the success or failure of the pairing operation. Next, the remaining half of the display is populated with scoring instances; whether or not a user can successfully detect matches in these values will determine his or her score. In case the pairing failed, a score is still displayed but the user will be asked to play another round until pairing succeeds. Matching values are generated by hashing a portion of the pertinent bit string, while mismatches are generated pseudorandomly.

Score Calculation. The method's state was recorded with each press of the screen (i.e., user's input) and at the end of every round. This was an important part of our score calculation. The score had two major components: performance points based on accuracy of comparison and free give-away points for trying. First, we compared the screen presses array with that of the non-matching instances. If the instance was a scoring instance and it was a correct identification of a non-matching number, we gave the user 5 points; otherwise, we gave the user 1 point. The last free give-away point was added to the score as part of our extrinsically motivated design, disregarding whether it was a pairing instance or a failed scoring instance. The give away points were the user's emotional reward, and to prevent the user from feeling bad about his performance, which could potentially affect future pairing attempts. At this point, we give the user another reward for his or her performance. If the user recieved a perfect score, we displayed "You Performed Excellent! - Great job". If the user was able to successfully match three quarters of their values, we displayed "You Performed Well! - Good work". Otherwise, we displayed "You Performed Fair".

3.2 Plain Comparison Method

In order to have a meaningful comparison of our system, we developed a non-scoring version of the implementation which we refer to as the "Plain Comparison" method. The non-scoring version differed from the scoring version in significant ways. First, as the name indicates there was no scoring involved. Second, this version of the implementation had exactly half of the comparisons the scoring version had. This was due to the fact that we added artificial sets of numbers to the scoring version, in order to

account for the score. In other words, in the Plain Comparison method, there were 4 comparisons of 5 digit numbers, one number per quadrant. Since both the Plain and Scored Comparison techniques used the same amount of scoring instances, they each provided the same theoretical level of security.

4 Usability Evaluation

4.1 Testing Framework

To implement our pairing mechanisms as part of our study, we used two Nokia N97 phones. In order to ensure that the data we collected from our test subjects was as meaningful as possible, we gave users hands on experience with an implementation of our prototype deployed on these devices. Although we were attempting to simulate as realistic of a pairing scenario as possible, we still desired to eliminate unnecessary complexity from our testing system. To this end, we removed the traditional wireless link between the two phones that were used throughout our study. Rather than transmitting key information over this channel and using it to derive OOB pairing values, OOB strings were generated by the N97s on the fly using a pseudorandom number generator.

To capture the efficiency and efficacy of our prototypes, user actions were logged. We captured both the time taken to complete pairing tasks as well as any errors committed along the way. We provided users with a post-conditional questionnaire in order to gauge the opinions users had about our pairing system. Rather than developing our own survey instrument from scratch, we made use of the System Usability Scale (SUS) [2]. SUS is a technique for measuring the usability of a system in an efficient and reliable manner [2]. Our survey consisted of the questions that comprise SUS along with demographic questions and the following queries: (1) The method was enjoyable, (2) The method took a long time, (3) I would like to pair with another user's devices by making use of this method, and (4) I perceive this method to be secure. Users who evaluated the Scored Comparison pairing method were asked two additional questions: (5) I was annoyed with the fact that the score was shown at the end of the method, and (6) I would prefer to see the score after every comparison rather than at the end of the method.

4.2 Participant Information

We decided to conduct a between-subjects study with two subsets of 20 volunteers.We recruited these individuals from students working and studying at our college campus. We spread awareness of the study through emails and by signing people up face to face. Movie theater coupons were provided to participants as compensation for their assistance. We aggregated the following data about our users' backgrounds: age, gender, education level, experience pairing wireless devices, and experience with video games.

Table 1 presents demographic information about our study volunteers. The two sets of participants had similar levels of experience with wireless technology. 61.9%, of scored users, stated that they had performed wireless device pairing before, while just over half, 52.4% of plain users said they had done so. All users unanimously responded that they had played video games in the past.

4.3 Experimental Design

To begin a test with a subject, the administrator navigated to the Scored or Plain Comparison entry in the devices' application menus. The two phones were then turned over to the testers for the remaining duration of the study. Before allowing users to begin their pairing trials, instructions on how to utilize the Scored or Plain Comparison pairing application were displayed to them. After making their way through the instructions, a "Play" button appeared. This initiated the numeric comparisons. Users then proceeded to use either the Scored or Plain Comparison method. The act of pairing was repeated five times in order to provide subjects with ample experience. When finished, the "Play" button that initialized the pairing procedure was replaced with an "Exit" button. After getting several opportunities to perform simulated pairing with one of our two prototypes, each volunteer was presented with a post-conditional questionnaire. This included the demographic queries listed above followed by a series of five point Likert items. There were 14 of these questions for the plain survey and sixteen for the scoring version. The first ten of these comprised the System Usability Scale [2] with the slight modification that our pairing implementation was referred to as a method as opposed to a system.

Table 1. Demographics of participants

Demographic Information	**Plain Comparison**	**Scored Comparison**
AGE		
17 - 25	52.4	52.4
26 - 29	33.3	38.1
30 - 40	14.3	9.5
GENDER		
Male	42.9	61.9
Female	57.1	38.1
EDUCATION		
School graduate	14.2	19.0
Bachelor degree	28.6	38.1
Masters degree	52.4	33.3
Doctorate degree	4.8	9.5

Table 2. Summary of Experimental Results

	Scored Comparison		**Plain Comparison**	
Execution Time	Average	Standard Deviation	Average	Standard Deviation
	22.0 sec	11.2 sec	16.6 sec	13.4 sec
Error Rates	Safe	Fatal	Safe	Fatal
	1.8%	2.8%	6.5%	4.8%
SUS Scores	Average	Standard Deviation	Average	Standard Deviation
	74.3	11.7	69.2	15.2

4.4 Experimental Results

We have summarized the main results of our study in Table 2. Each volunteer who tested the Plain or Scored Comparison prototype executed 5 pairing sessions, accounting for a total of 100 test cases. The average execution time for the Scored Comparison method was 22.0 seconds for one device and 22.1 seconds for the other with standard deviations of 11.2 and 11.3 respectively. The Plain Comparison application completed in 16.7 seconds on average and a 13.6 standard deviation for the first device and 16.4 seconds with a standard deviation of 13.3 on the second. The small variation in execution time between the two mobile devices is attributable to the brief delay in pressing the "Finish" buttons on the two devices.

For the method of Scored Comparison, users were presented with 2 matching values and 6 mismatching values per pairing session for a total of 200 matches and 600 mismatched numbers over all 100 test cases. Out of these, 3 safe errors and 11 fatal errors were committed on one device. This yields a safe error rate of $3/200 = 1.5\%$ and a fatal error rate of $11/600 = 1.8\%$. On the other device, 4 safe errors and 23 fatal errors were committed, producing a $4/200 = 2.2\%$ safe error rate and a $23/600 = 3.8\%$ fatal error rate. The average of the error rates on the two devices is 1.8% safe and 2.8% fatal.

With respect to Plain Comparison, each pairing attempt was comprised of one matching entry and 3 mismatching entries, resulting in a total of 100 matches and 300 mismatches over the 100 test cases completed by our volunteers. On one device, 6 safe errors and 18 fatal errors were committed, while users committed 7 safe and 11 fatal errors on the other device. This results in a safe error rate of $6/100 = 6.0\%$ and a fatal error rate of $18/300 = 6.0\%$ for the first device; and a safe error rate of $7/100 = 7.0\%$ and a fatal error rate of $11/300 = 3.7\%$ for the second device. The average error rate across both devices is 6.5% and 4.8% for safe and fatal errors respectively.

On average, users assigned the Scored Comparison technique a 74.3 on the SUS scale with a standard deviation of 11.7. The Plain Comparison method was given a SUS rank of 69.2 and a standard deviation of 15.2. Test subjects who utilized the Scored Comparison method responded with a 3.5 on average when asked if they felt that the pairing technique they used was enjoyable, while those using Plain Comparison provided a slightly higher 3.7 response to this query on average. Scored users assigned an average score of 2.5 when asked if their pairing method took a long time. Volunteers who worked with the Plain Comparison method gave this question a 2.6 average. When asked if they would like to use the method, users of the Scored Comparison approach provided an average response of 3.6 and Plain Comparison volunteers gave an average rank of 3.7. When asked the question regarding their perception of the security of their given method, the average responses were 3.4 from scored users and 3.7 from users of the plain method. This was the last question posed to the Plain Comparison user group. Scored users provided 2.5 and 3.0 average responses when asked if they were annoyed about the score being withheld from them until after pairing and whether they would prefer to see the score after each comparison, respectively.

4.5 Interpretation and Analysis of Results

Looking at the average execution times, the Plain Comparison method was faster than the Scored Comparison variant by 5.2 seconds on one device and 5.7 seconds on the

other (unpaired t-tests, however, did not indicate any statistically significant difference between the two methods). This is intuitive because the latter method involved more comparisons due to the presence of scoring instances. It must be noted, however, that users were required to make twice as many comparisons with the scored approach but time taken was not doubled. This suggests that providing users with a score which assesses their pairing performance could possibly encourage them to compare individual pairing values more rapidly.

The effect of scoring on our participants' pairing performance was one of the most interesting results of our study. The inclusion of a score had a dramatic impact on users' ability to successfully detect which pairing entries were matches and which were not. The average safe error rate across the two devices fell from 6.5% to 1.8% when a score was in use. Similarly, both devices' average fatal error rate dropped from 4.8% to 2.8%. The two proportions z-tests indicate that this difference is significant. In particular, the change was statistically significant at a 95% confidence level with p-value of 0.0327 for safe errors, and marginally significant at a 90% confidence level with p-value of 0.0874 for fatal errors. Since the only difference between the two techniques was the presence or absence of a score, it can be deduced that providing users with a numeric evaluation of their performance caused them to be more aware of their pairing decisions. Users made errors at a far lower rate as a result.

Average SUS scores traditionally fall between 60 and 70 [3]. Therefore both the Scored and Plain Comparison pairing methods can be considered rather positive. We ran an unpaired t-test on the SUS responses provided by the users of each method. This resulted in a low-end p-value of 0.12, but indicated a lack of statistical significance between the two sets of answers. It is also worth noting that the standard deviation of our participant's SUS responses fell by 23%, or 3.5 points, between the group that did not have scoring integrated into their pairing solution and the one that did. This positive result indicates that there was more agreement among our testers regarding the usability of the scored solution than there was for the plain pairing method.

5 Conclusions

In this paper, we explored the use of visual scores as user incentives in the context of the secure device pairing task. By means of scoring and grading users' performance, we hoped to extrinsically motivate users and thereby improve the security and usability of the pairing task. This approach is particularly applicable to certain population segments, such as children, who are particularly vulnerable due to a lack of security awareness. We developed a scoring functionality that can be integrated with traditional pairing approaches. Further, we reported on a between-subjects study which we performed to evaluate the effect of scoring on the performance of numeric comparisons. The results of our study demonstrate that users are likely to commit fewer errors when working with the pairing approach based on scored comparisons. We believe that our work opens up a new area of research in usable security where security tasks can be combined with gaming elements including perceptible scores and other rewards.

Acknowledgements: The authors would like to thank their Financial Cryptography reviewers for their valuable feedback. This work is funded in part by NSF CNS-1255919 and a Google Research Award.

References

1. Uzun, E., Karvonen, K., Asokan, N.: Usability Analysis of Secure Pairing Methods. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 307–324. Springer, Heidelberg (2007)
2. Brooke, J.: SUS - A quick and dirty usability scale. In: Usability Evaluation in Industry (1996)
3. Lewis, J.R., Sauro, J.: The Factor Structure of the System Usability Scale. In: Kurosu, M. (ed.) Human Centered Design, HCII 2009. LNCS, vol. 5619, pp. 94–103. Springer, Heidelberg (2009)
4. Kainda, R., Flechais, I., Roscoe, A.W.: Usability and security of out-of-band channels in secure device pairing protocols. In: SOUPS: Symposium on Usable Privacy and Security (2009)
5. Saxena, N., Voris, J.: Pairing Devices with Good Quality Output Interfaces. In: Workshop on Wireless Security and Privacy (2008)
6. Ryan, R., Deci, E.L.: Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American Psychologist 55(1), 68–78 (2000)
7. Ryan, R.M., Deci, E.L.: Intrinsic and extrinsic motivations: Classic definitions and new directions. Contemporary Educational Psychology 25(1), 54–67 (2000)
8. Vaudenay, S.: Secure Communications over Insecure Channels Based on Short Authenticated Strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005)

RelationGram: Tie-Strength Visualization for User-Controlled Online Identity Authentication

Tiffany Hyun-Jin Kim[1], Akira Yamada[2], Virgil Gligor[1], Jason Hong[1], and Adrian Perrig[1]

[1] Carnegie Mellon University
{hyunjin,gligor,jasonh,perrig}@cmu.edu
[2] KDDI R&D Laboratories Inc.
yamada.akira@kddilabs.jp

Abstract. We consider the specific problem of how users can securely authenticate online identities (e.g., associate a Facebook ID with its owner). Based on prior social science research demonstrating that the social tie strength is a useful indicator of trust in many real-world relationships, we explore how tie strength can be visualized using well-defined and measurable parameters. We then apply the visualization in the context of online friend invitations and propose a protocol for secure online identity authentication. We also present an implementation on a popular online social network (i.e., Facebook). We find that tie strength visualization is a useful primitive for online identity authentication.

Keywords: Online identity authentication, tie strength visualization, trust establishment.

1 Introduction

Many real-world social interactions are based on various types of trust relations derived from strong social ties [4, 11–13]. As social interactions migrate from the physical to the online world, current systems do not provide many cues upon which users can base their identity authentication. For example, consider Facebook: how can a user be certain that a Facebook invitation is really from the claimed individual? As anyone can trivially set up a Facebook page with someone else's photo, Facebook provides almost no help in ensuring correspondence between the online and physical identity [1, 2, 5], even fooling security-conscious individuals [14]. Furthermore, Irani et al. [7] recently propose reverse social engineering attacks in Online Social Networks (OSNs), where the attacker sets up fake accounts and lets the victim discover and contact the fake account.

Although at first glance Public Key Infrastructures (PKI) and Pretty Good Privacy (PGP) appear to enable users to link an online identity to an individual, these approaches have significant shortcomings. Despite the long existence of Certification Authorities (CAs), few users have personal certificates, which are cumbersome to obtain. Moreover, CAs have recently suffered from several attacks [3]. Unlike PKIs, PGP is a distributed approach based on the notion of "Web of Trust" enabling identity certification. However, PGP's chains of trust are often unwieldy and offer limited security.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 69–77, 2013.

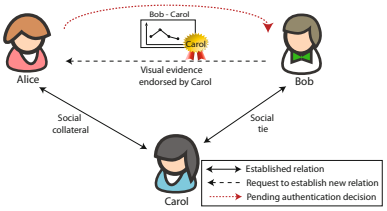


Fig. 1. Our approach for online identity authentication. Alice confirms Bob's invitation based on a *RelationGram* – a visual evidence of Bob and Carol's tie strength.

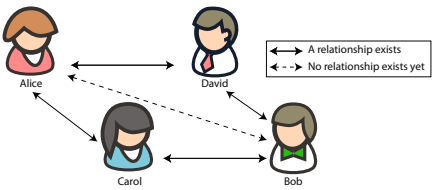


Fig. 2. An example of a trust graph. Bob wants to be Alice's online friend, and Carol and David are their mutual friends.

Personal recommendation systems may appear to address these issues, where a user would digitally sign a statement such as: "I trust that public key K_A really belongs to Alice, and I trust Alice to correctly validate other users." In the context of PGP, users could specify how much they trust others to assist validation of a chain of trust. Unfortunately, this approach suffers from the *distrust revelation problem*, defined by Kim et al. [8], where a polite or conflict-averse user does not want to publicly admit distrusting another individual, and thus specifies the untrusted user as trusted. Avoiding the distrust revelation problem is a core challenge we aim to address.

According to a social collateral model, users can accept online friend invitations from unknown senders based on explicitly endorsed recommendations made by social relations [9]. We extend the social collateral to provide different degrees of accountability for accepting friend invitations of unknown senders as follows: invitee Alice holds recommender Carol accountable for her actions such that Carol does not deceive Alice by creating or certifying bogus online identities. If Carol signs a bogus identity, the signature provides a non-repudiable statement, which may result in loss of the social collateral (e.g., loss of social relationship with Alice, loss of self-esteem, and feeling of guilt) for Carol. Preliminary evidence shows that social accountability can have a stronger deterrence effect than formal/legal punishment [6].

In this paper, we study how to enable users to authenticate OSN invitations to ensure that an invitation from an online individual is indeed tied to the correct individual. Our key idea is to derive tie strength between inviters and their mutual friends to represent real-world physical interactions, and provide it as evidence to empower users to authenticate online identities. More specifically, prior research indicates that in practice, tie strength can be represented using simple proxies such as frequency, reciprocity, and recency of communication, which we believe can be feasibly acquired by smartphones using call logs, emails, OSN comments, etc. Based on the simple proxies, we propose a *RelationGram* – a visualization of tie strength between an inviter and the invitee's friend(s) by which the invitee can easily understand the degree to which her friends know the inviter before she makes her own context-dependent authentication decisions.

As shown in Fig. 1, Alice can authenticate Bob's online identity using a RelationGram as follows. First, Alice personally knows Carol, and some level of social accountability exists between them. Second, Bob has sent a friend invitation to Alice and claims that Carol is a mutual friend. Before accepting the invitation, Alice wants to validate that Carol has a strong tie with Bob. Thanks to the RelationGram, visualizing tie strength between Bob and Carol, along with Carol's digital signature of the visualization and Bob's public key, Alice gains evidence and endorsement implying the strong tie. Hence, the combination of Carol's social accountability to Alice and the strong tie between Bob and Carol results in Alice authenticating Bob's online identity.

2 Problem Definition, Adversary Model, Assumptions

In this paper, we explore how to provide the evidence of social distance between users through a simple visualization that is endorsed in the form of a digital signature. Our goal is to help users correctly authenticate online identities using endorsed visualizations of social tie strength.

A challenge then is to accurately capture aspects of tie strength among OSN users and visually represent it to convey social proximity to other OSN users.

Relevance with Respect to Social Parameters. Every individual is unique and has different criteria in judging social distance. Hence, it is important to carefully select *relevant* parameters which accurately convey tie strength.

Robustness. Tie strength represented using social parameters must be robust against active attackers who attempt to claim close social proximity to others. Also, tie strength must be difficult to inflate due to social pressure (i.e., *distrust revelation problem* [8]), because users do not want to publicly admit that they do not trust another user.

Usability. It is crucial that OSN users can correctly interpret the visualization of relevant social parameters and easily understand social tie strength.

We consider an adversary whose goal is to manipulate social parameters for measuring tie strength such that he can claim to have a strong tie to a victim's friend. When the adversary deceives the victim who accepts the friend invitation, he can successfully gather sensitive personal information of the victim and possibly her friends.

We assume that trusted friends of a user do not misbehave due to their social accountability. Furthermore, we consider an attacker who compromises a user's account to be orthogonal to the issues we address in this paper.

3 Interpersonal Tie Strength Visualization

Bob wants to be Alice's online friend. Bob already knows Alice's friends who have social collateral with her. Rather than verifying any evidence provided by Bob (since Alice has not met Bob in person), we want to help Alice make a

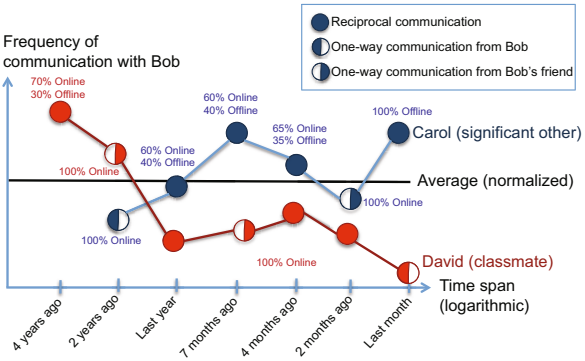


Fig. 3. RelationGram: a tie strength visualization from a trust graph

decision based on the evidence provided by her mutual friends who are socially accountable to her. In Fig. 2, these mutual friends would be Carol and David.

Using this scenario, Fig. 3 depicts a RelationGram, a visualization from which Alice can deduce appropriate social relationships between Bob and his friends, Carol and David. Below are the 7 parameters that the RelationGram visualizes:

Interaction Frequency. This parameter is displayed on the y-axis without a detailed scale, and we assume multiple types of communication (e.g., phone calls, emails, OSN comments). Note that we also display a normalized average line of the communication frequency (i.e., the normalized average frequency of communication between Carol and all her other friends). The same line also represents the average frequency of communication between David and all his other friends. Based on this average frequency, Alice can distinguish how frequently her friends interact with Bob as compared to their other friends, such that Alice can evaluate their tie strengths in a fair and unbiased manner.

Communication Reciprocity. This parameter is shown using the amount of shade on each plotted dot. For example, a fully-colored dot implies that the interaction is reciprocal, and a half-colored dot implies that the interaction is one-way where the originator is based on the side of the color as shown in Fig. 3.

Recency of Communication. The x-axis to the right represents a more recent time span than the left side. Fig. 3 confirms that Bob has communicated with Carol more recently (last month) than with David (2 months ago).

The Existence of More than One Mutual Friend. This parameter is depicted by the number of graphs in the same plot. In this case, Alice can infer that two of her friends are also friends with Bob.

Length of Relationship. The x-axis represents a logarithmic timeline which captures the length of Bob's social relationships with Carol and David.

Relationship Status. Fig. 3 display the relationship status (e.g., classmates, family, etc.) between the inviter and the mutual friends, and this relationship

label is assigned by the mutual friend. As a result, the inviter has no control over the relationship label.

Communication Type. The RelationGram labels the composition of *online* and *offline* communication frequencies, where online communication may include emails, OSN conversations, etc., and offline communication may include phone conversations, physical interactions, etc. The "100% online" labels will help indicate individuals who have only established a relationship over purely online means. As a result, Alice may be able to infer, with higher confidence, that Carol's graph indicates a strong tie with Bob with both online and offline interactions in Fig. 3.

We entrust full disclosure control to users such that the users themselves can decide to either reveal or protect their own graphs depicting their interactions with a particular set of friends. For extensive explanation of visualized parameters, including how they are computed, as well as security and privacy analysis of the RelationGram, please refer to our technical report for details [10].

4 Authenticating Online Friend Inviters

We introduce Indirect Friend Authentication (`IFA`) that can be used to authenticate an online friend inviter who is a friend of the invitee.

For this application context, we assume that people use smartphones for communication, as a greater number of smartphones are being sold.[1] We further assume that every user can use a smartphone to generate a public-private key pair, measure the parameters to represent tie strength, and automatically communicate with cloud application providers. Cloud application providers may be similar to Google which provides a backup service for contact information on phones, and we trust them for the availability of user information.

4.1 Indirect Friend Authentication

Alice receives an online invitation from someone named Bob, and this invitation indicates that they have two mutual friends: Carol and David.

Our Indirect Friend Authentication (`IFA`) protocol helps Alice authenticate Bob by leveraging two mutual friends. In a nutshell, the `IFA` protocol presents evidence that reflects the interpersonal tie strength between Alice's friend(s) and Bob in a RelationGram as explained in Section 3. Based on the visual evidence and the strength of social ties with her friends, Alice can exercise sound judgment when accepting Bob's invitation.

Evidence Generation. Bob and Carol mutually agree to disclose the information that reflects their social tie, and so do Bob and David. There are different ways of gathering information to represent these parameters. For example, Bob's

[1] As of Q3 of 2012, 56% of all mobile consumers in the U.S. own smartphones (`http://blog.nielsen.com/nielsenwire/online_mobile/nielsen-tops-of-2012-digital/`).

and David's phones can automatically detect and record the duration of a meeting, the call history between them, exchanged SMS text messages, Facebook posts, etc. Furthermore, OSNs can analyze information about their online message exchanging behavior, photos in which both are tagged together, etc. Note that these are the optional features that users opt-in for usage and users with privacy concerns may decline to use our protocols. When all the information representing tie strength is properly gathered on David's phone, it would sign the visual graph of their tie strength from David's perspective, sign Bob's public key, and hand it over to Bob. (With David's permission, this process is transparent to David.) Carol does the same for Bob. Thanks to Carol and David's release of the visual graphs, Bob has the evidence implying his social relations with Alice's friends and he inserts the graphs into the invitation.

Evidence Verification. When Alice receives the invitation from Bob, she has an option of seeing the RelationGram to determine the tie strength between Bob and her friends. Alice first verifies Carol's signed graph and Bob's public key using Carol's public key that she can retrieve from her own phone or from the cloud application provider. She also verifies David's graph in the same manner. When Alice successfully verifies that the graphs are generated by her real, trusted, and accountable friends (e.g., by verifying their digital signatures), she may decide to accept Bob's invitation based on his strength of social ties with her friends. However, it is possible that the authentication fails or the graphs do not convey strong ties, possibly due to some abnormal interaction conditions (e.g., David could have recently relocated, reducing his interaction frequency with Bob and limiting the communication medium to Facebook only). We emphasize that the visualization is one type of available evidence for users to make better tie strength evaluation, and the `IFA` protocol recommends that Alice gathers other evidence before accepting Bob's invitation.

For the security analysis of `IFA`, please refer to our technical report [10].

5 Implementation

We have implemented the `IFA` protocol in the context of Facebook friend invitations. Fig. 4 shows the architecture and the flow of our protocol to validate Facebook friend invitations.

"Do I Really Know You?" is an integrated Facebook web application such that (1) users can access their friends' invitations and present visualizations in a seamless manner and (2) the visualizations can be displayed on any smartphone with a web browser.

We have implemented our application using three types of APIs that Facebook provides: GraphAPI, OldREST API, and Facebook Query Language (FQL). This application seeks permission from users to retrieve posts and comments from their walls.

When a user invokes *"Do I Really Know You?"*, it gets a token which enables this application to access the Facebook database on behalf of the user. The application then queries the database according to the user's policy. First, the

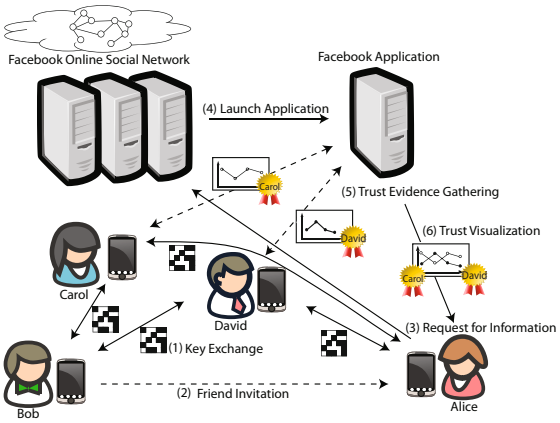


Fig. 4. Architecture of IFA protocol on Facebook. This diagram illustrates the process of IFA on Facebook as follows: (1) Alice exchanges keys with her friends Carol and David. (2) Bob wants to be Alice's friend. (3) Alice requests a RelationGram. (4) Facebook launches our application. (5) Our application obtains RelationGrams from Alice's friends after endorsement. (6) The application returns the endorsed RelationGram to Alice. (7) Alice verifies the tie strength between Bob and her friends (Carol and David) using the endorsed RelationGram.

application retrieves a list of pending friend invitations (via Facebook's *notifications.get* API). With at least one invitation, the application queries information about the inviter and the mutual friends (via *friends.getMutualFriends*). Then, the application retrieves a stream of wall information (via *stream.get* query with *limit=0* as a parameter).

When there are more than three mutual friends, this application prompts the user to select the "best" friends with whom he wants to infer the inviter's tie strengths. Based on the comments from the selected mutual friend's Facebook wall, this application computes the number of comments between each mutual friend and the inviter, and plots the interaction frequency for a RelationGram on the web browser.

We conducted an online user study with 93 participants to verify how much OSN users understand tie strengths between their own friends and the invitation senders, and whether our visual approach provided more convincing evidence to accept invitations as compared to the current OSN approaches. Participants found RelationGrams to be relevant with social parameters, robust against attackers, and easy to use. They also appreciated the visualizations to make informed authentication decisions. Please refer to the technical report for the evaluation results [10].

6 Discussion and Future Work

A first question is how usable such a system would really be, or whether it would be a burden on the user such that its utility would be negated. Although

further research is needed, several points indicate that the burden would be minimal. Existing systems could automatically collect interaction information without burdening users by aggregating email, SMS, Google+, etc. exchanges. Smartphones could also collect information about people users physically meet, through the use of voice recognition or by detecting the proximity of the other party's smartphone. Generation of evidence, endorsement (i.e., digital signature), and distribution to friends could also be automated. A minor burden would be configuration, where a user can decide which tie strength visualizations to share with others. This could occur through an opt-in process, where a user could add friends whose tie strength information could be shared.

Another important question is on incentives: would users really have incentives to share their tie strength visualizations, and how can privacy concerns be dealt with? In our user studies, it was clear that users seemed eager to obtain such information to validate online invitations with confidence. Although further studies are needed, we believe that people's inherent altruism that explains Internet phenomenons such as Wikipedia would also encourage users to share their tie strength visualizations, because little burden is required on their part, and they can help their friends to befriend each other with more safety.

7 Conclusion

Online user behavior is faced with an uncomfortable trade-off: should we really accept unauthenticated friends' invitations that might represent impersonation attempts to deceive; or should we deny them at the cost of losing potentially valuable relationships and become socially isolated? Currently, there is no secure and usable mechanism that would enable us to resolve this dilemma.

Our online identity authentication system implements a simple identity authentication logic in a visually compelling manner that is consistent with mental models derived from real-life experience. That is, it enables a casual user to authenticate online identities in a safe and easy-to-use manner.

References

1. Sophos Facebook ID Probe, http://www.sophos.com/pressoffice/news/articles/2007/08/facebook.html
2. Bilge, L., Strufe, T., Balzarotti, D., Kirda, E.: All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks. In: Proceedings of WWW (2009)
3. Economist. Duly notarised (September 2011), http://www.economist.com/blogs/babbage/2011/09/internet-security
4. Granovetter, M.S.: The Strength of Weak Ties. The American Journal of Sociology 78(6), 1360–1380 (1973)
5. Hamiel, N., Moyer, S.: Satan Is on My Friends List: Attacking Social Networks. In: Black Hat Conference (2008)
6. Hu, Q., Xu, Z., Dinev, T., Ling, H.: Does Deterrence Work in Reducing Information Security Policy Abuse by Employees? Communications of the ACM 84(6), 54–60 (2011)

7. Irani, D., Balduzzi, M., Balzarotti, D., Kirda, E., Pu, C.: Reverse Social Engineering Attacks in Online Social Networks. In: Holz, T., Bos, H. (eds.) DIMVA 2011. LNCS, vol. 6739, pp. 55–74. Springer, Heidelberg (2011)
8. Kim, T.H.-J., Bauer, L., Newsome, J., Perrig, A., Walker, J.: Challenges in Access Right Assignment for Secure Home Networks. In: Proceedings of USENIX Workshop on Hot Topics in Security, HotSec (August 2010)
9. Kim, T.H.-J., Gligor, V., Perrig, A.: Street-Level Trust Semantics for Attribute Authentication. In: Christianson, B., Malcolm, J., Stajano, F., Anderson, J. (eds.) Security Protocols 2012. LNCS, vol. 7622, pp. 96–115. Springer, Heidelberg (2012)
10. Kim, T.H.-J., Yamada, A., Gligor, V., Hong, J.I., Perrig, A.: RelationGrams: Tie-Strength Visualization for User-Controlled Online Identity Authentication. Technical Report CMU-CyLab-11-014, Carnegie Mellon University (2011)
11. Krackhardt, D.: The Strength of Strong Ties: The Importance of Philos in Organizations. In: Nohria, N., Eccles, R. (eds.) Networks and Organizations: Structure, Form, and Action, pp. 216–239 (1992)
12. Levin, D.Z., Cross, R.: The Strength of Weak Ties You Can Trust: The Mediating Role of Trust in Effective Knowledge Transfer. Management Science 50(11), 1477–1490 (2004)
13. Reagans, R., McEvily, B.: Network Structure and Knowledge Transfer: The Effects of Cohesion and Range. Administrative Science Quarterly 48(2), 240–267 (2003)
14. Ryan, T.: Getting in Bed with Robin Sage. In: Black Hat Conference (2010)

Practical Fully Simulatable Oblivious Transfer with Sublinear Communication

Bingsheng Zhang[1], Helger Lipmaa[2], Cong Wang[3], and Kui Ren[1]

[1] State University of New York at Buffalo, United States
[2] University of Tartu, Estonia
[3] City University of Hong Kong, China

Abstract. During an adaptive k-out-of-N oblivious transfer (OT), a sender has N private documents, and a receiver wants to adaptively fetch k documents from them such that the sender learns nothing about the receiver's selection and the receiver learns nothing more than those chosen documents. Many fully simulatable and universally composable adaptive OT schemes have been proposed, but those schemes typically require $\mathcal{O}(N)$ communication in the initialization phase, which yields $\mathcal{O}(N)$ overall communication. On the other hand, in some applications, the receiver just needs to fetch a small number of documents, so the initialization cost dominates in the entire protocol, especially for 1-out-of-N OT. We propose the first fully simulatable adaptive OT with sublinear communication under the DDH assumption in the plain model. Our scheme has $\mathcal{O}(N^{1/2})$ communication in both the initialization phase and each transfer phase. It achieves better (amortized) overall communication complexity compared to existing schemes when $k = \mathcal{O}(N^{1/2})$.

Keywords: Adaptive oblivious transfer, fully simulatable security, sublinear communication, zero knowledge batch argument.

1 Introduction

Data outsourcing and online shopping have become during the recent years. To address the related information security and privacy concerns, many cryptographic protocols have been studied to accomplish tasks with minimal information disclosure. Consider the case that an online store sells digital goods, such as movies, books, music, etc. The buyer wants to purchase some of them without revealing his/her choices. Here, we assume that there is a uniform price for those goods in the same category, e.g. movies. [1] Oblivious transfer (OT) is a handy primitive, which has found its usage in many security applications with this kind of privacy requirements as in the aforementioned case.

OT family mainly consists of 1-out-of-2 OT, denoted as OT_1^2, 1-out-of-N OT, denoted as OT_1^N, k-out-of-N OT, denoted as OT_k^N and k-out-of-N OT with adaptive queries (also known as adaptive k-out-of-N OT), denoted as $\mathsf{OT}_{k\times 1}^N$.

[1] It is also possible to protect the buyer's privacy even if all the goods are paid at unique prices. This problem is addressed by priced OT, see [33] for details.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 78–95, 2013.

OT_1^2 is widely used in secure multi-party/two-party computation, for example, it serves as an important building block of Yao's garbled circuit [34] in two-party *secure function evaluation* (SFE). Based on earlier work of Lipmaa [26], Ishai and Paskin [18] showed how to privately evaluate a branching program with OT_1^2. OT_1^N also has rich applications in financial cryptography, for instance, one can use OT_1^N to construct simultaneous contract signing schemes [29].

We focus on $\mathsf{OT}_{k\times 1}^N$ as well as its special case, OT_1^N when $k = 1$. During an $\mathsf{OT}_{k\times 1}^N$ protocol, a sender has N private documents, and a receiver can adaptively fetch k documents from them such that the sender learns nothing about the receiver's selection and the receiver learns nothing more than those k documents. The notion of $\mathsf{OT}_{k\times 1}^N$ was first introduced by Naor and Pinkas [30] who also gave several applications, including oblivious search. In an oblivious search protocol, the server owns a sorted database that the client wants to search. Given the element that the client is searching for, they invoke an $\mathsf{OT}_{k\times 1}^N$ protocol using binary search, where $k = \log N$. After the protocol execution, the client can determine whether the element is in the database while the server only has revealed limited database information ($\log N$ elements).

There is always a trade-off between security and efficiency. Due to bandwidth limitation, most applications employ OT schemes with so-called *half-simulation* security, where the sender's and receiver's security are handled separately. Such OT_1^N can achieve logarithmic communication [10] or sublinear computation [27] and OT_k^N can achieve optimal rate [1]. The receiver's security is defined as indistinguishability of the sender's view of the protocol when the receiver's choices are different. The sender's security follows the real-world/ideal-world paradigm and guarantees that for any malicious receiver in the real world there is a receiver in an ideal world where OT is implemented by a trusted party. However, this security definition is vulnerable to the selective failure attack [30]. Namely, the sender is able to cause protocol failure based on some property of the receiver's selection. Based on an arbitrary semisimulatable OT protocol, Laur and Lipmaa proposed a consistent OT protocol [24] of virtually the same complexity that allows to detect selective failures, but still does not obtain ideal security.

On the other hand, all existing *fully simulatable* and *universally composable* adaptive OT schemes typically require $\mathcal{O}(N)$ communication in the initialization phase. The huge initialization cost is not acceptable in many applications, especially when the receiver is only required to fetch a small number of documents. For example, in 1-out-of-N OT the initialization cost dominates the entire protocol, so the overall communication cost becomes $\mathcal{O}(N)$. Can we make OT more communication-efficient without sacrificing its security level? In this paper, we try to answer this question by investigating a practical fully simulatable $\mathsf{OT}_{k\times 1}^N$ scheme with sublinear communication.

Our Contribution and Related Work. In theory, one can transform any secure OT protocol in semi-honest model to an OT protocol that is secure against malicious adversaries by plug-in *zero-knowledge* (ZK) proofs/arguments. To achieve sublinear communication, we may use *probabilistically checkable proofs* (PCP), e.g., [4] or a sublinear ZK argument [17,28]. The problem with such

Table 1. Comparison of $\mathsf{OT}^N_{k\times 1}$ schemes. The trivial factor $\log N$ is ignored.

Scheme	Init Cost	Transfer Cost	Assumption	Security
Prot. 3.1 [31]	$\mathcal{O}(N)$	$\mathcal{O}(N^{1/2})$	DDH	Full Sim
[5]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	q-Power DDH + q-Strong DH	Full Sim
[13]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	DLIN + q-Hidden LRSW	UC
[20]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	Dec. n-th Residuosity/DDH	Full Sim
[19]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	Dec. Residuosity + q-DDHI	Full Sim
[33]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	DLIN + q-Hidden SDH + q-TDH	UC
[21]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	DDH	Full Sim
[14]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	3-DDH + DLIN	Full Sim
[22]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	DDH/DLIN/DCR/QR/LWE	Full Sim
[35]	$\mathcal{O}(N)$	$\mathcal{O}(1)$	DDH/Dec. n-th Residuosity	Full Sim
this work	$\mathcal{O}(N^{1/2})$	$\mathcal{O}(N^{1/2})$	DDH	Full Sim

approaches is that the OT protocol has to be reduced to some NP-complete language, which is neither efficient nor practical. In this paper, we propose the first fully simulatable $\mathsf{OT}^N_{k\times 1}$ scheme with $\mathcal{O}(\sqrt{N})$ communication in both the initialization phase and each transfer phase based on the standard DDH assumption. When $k = \mathcal{O}(\sqrt{N})$, our $\mathsf{OT}^N_{k\times 1}$ scheme has better amortized overall communication complexity compared to existing schemes. In order to achieve sublinear communication complexity, we constructed a few efficient batch ZK arguments, such as masked multi-exponentiation batch argument (c.f. Sect. 4.3, below). We use Lim's multi-exponentiation algorithm in our implementation, and a benchmark is given at the end of this paper.

We now give a survey on recent *fully simulatable* and *universally composable* $\mathsf{OT}^N_{k\times 1}$ schemes. As shown by Canetti and Fischlin [6], an OT cannot be realized in UC security without additional trusted setup assumptions. All the UC-secure $\mathsf{OT}^N_{k\times 1}$ schemes mentioned here are in *common reference string* (CRS) model, i.e. $\mathcal{F}_{crs}$-hybrid model. Whereas many fully simulatable $\mathsf{OT}^N_{k\times 1}$ schemes in this survey, as well as our construction, are realized in the plain model. Table 1 lists several existing $\mathsf{OT}^N_{k\times 1}$ schemes, together with our proposed scheme for comparison. In 2007, Camenisch, Neven and shelat [5] proposed $\mathsf{OT}^N_{k\times 1}$ under the q-strong Diffie-Hellman and q-power decisional Diffie-Hellman assumptions in bilinear groups. They used signatures as a key ingredient in their scheme. Later, Green and Hohenberger [12] showed an $\mathsf{OT}^N_{k\times 1}$ in random oracle model under decisional bilinear Diffie-Hellman assumption. In their scheme, the sender encrypts message m_i by identity-based encryption under identity i. The receiver executes a blind key extraction protocol such that he/her can obliviously obtain the secret key of any identity. In 2008, Green and Hohenberger [13] introduced another OT that achieves UC security in the $\mathcal{F}_{crs}$-hybrid model, using a Groth-Sahai non-interactive ZK (NIZK) proof for pairing product equations. The scheme is based on the decisional linear and q-Hidden LRSW assumptions. Jarecki and

Liu [19] simplified the Camenisch et al. construction to a fully simulatable OT under the composite decisional residuosity and q-decisional Diffie-Hellman Inversion assumptions. Rial, Kohlweiss and Preneel [33] presented an adaptive priced OT that achieves UC security using "assisted decryption". In 2009, Kurosawa and Nojima [20] gave adaptive OT constructions based on Paillier and ElGamal encryption schemes. Later, Kurosawa, Nojima and Phong [21] improved the scheme [20] by increasing the complexity of initialization phase. In 2011, Green and Hohenberger [14] proposed another fully simulatable OT under decisional 3-party Diffie-Hellman assumption. Recently, Kurosawa et al. [22] and Zhang [35] generalized the scheme in [21] to various schemes with different assumptions.

We emphasize that Prot. 3.1 in [31] is essentially different from our scheme. In [31], the sender first 'commits' the documents to the receiver in the initialization phase. This step takes $\mathcal{O}(N)$ communication, because each "commitment" serves as an encryption and the receiver should be able to extract (or decrypt) the committed document from it later. Therefore, it is not possible to directly plug a succinct commitment scheme in the Naor-Pinkas scheme.

2 Preliminaries

Let $[n] := \{1, \ldots, n\}$. By $\boldsymbol{a}$, we denote a vector $\boldsymbol{a} = (a_1, \ldots, a_n)^T$. When S is a set, $a \leftarrow_\$ S$ means that a is uniformly and randomly chosen from S. Let λ be the security parameter. By $A \overset{c}{\approx} B$, we mean that A and B are computationally indistinguishable. We abbreviate *probabilistic polynomial-time* as p.p.t. and let $\mathrm{poly}(\cdot)$ be a polynomially-bounded function.

Elliptic Curves Over $\mathbb{F}_p$. The implementation of our scheme is based on elliptic curve groups for efficiency. Let $\sigma := (p, a, b, g, q, \zeta)$ be the elliptic curve domain parameters over $\mathbb{F}_p$, consisting of a prime p specifying the finite field $\mathbb{F}_p$, two elements $a, b \in \mathbb{F}_p$ specifying an elliptic curve $E(\mathbb{F}_p)$ defined by $E : y^2 \equiv x^3 + ax + b \pmod p$, a base point $g = (x_g, y_g)$ on $E(\mathbb{F}_p)$, a prime q which is the order of g, and an integer ζ which is the cofactor $\zeta = \#E(\mathbb{F}_p)/q$. We denote the cyclic group generated by g by $\mathbb{G}$, and it is assumed that the DDH assumption holds over $\mathbb{G}$, that is for all p.p.t. adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(\mathcal{A}) = \left| \Pr \begin{bmatrix} x, y \leftarrow_\$ \mathbb{Z}_q; b \leftarrow_\$ \{0,1\}; h_0 = g^{xy}; \\ h_1 \leftarrow_\$ \mathbb{G} : \mathcal{A}(g, g^x, g^y, h_b) = b \end{bmatrix} - \frac{1}{2} \right| \leq \epsilon(\lambda) \ ,$$

where $\epsilon(\cdot)$ is a negligible function.

Security Definition (Fully Simulation Security). We use the same security definition as in [30,5,14]. Let $(\mathscr{S}_I, \mathscr{R}_I, \mathscr{S}_T, \mathscr{R}_T)$ be an $\mathsf{OT}^N_{k\times 1}$ protocol. Let S_*, R_* be private states. During the initialization phase, the sender sets $S_0 \leftarrow \mathscr{S}_I(m_1, \ldots, m_N)$, and the receiver sets $R_0 \leftarrow \mathscr{R}_I()$. During the ℓ-th transfer phase, $\ell \in [k]$, the sender sets $S_\ell \leftarrow \mathscr{S}_T(S_{\ell-1})$, and the receiver sets $(R_\ell, m^*_{\sigma_\ell}) \leftarrow \mathscr{R}_T(R_{\ell-1}, i_\ell)$, where $i_\ell \in [N]$ is the index of the message to be received. $m^*_{\sigma_\ell} = m_{\sigma_\ell}$ if retrieval succeeds, $m^*_{\sigma_\ell} = \perp$ if fails. The security of an

$\mathsf{OT}^N_{k\times 1}$ scheme is defined in the real-world/ideal-world paradigm with static corruption, i.e. the adversary $\mathcal{A}$ can only choose to corrupt either the sender or the receiver at the beginning of the experiment.

Real experiment. In experiment $\mathbf{Real}_{\hat{\mathbf{S}},\hat{\mathbf{R}}}(N,k,m_1,\ldots,m_N,\mathcal{I})$, a presumably cheating sender $\hat{\mathbf{S}}$ is given messages $(m_1,\ldots,m_N)$ as input and interacts with a presumably cheating receiver $\hat{\mathbf{R}}(\mathcal{I})$, where $\mathcal{I}$ is a selection algorithm that on input messages $\{m_{i_t}\}_{t=1}^{\ell-1}$ outputs the index i_ℓ of the next message to be queried. In the initialization phase, $\hat{\mathbf{S}}$ and $\hat{\mathbf{R}}$ output the initial states S_0 and R_0. In the ℓ-th transfer phase, for $\ell\in[k]$, the sender runs $S_\ell\leftarrow\hat{\mathbf{S}}(S_{\ell-1})$, and the receiver runs $(R_\ell,m^*_{i_\ell})\leftarrow\hat{\mathbf{R}}(R_{\ell-1})$. After the k-th transfer, the output of the experiment $\mathbf{Real}_{\hat{\mathbf{S}},\hat{\mathbf{R}}}$ is the tuple (S_k,R_k).

We define the honest sender algorithm $\mathbf{S}$ as the one that runs $\mathscr{S}_I(m_1,\ldots,m_N)$ in the initialization phase, runs $\mathscr{S}_T()$ during each transfer phase, and returns $S_k=\emptyset$ as its final output. The honest receiver algorithm $\mathbf{R}$ runs $\mathscr{R}_I()$ in the initialization phase, runs $\mathscr{R}_T(R_{\ell-1},i_\ell)$ during the ℓ-th transfer phase, where the index i_ℓ is generated by $\mathcal{I}$, and returns $R_k=(m_{i_1},\ldots,m_{i_k})$ as its final output.

Ideal experiment. In experiment $\mathbf{Ideal}_{\hat{\mathbf{S}}',\hat{\mathbf{R}}'}(N,k,m_1,\ldots,m_N,\mathcal{I})$, the presumably cheating sender $\hat{\mathbf{S}}'$ and the presumably cheating receiver $\hat{\mathbf{R}}'$ communicate with the ideal functionality $\mathcal{F}_{OT}^{N\times 1}$. In the initialization phase, $\hat{\mathbf{S}}'(m_1,\ldots,m_N)$ sends messages $m_1^*,\ldots,m_N^*$ to $\mathcal{F}_{OT}^{N\times 1}$. In the ℓ-th transfer phase, $\ell\in[k]$, $\hat{\mathbf{R}}'(\mathcal{I})$ sends to $\mathcal{F}_{OT}^{N\times 1}$ an index i_ℓ^*. $\mathcal{F}_{OT}^{N\times 1}$ then sends a tag '$\mathsf{Received}$' to $\hat{\mathbf{S}}'$, and $\hat{\mathbf{S}}'$ replies a bit $b_\ell\in\{0,1\}$ to $\mathcal{F}_{OT}^{N\times 1}$. If $b_\ell=1$ and $i_\ell^*\in[N]$, $\mathcal{F}_{OT}^{N\times 1}$ sends $m^*_{i_\ell^*}$ to $\hat{\mathbf{R}}'$; otherwise, it sends $\perp$ to $\hat{\mathbf{R}}'$. After the k-th transfer, the output of the experiment $\mathbf{Ideal}_{\hat{\mathbf{S}}',\hat{\mathbf{R}}'}$ is the tuple (S_k,R_k).

We define the honest sender algorithm $\mathbf{S}'(m_1,\ldots,m_N)$ as the one that sends $m_1,\ldots,m_N$ to $\mathcal{F}_{OT}^{N\times 1}$ in the initialization phase, and sends $b_\ell=1$ during each transfer phase, and returns $S_k=\emptyset$ as its final output. The honest receiver $\mathbf{R}'$ submits the indices i_ℓ that generated by $\mathcal{I}$ to $\mathcal{F}_{OT}^{N\times 1}$, and returns $R_k=(m_{i_1},\ldots,m_{i_k})$ as its final output.

Sender Security. An $\mathsf{OT}^N_{k\times 1}$ is sender-secure if for every real-world p.p.t. receiver $\hat{\mathbf{R}}$, there exists an ideal-world p.p.t. receiver $\hat{\mathbf{R}}'$, s.t. for every $N=\mathrm{poly}(\lambda)$, $k\in[N]$, $(m_1,\ldots,m_N)$, selection algorithm I, and p.p.t. distinguisher $\mathbf{D}$,

$$\mathbf{Real}_{\mathbf{S},\hat{\mathbf{R}}}(N,k,m_1,\ldots,m_N,\mathcal{I}) \quad\overset{c}{\approx}\quad \mathbf{Ideal}_{\mathbf{S}',\hat{\mathbf{R}}'}(N,k,m_1,\ldots,m_N,\mathcal{I})\ .$$

Receiver Security. An $\mathsf{OT}^N_{k\times 1}$ is receiver-secure if for every real-world p.p.t. sender $\hat{\mathbf{S}}$, there exists an ideal-world p.p.t. sender $\hat{\mathbf{S}}'$, s.t. for every $N=\mathrm{poly}(\lambda)$, $k\in[N]$, $(m_1,\ldots,m_N)$, selection algorithm $\mathcal{I}$, and p.p.t. distinguisher $\mathbf{D}$,

$$\mathbf{Real}_{\hat{\mathbf{S}},\mathbf{R}}(N,k,m_1,\ldots,m_N,\mathcal{I}) \quad\overset{c}{\approx}\quad \mathbf{Ideal}_{\hat{\mathbf{S}}',\mathbf{R}'}(N,k,m_1,\ldots,m_N,\mathcal{I})\ .$$

Definition 1. $\mathsf{OT}^N_{k\times 1}$ *is fully simulatable iff it is both sender- and receiver-secure.*

Special Honest Verifier Zero-Knowledge Argument. Let $\mathcal{R}$ be a polynomial time decidable binary relation, we say w is a witness for a statement x if $(x,w)\in\mathcal{R}$. We define the language $\mathcal{L} := \{x \mid \exists w : (x,w)\in\mathcal{R}\}$ as the set of all statements x that have a witness w for the relation $\mathcal{R}$. Let a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ be two p.p.t. interactive algorithms. Denote $\tau \leftarrow \langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle$ as the public transcript produced by $\mathcal{P}$ and $\mathcal{V}$. After the protocol, $\mathcal{V}$ accepts iff $\Phi(x,\tau)=1$, where Φ is a predicate function.

Definition 2. *We say $(\mathcal{P},\mathcal{V})$ is a perfectly complete argument for a relation $\mathcal{R}$ if for all non-uniform p.p.t. interactive adversaries $\mathcal{A}$ it satisfies*

- *Perfect completeness:* $\Pr[(x,w)\leftarrow\mathcal{A}; \tau\leftarrow\langle\mathcal{P}(x,w),\mathcal{V}(x)\rangle : (x,w)\in\mathcal{R}\vee \Phi(x,\tau)=1]=1$*;*
- *Computational soundness:* $\Pr[x\leftarrow\mathcal{A};\tau\leftarrow\langle\mathcal{A},\mathcal{V}(x)\rangle : x\notin\mathcal{L}\wedge\Phi(x,\tau)=1]\approx 0$*.*

Denote $\mathcal{V}(x;r)$ as the verifier $\mathcal{V}$ on input x, given r as the randomness. An argument $(\mathcal{P},\mathcal{V})$ is *public coin* if the verifier $\mathcal{V}$ picks his challenges randomly and independently of the messages sent by the prover $\mathcal{P}$.

Definition 3. *A public coin argument $(\mathcal{P},\mathcal{V})$ is called a perfect special honest verifier zero-knowledge (SHVZK) argument for a relation $\mathcal{R}$ if there exists a p.p.t. simulator $\mathcal{S}$ such that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\begin{aligned}\Pr[(x,w,r)\leftarrow\mathcal{A};&\tau\leftarrow\langle\mathcal{P}(x,w),\mathcal{V}(x;r)\rangle : (x,w)\in\mathcal{R}\wedge\mathcal{A}(\tau)=1]\\ &=\Pr[(x,w,r)\leftarrow\mathcal{A};\tau\leftarrow\mathcal{S}(x;r) : (x,w)\in\mathcal{R}\wedge\mathcal{A}(\tau)=1]\ .\end{aligned}$$

We define the SHVZK *argument of knowledge* similiarly to the definition of [15,16,2]; namely, given an adversary that produces an acceptable argument with probability p, there exists a witness-extended emulator that produces a similar argument with probability p and outputs a witness. The standard definition of "*proofs of knowledge* (PoK)" by Bellare and Goldreich [3] does not work for "*arguments of knowledge* (AoK)". See [8] for more discussion of this issue and an alternative definition of knowledge soundness.

Definition 4. *A public coin argument $(\mathcal{P},\mathcal{V})$ has a witness extended emulator if for all p.p.t. $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{X}=\mathcal{X}^{\mathcal{P}^*}$ such that for all non-uniform polynomial time adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l}(x,\psi)\leftarrow\mathcal{A};\\ \tau\leftarrow\langle\mathcal{P}^*(x,\psi),\mathcal{V}(x)\rangle :\\ \mathcal{A}(\tau)=1\end{array}\right]=\Pr\left[\begin{array}{l}(x,\psi)\leftarrow\mathcal{A};\\ (\tau,w)\leftarrow\mathcal{X}^{\langle\mathcal{P}^*(x,\psi),\mathcal{V}(x)\rangle}(x,\psi):\\ \mathcal{A}(\tau)=1\wedge(\Phi(x,\tau)=0\vee(x,w)\in\mathcal{R})\end{array}\right].$$

Here, $\mathcal{X}$ has access to a transcript oracle $\langle\mathcal{P}^*(x,\psi),\mathcal{V}(x)\rangle$ that can be rewound to a particular round and run again with the verifier using fresh randomness. Let ψ be the state of $\mathcal{P}^*$, including the randomness. Whenever $\mathcal{P}^*$ is able to make a convincing argument with state ψ, the emulator $\mathcal{X}$ can extract a witness w.

3 Building Blocks

Additively Homomorphic Public-Key Cryptosystem. The lifted ElGamal public-key cryptosystem consists of the following 4 p.p.t. algorithms:

- $\mathsf{Gen}_{\mathsf{gk}}(1^\lambda)$: inputs a security parameter λ, and outputs $\sigma := (p, a, b, g, q, \zeta)$.
- $\mathsf{Gen}_{\mathsf{pkc}}(\sigma)$: picks $sk \leftarrow_{\$} \mathbb{Z}_q^*$, sets $pk := h = g^{sk}$, and outputs (pk, sk).
- $\mathsf{Enc}_{pk}(m; r)$: outputs $e := (e_1, e_2) = (g^r, g^m h^r)$.
- $\mathsf{Dec}_{sk}(e)$: outputs $\mathsf{DL}_g(e_2 \cdot e_1^{-sk})$, where $\mathsf{DL}_g(x)$ is the discrete logarithm of x. (Note that since $\mathsf{DL}_g(\cdot)$ is not efficient, the message space should be a small set, say $\{0,1\}^\xi$, for $\xi \leq 30$.)

It is well known that lifted ElGamal encryption scheme is IND-CPA secure under the DDH assumption. It is additively homomorphic: $\mathsf{Enc}_{pk}(m_1; r_1) \cdot \mathsf{Enc}_{pk}(m_2; r_2) = \mathsf{Enc}_{pk}(m_1 + m_2; r_1 + r_2)$.

Additively Homomorphic Succinct Vector Commitment. In our protocol, we use a generalized version of the Pedersen commitment scheme [32]. The generalized Pedersen commitment scheme consists of the following 4 algorithms:

- $\mathsf{Gen}_{\mathsf{gk}}(1^\lambda)$: inputs security parameter λ, and outputs $\sigma := (p, a, b, g, q, \zeta)$.
- $\mathsf{Gen}_{\mathsf{ped}}(\sigma)$: outputs distinct generators $ck := (g_1, \ldots, g_n, f)$.
- $\mathsf{Com}_{ck}(\boldsymbol{m}; r)$: outputs a commitment $c := f^r \prod_{i=1}^n g_i^{m_i}$ for $\boldsymbol{m} \in \mathbb{Z}_q^n$ and $r \in \mathbb{Z}_q$.
- $\mathsf{Open}_{ck}(c)$: outputs $\boldsymbol{m} \in \mathbb{Z}_q^n, r \in \mathbb{Z}_q$ such that $c = f^r \prod_{i=1}^n g_i^{m_i}$. Open also receives some private information that was created during the commitment.

The generalized Pedersen commitment is perfect hiding and computationally binding if the discrete logarithm problem is hard in $\mathbb{G}$. It is additively homomorphic: $\mathsf{Com}_{ck}(\boldsymbol{m_1}; r_1) \cdot \mathsf{Com}_{ck}(\boldsymbol{m_2}; r_2) = \mathsf{Com}_{ck}(\boldsymbol{m_1} + \boldsymbol{m_2}; r_1 + r_2)$.

In the plain model, if Alice wants to commit N elements to Bob, the best communication complexity with generalized Pedersen commitment scheme is $\mathcal{O}(\sqrt{N})$. Namely, Bob first sends to Alice $n := \sqrt{N}$ commitment keys ck, and Alice commits and sends to Bob $\sqrt{N}$ commitments.

4 Fully Simulatable $\mathsf{OT}_{k\times 1}^N$ with Square-Root Communication

We now propose a fully simulatable $\mathsf{OT}_{k\times 1}^N$ protocol with a square-root overall communication complexity. The basic idea comes from the classic KO *private information retrieval* (PIR) scheme [23]. Intuitively, when not concerned about privacy, the receiver sends two n-dimensional unit-vectors $\boldsymbol{u}, \boldsymbol{v}$ to the sender, where $n = \sqrt{N}$. The sender computes and sends $m^* = \boldsymbol{u} \cdot \boldsymbol{M} \cdot \boldsymbol{v}^T$ to the receiver, where $\boldsymbol{M} = \{m_{i,j}\}_{i,j\in[n]}$ is the sender's database.

Both the ElGamal encryption scheme and the generalized Pedersen commitment scheme are based on elliptic curves, so the membership of a group element is

Common input: $\sigma := (p, a, b, g, q, \zeta) \leftarrow \mathsf{Gen}_{\mathsf{gk}}(1^\lambda)$.
Sender's input: $\boldsymbol{M} := \{m_{i.j}\}_{i,j\in[n]} = \{\boldsymbol{m}_j\}_{j\in[n]}$, where $\boldsymbol{m}_j = (m_{1,j}, \ldots, m_{n,j})^T$.
Receiver's output: $i_\ell := m_{x_\ell, y_\ell}$ for the ℓ-th transfer phase, where $x_\ell, y_\ell \in [n]$.

Initialization Phase:

1. $\mathbf{R}$ computes $(pk, sk) \leftarrow \mathsf{Gen}_{\mathsf{pkc}}(\sigma)$ and $ck \leftarrow \mathsf{Gen}_{\mathsf{ped}}(\sigma)$, and sends pk, ck to $\mathbf{S}$.
2. $\mathbf{S}$ picks $\boldsymbol{r} \leftarrow_\$ \mathbb{Z}_q^n$ and sends $\{c_i := \mathsf{Com}_{ck}(\boldsymbol{m}_i; r_i)\}_{i=1}^n$, where $\boldsymbol{r} := (r_1, \ldots, r_n)^T$.

The ℓ-th Transfer Phase:

1. $\mathbf{R}$ sets two unit vectors $\boldsymbol{u}, \boldsymbol{v} \in \{0,1\}^n$ according to (x_ℓ, y_ℓ), i.e., $u_{x_\ell} = 1$, $v_{y_\ell} = 1$ and the rest are 0's. He picks $\boldsymbol{t_u}, \boldsymbol{t_v} \leftarrow_\$ \mathbb{Z}_q^n$, and sends $\boldsymbol{e_u} := \{e_u^{(i)} := \mathsf{Enc}_{pk}(u_i; t_u^{(i)})\}_{i=1}^n$ and $\boldsymbol{e_v} := \{e_v^{(i)} := \mathsf{Enc}_{pk}(v_i; t_v^{(i)})\}_{i=1}^n$ to $\mathbf{S}$, where $\boldsymbol{u} := (u_1, \ldots, u_n)^T$, $\boldsymbol{v} := (v_1, \ldots, v_n)^T$, $\boldsymbol{t_u} := (t_u^{(1)}, \ldots, t_u^{(n)})^T$ and $\boldsymbol{t_v} := (t_v^{(1)}, \ldots, t_v^{(n)})^T$.
2. $\mathbf{R}$ proves to $\mathbf{S}$ that $\boldsymbol{e_u}, \boldsymbol{e_v}$ encrypt unit vectors by *unit vector PoK* (c.f. Sect. 4.1).
3. $\mathbf{S}$ picks $\boldsymbol{t_s} \leftarrow_\$ \mathbb{Z}_q^n, \boldsymbol{a}, \boldsymbol{t_a} \leftarrow_\$ \mathbb{Z}_q^n, t_z \leftarrow_\$ \mathbb{Z}_q$. $\mathbf{S}$ sets for $i \in [n]$, $c_a^{(i)} = \mathsf{Com}_{ck}(a_i; t_a^{(i)})$, $w_i = \mathsf{Enc}_{pk}(a_i; t_s^{(i)}) \prod_{j=1}^n (e_u^{(j)})^{m_{j,i}}$, and $z = \mathsf{Enc}_{pk}(0; t_z) \prod_{j=1}^n (e_v^{(j)})^{a_j}$.
4. $\mathbf{S}$ sends $\boldsymbol{c_a}, \boldsymbol{w}, z$ to $\mathbf{R}$.
5. $\mathbf{S}$ proves to $\mathbf{R}$ that
   - $\boldsymbol{w}, \boldsymbol{e_u}, \boldsymbol{c}, \boldsymbol{c_a}$ are consistent by *masked multi-exp batch AoK* (c.f. Sect. 4.3);
   - $\boldsymbol{e_u}, z, \boldsymbol{c_a}$ are consistent by *multi-exp AoK* (c.f. Sect. 4.2, below).
6. $\mathbf{R}$ computes $g^{a_{y_\ell}} = z_2 \cdot z_1^{-sk}$ and then returns m_{i_ℓ} as $\mathsf{DL}_g(w_{x_\ell,2} \cdot w_{x_\ell,1}^{-sk} \cdot (g^{a_{y_\ell}})^{-1})$.

Fig. 1. Fully Simulatable $\mathsf{OT}_{k\times 1}^N$ With Square-root Communication

efficiently decidable. Hence, during our protocol, if the message consists of group elements/generators, the parties always first check their group membership. We will not mention this step in the protocol description explicitly.

We give the protocol description in Fig. 1. Our $\mathsf{OT}_{k\times 1}^N$ scheme consists of the initialization phase and the transfer phase. If the prover is honest, then w_i encrypts $a_i + \sum m_{j,i}u_j = a_i + m_{x_\ell,i}$ and z encrypts $\sum a_j v_j = a_{y_\ell}$, and thus the verifier can retrieve m_{x_ℓ,y_ℓ} as claimed.We show how to construct the SHVZK proofs/arguments in the following sections. All the SHVZK proofs/arguments should be compiled to general ones via a standard transformation by using commitments and a public coin flipping protocol, e.g. [9]. To keep the exposition simple, we will not explicitly mention the transformation in the protocol.

4.1 SHVZK Unit Vector Proof

Now we show how to construct the SHVZK Unit Vector Proof that is used in our $\mathsf{OT}_{k\times 1}^N$ scheme. In a Unit Vector Proof, given an encrypted vector $\boldsymbol{e} := (e_1, \ldots, e_n)^T = (\mathsf{Enc}_{pk}(b_1; r_1), \ldots, \mathsf{Enc}_{pk}(b_n; r_n))^T \in (\mathbb{G} \times \mathbb{G})^n$, the prover wants to convince the verifier that $\boldsymbol{b} := (b_1, \ldots, b_n)^T \in \mathbb{Z}_q^n$ is a unit vector, i.e., there is exactly one $i \in [n]$ such that $b_i = 1$ and $\forall j \neq i : b_j = 0$. Considering the lifted ElGamal encryption, we have $e_i := (e_{i,1}, e_{i,2}) = (g^{r_i}, g^{b_i} h^{r_i})$. As depicted in

Fig. 2, we give a ZK proof of knowledge (PoK) of $\boldsymbol{b}, \boldsymbol{r} \in \mathbb{Z}_q^n$ such that for $i \in [n]$, $e_i = \mathsf{Enc}_{pk}(b_i; r_i)$, $b_i \in \{0,1\}$ and $\sum_{i=1}^n b_i = 1$, where $\boldsymbol{r} := (r_1, \ldots, r_n)^T$. The proof uses $\vee, \wedge$ compositions of the basic Σ protocol to prove that e_i encrypts 0 or 1, based on DDH tuple proof technique [7].

Common input: Group information σ and the public key $pk := h$.
Prover's private input: $\boldsymbol{b} \in \{0,1\}^n$ and $\boldsymbol{r} \in \mathbb{Z}_q^n$.
Statement: $\{(e_{i,1}, e_{i,2}) := (g^{r_i}, g^{b_i} h^{r_i})\}_{i=1}^n$. Let $E_1 := \prod_{i=1}^n e_{i,1}$, $E_2 := \prod_{i=1}^n e_{i,2}$.

1. Since $b_i \in \{0,1\}$, let $\bar{b}_i := 1 - b_i$. For $i \in [n]$, the prover picks random $s_i, \rho_{i,\bar{b}_i}, z_{i,\bar{b}_i} \leftarrow_\$ \mathbb{Z}_q$ and computes $a_{i,1}^{(b_i)} = g^{s_i}, a_{i,2}^{(b_i)} = h^{s_i}, a_{i,1}^{(\bar{b}_i)} = g^{z_{i,\bar{b}_i}} e_{i,1}^{-\rho_{i,\bar{b}_i}}$ and $a_{i,2}^{(\bar{b}_i)} = h^{z_{i,\bar{b}_i}} \cdot (e_{i,2} \cdot g^{-\bar{b}_i})^{-\rho_{i,\bar{b}_i}}$. He picks random $s \leftarrow_\$ \mathbb{Z}_q$, sets $A_1 = g^s, A_2 = h^s$, and sends $\{(a_{i,1}^{(0)}, a_{i,2}^{(0)}), (a_{i,1}^{(1)}, a_{i,2}^{(1)})\}_{i=1}^n$ and (A_1, A_2) to the verifier.
2. The verifier picks random challenge $\rho \leftarrow_\$ \mathbb{Z}_q^*$ and sends ρ to the prover.
3. For $i \in [n]$, the prover sets $\rho_{i,b_i} = \rho - \rho_{i,\bar{b}_i}$ and $z_{i,b_i} = r_i \rho_{i,b_i} + s_i$. He computes $Z = \rho \cdot \sum_{i=1}^n r_i + s$, and sends $\{\rho_{i,0}, z_{i,0}, z_{i,1}\}_{i=0}^n$ and Z to the verifier.

Verification:

1. The verifier checks $E_1^\rho A_1 = g^Z \wedge (E_2/g)^\rho A_2 = h^Z$. For $i \in [n]$, the verifier computes $\rho_{i,1} = \rho - \rho_{i,0}$ and checks

$$e_{i,1}^{\rho_{i,0}} a_{i,1}^{(0)} = g^{z_{i,0}} \wedge e_{i,2}^{\rho_{i,0}} a_{i,2}^{(0)} = h^{z_{i,0}} \wedge e_{i,1}^{\rho_{i,1}} a_{i,1}^{(1)} = g^{z_{i,1}} \wedge (e_{i,2}/g)^{\rho_{i,1}} a_{i,1}^{(1)} = h^{z_{i,1}}.$$

Fig. 2. Public Coin SHVZK Unit Vector Proof

Theorem 1. *The protocol depicted in Fig. 2 is a* 3*-move public coin perfect special honest verifier zero-knowledge proof of knowledge of* $\boldsymbol{b}$ *and* $\boldsymbol{r}$ *such that* $e_i = \mathsf{Enc}_{pk}(b_i; r_i) \wedge b_i \in \{0,1\} \wedge \sum_{i=1}^n b_i = 1$.

Proof. For perfect completeness, if $b_i \in \{0,1\}$, it is easy to verify that all the equations hold. For soundness, we have to construct an extractor $\mathcal{X}$ that runs on $\langle \mathcal{P}^*, \mathcal{V} \rangle$ to get a transcript. It rewinds the protocol to the challenge phase and runs it with fresh challenges until it has 2 acceptable proofs. Assuming the prover $\mathcal{P}$ has probability of $p(\lambda)$ of making an acceptable proof, so the extractor $\mathcal{X}$ will take an average of $2/p(\lambda)$ rewinds, which is polynomial running time. Thus, there is overwhelming probability that we have transcripts with 2 different challenges $\rho^{(1)}, \rho^{(2)}$. From those transcripts, the extractor can extract the knowledge $\boldsymbol{b}$ and $\boldsymbol{r}$. Namely, for each $i \in [n]$, we have at least one different pair between $(\rho_{i,0}^{(1)}, \rho_{i,0}^{(2)})$ and $(\rho_{i,1}^{(1)}, \rho_{i,1}^{(2)})$. Assume $\rho_{i,x}^{(1)}, \rho_{i,x}^{(2)}$ are different, we can compute $r_i = (z_{i,x}^{(1)} - z_{i,x}^{(2)})/(\rho_{i,x}^{(1)} - \rho_{i,x}^{(2)})$. Subsequently, $\mathcal{X}$ can extract b_i by checking e_i. Hence, we have constructed an extractor $\mathcal{X}$ that outputs $\boldsymbol{b}$ and $\boldsymbol{r}$.

For perfect zero-knowledge, we construct a simulator $\mathcal{S}$ that on challenge ρ outputs simulated proof that is indistinguishable from a real proof with challenge ρ. On challenge ρ, for $i \in [n]$, $\mathcal{S}$ randomly picks $\rho_{i,0}, z_{i,0}, z_{i,1}, Z \leftarrow_\$ \mathbb{Z}_q$ and computes $\rho_{i,1} = \rho - \rho_{i,0}, A_1 = g^Z E_1^{-\rho}, A_2 = h^Z (E_2/g)^{-\rho}, a_{i,1}^{(0)} = g^{z_{i,0}} e_{i,1}^{-\rho_{i,0}}, a_{i,2}^{(0)} = h^{z_{i,0}} e_{i,2}^{-\rho_{i,0}}, a_{i,1}^{(1)} = g^{z_{i,1}} e_{i,1}^{-\rho_{i,1}}, a_{i,2}^{(1)} = h^{z_{i,1}} (e_{i,2}/g)^{-\rho_{i,1}}$. $\mathcal{S}$ outputs

$$\tau^* := \left(\{(a_{i,1}^{(0)}, a_{i,2}^{(0)}), (a_{i,1}^{(1)}, a_{i,2}^{(1)})\}_{i=1}^n, (A_1, A_2), \rho, \{\rho_{i,0}, z_{i,0}, z_{i,1}\}_{i=0}^n \right) .$$

Note that simulated $z_{i,0}, z_{i,1}, Z$ have the same distribution as in the real proof, because s_i, s are uniformly random. It is easy to see that ρ_0 and ρ_1 have identical distribution of them in a real proof. Finally, we argue that $\{(a_{i,1}^{(0)}, a_{i,2}^{(0)}), (a_{i,1}^{(1)}, a_{i,2}^{(1)})\}_{i=1}^n, (A_1, A_2)$ are uniquely determined for fixed $\rho_0, \rho_0, z_{i,0}, z_{i,1}, Z$. Therefore, we have shown that the distribution of simulated τ^* is identical to τ in a real proof. □

4.2 Multi-exponentiation Argument

In Fig. 3, we give an argument of knowledge of $\boldsymbol{m} := (m_1, \ldots, m_n)^T, \boldsymbol{r} := (r_1, \ldots, r_n)^T \in \mathbb{Z}_q^n$ and $t \in \mathbb{Z}_q$ such that $v = \mathsf{Enc}_{pk}(0; t) \prod_{j=1}^n e_j^{m_j}$ and $c_i = \mathsf{Com}_{ck}(m_i; r_i)$, for $i \in [n]$.

Common input: Group information σ and pk, ck.
Statement: $\boldsymbol{e} \in (\mathbb{G} \times \mathbb{G})^n, v \in (\mathbb{G} \times \mathbb{G})$ and $\boldsymbol{c} \in \mathbb{G}^n$.
Prover's private input: $\boldsymbol{m}, \boldsymbol{r} \in \mathbb{Z}_q^n$ and $t \in \mathbb{Z}_q$.

1. The prover picks $\boldsymbol{x}, \boldsymbol{y} \leftarrow_\$ \mathbb{Z}_q^n, t' \leftarrow_\$ \mathbb{Z}_q$ and sends $v' := \mathsf{Enc}_{pk}(0; t') \prod_{i=1}^n e_i^{x_i}$, $u_i := \mathsf{Com}_{ck}(x_i; y_i)$ to the verifier.
2. The verifier picks a random challenge $\rho \leftarrow_\$ \mathbb{Z}_q^*$ and sends ρ to the prover.
3. The prover sends $\boldsymbol{w} := \rho \cdot \boldsymbol{m} + \boldsymbol{x}$, $\hat{t} = \rho \cdot t + t'$ and $\boldsymbol{z} := \rho \cdot \boldsymbol{r} + \boldsymbol{y}$ to the verifier.

Verification:

1. The verifier checks $c_i^\rho u_i = \mathsf{Com}_{ck}(w_i; z_i) \wedge v^\rho v' = \mathsf{Enc}_{pk}(0; \hat{t}) \prod_{i=1}^n e_i^{w_i}$.

Fig. 3. Public Coin SHVZK Multi-exponentiation Argument

Theorem 2. *The protocol depicted in Fig. 3 is a* 3*-move public coin perfect special honest verifier zero-knowledge argument of knowledge of* $\boldsymbol{m}, \boldsymbol{r}, t$ *such that* $v = \mathsf{Enc}_{pk}(0; t) \prod_{j=1}^n e_j^{m_j} \wedge c_i = \mathsf{Com}_{ck}(m_i; r_i)$.

Proof. For perfect completeness, it is easy verify that all the equations hold. Now we prove soundness and show that the protocol is an argument of knowledge (AoK). Since $\rho \in \mathbb{Z}_q^*$ is randomly chosen, by Schwartz-Zippel lemma, the prover has negligible probability of convincing the verifier unless all ρ related terms match on each side of the equality. Now we construct the witness-extended emulator $\mathcal{X}$ runs $\langle \mathcal{P}^*, \mathcal{V} \rangle$ to get a transcript. If the prover $\mathcal{P}$ has probability $p(\lambda)$ of making an acceptable argument, the black-box witness-extended emulator $\mathcal{X}$ also has success probability $p(\lambda)$ to produce an accepting argument. It rewinds the protocol to the challenge phase and runs it with fresh challenges until it has 2 acceptable arguments. Since the prover $\mathcal{P}$ has probability $p(\lambda)$ of making an accepting argument in the first place, the emulator $\mathcal{X}$ will take an average of $2/p(\lambda)$ rewinds, which is polynomial running time. Again, there is overwhelming probability that we have transcripts with 2 different challenges $\rho^{(1)}, \rho^{(2)}$.

After obtaining $\boldsymbol{w}^{(\eta)} = \rho^{(\eta)} \cdot \boldsymbol{m} + \boldsymbol{x}$, $\hat{t}^{(\eta)} = \rho^{(\eta)} \cdot t + t'$, $\boldsymbol{z}^{(\eta)} = \rho^{(\eta)} \cdot \boldsymbol{r} + \boldsymbol{y}$ for $\eta \in \{1, 2\}$, $\mathcal{X}$ computes $\boldsymbol{m} = (\boldsymbol{w}^{(1)} - \boldsymbol{w}^{(2)})/(\rho^{(1)} - \rho^{(2)})$, $t = (\hat{t}^{(1)} - \hat{t}^{(2)})/(\rho^{(1)} - \rho^{(2)})$ and $\boldsymbol{r} = (\boldsymbol{z}^{(1)} - \boldsymbol{z}^{(2)})/(\rho^{(1)} - \rho^{(2)})$. Hence, we have extracted a valid witness $\boldsymbol{m}, r, t$ for the statement.

For perfect zero-knowledge, we have to construct a simulator $\mathcal{S}$ on challenge ρ outputs the simulated argument that is indistinguishable from a real argument with challenge ρ. On challenge ρ, the simulator $\mathcal{S}$ randomly picks $\boldsymbol{w}, \boldsymbol{z} \leftarrow_{\$} \mathbb{Z}_q^n$ and $\hat{t} \leftarrow_{\$} \mathbb{Z}_q$. $\mathcal{S}$ computes $v' = \mathsf{Enc}_{pk}(0; \hat{t}) \prod_{i=1}^{n} e_i^{w_i} \cdot v^{-\rho}$ and $u_i = \mathsf{Com}_{ck}(w_i; z_i) \cdot c_i^{-\rho}$. $\mathcal{S}$ outputs $\tau^* := (v', \boldsymbol{u}, \rho, \boldsymbol{w}, \hat{t}, \boldsymbol{z})$. Since $\boldsymbol{x}, \boldsymbol{y}, t$ are uniformly random in a real argument, the distribution of simulated $\boldsymbol{w}, \hat{t}, \boldsymbol{z}$ is identical to the distribution of them in a real argument. Furthermore, $v', \boldsymbol{u}$ are uniquely determined for fixed $\rho, \boldsymbol{w}, \hat{t}, \boldsymbol{z}$; therefore, simulated τ^* has the same distribution as τ in a real argument. □

4.3 Masked Multi-exponentiation Batch Argument

In this section, we propose the masked multi-exponentiation batch argument. Given two vectors of ciphertexts $\boldsymbol{e} := (e_1, \ldots, e_n)^T \in (\mathbb{G} \times \mathbb{G})^n$, $\boldsymbol{v} := (v_1, \ldots, v_\ell)^T \in (\mathbb{G} \times \mathbb{G})^\ell$ and two vectors of commitments $\boldsymbol{c} := (c_1, \ldots, c_\ell)^T \in \mathbb{G}^\ell$ and $\boldsymbol{u} := (u_1, \ldots, u_\ell)^T \in \mathbb{G}^\ell$, as depicted in Fig. 4, we will give an argument of knowledge of $\boldsymbol{M} := \{m_{j,i}\}_{j,i=1}^{n,\ell} \in \mathbb{Z}_q^{n \times \ell}$, $\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{d} \in \mathbb{Z}_q^\ell$ such that for $i \in [\ell]$,

$$v_i = \mathsf{Enc}_{pk}(s_i; t_i) \prod_{j=1}^{n} e_j^{m_{j,i}} \quad , \quad u_i = \mathsf{Com}_{ck}(s_i; d_i) \quad \text{and} \quad c_i = \mathsf{Com}_{ck}(\boldsymbol{m}_i; r_i)$$

where $\boldsymbol{m}_i := (m_{1,i}, \ldots, m_{n,i})^T$, $\boldsymbol{r} := (r_1, \ldots, r_\ell)^T$, $\boldsymbol{s} := (s_1, \ldots, s_\ell)^T$, $\boldsymbol{t} := (t_1, \ldots, t_\ell)^T$ and $\boldsymbol{d} := (d_1, \ldots, d_\ell)^T$.

Theorem 3. *The protocol depicted in Fig. 4 is a 3-move public coin perfect special honest verifier zero-knowledge argument of knowledge of $\boldsymbol{M}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{d}$ such that for $i \in [\ell]$,*

$$v_i = \mathsf{Enc}_{pk}(s_i; t_i) \prod_{j=1}^{n} e_j^{m_{j,i}} \quad , \quad u_i = \mathsf{Com}_{ck}(s_i; d_i) \quad \text{and} \quad c_i = \mathsf{Com}_{ck}(\boldsymbol{m}_i; r_i) \ .$$

Proof. For perfect completeness, it is easy to verify that all the equations hold. Now we prove soundness and show that the protocol is an argument of knowledge (AoK), by showing that it has a witness-extended emulator. Since $\rho \in \mathbb{Z}_q^*$ is randomly chosen, by Schwartz-Zippel lemma, the prover has negligible probability of convincing the verifier unless all ρ^i related terms match on each side of the equality for all $i \in [\ell]$. The witness-extended emulator $\mathcal{X}$ runs $\langle \mathcal{P}^*, \mathcal{V} \rangle$ to get a transcript. If the prover $\mathcal{P}$ has probability $p(\lambda)$ of making an acceptable argument, the black-box witness-extended emulator $\mathcal{X}$ also has success probability $p(\lambda)$ to produce an accepting argument. It rewinds the protocol to the challenge phase and runs it with fresh challenges until it has $\ell + 1$ acceptable arguments.

Common input: Group information σ and pk, ck.
Statement: $\boldsymbol{e} \in (\mathbb{G} \times \mathbb{G})^n, \boldsymbol{v} \in (\mathbb{G} \times \mathbb{G})^\ell$ and $\boldsymbol{c}, \boldsymbol{u} \in \mathbb{G}^\ell$.
Prover's private input: $\boldsymbol{M} \in \mathbb{Z}_q^{n \times \ell}$ and $\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{d} \in \mathbb{Z}_q^\ell$.

1. The prover picks $\boldsymbol{x} \leftarrow_\$ \mathbb{Z}_q^n, s_0, t_0, r_0, d_0 \leftarrow_\$ \mathbb{Z}_q$ and sends $v_0 := \mathsf{Enc}_{pk}(s_0; t_0) \prod_{i=1}^n e_i^{x_i}$, $c_x := \mathsf{Com}_{ck}(\boldsymbol{x}; r_0)$ and $u_0 := \mathsf{Com}_{ck}(s_0; d_0)$ to the verifier.
2. The verifier randomly picks a challenge $\rho \leftarrow_\$ \mathbb{Z}_q^*$ and sends ρ to the prover.
3. Set $\boldsymbol{\rho} := (\rho, \rho^2, \ldots, \rho^\ell)^T$. The prover sends $\boldsymbol{w} := \boldsymbol{M} \cdot \boldsymbol{\rho} + \boldsymbol{x}$, $t' = \boldsymbol{t}^T \cdot \boldsymbol{\rho} + t_0$, $s' = \boldsymbol{s}^T \cdot \boldsymbol{\rho} + s_0$, $d' = \boldsymbol{d}^T \cdot \boldsymbol{\rho} + d_0$ and $r' := \boldsymbol{r}^T \cdot \boldsymbol{\rho} + r_0$ to the verifier, where $\boldsymbol{w} := (w_1, \ldots, w_n)^T$.

Verification:

1. The verifier checks

$$u_0 \prod_{i=1}^{\ell} u_i^{\rho^i} = \mathsf{Com}_{ck}(s'; d') \wedge c_x \prod_{i=1}^{\ell} c_i^{\rho^i} = \mathsf{Com}_{ck}(\boldsymbol{w}; r') \wedge v_0 \prod_{i=1}^{\ell} v_i^{\rho^i} = \mathsf{Enc}_{pk}(s'; t') \prod_{i=1}^{n} e_i^{w_i}.$$

Fig. 4. Public Coin SHVZK Masked Multi-exponentiation Batch Argument

Since the prover $\mathcal{P}$ has probability $p(\lambda)$ of making an accepting argument in the first place, the emulator $\mathcal{X}$ will take an average of $\frac{\ell+1}{p(\lambda)}$ rewinds, which takes $\mathrm{poly}(\lambda)$ running time. Again, there is overwhelming probability that we have transcripts with $\ell + 1$ different challenges. The $\ell + 1$ different challenges give us a $(\ell + 1) \times (\ell + 1)$ transposed Vandermonde matrix

$$\boldsymbol{V} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \rho^{(1)} & \rho^{(2)} & \cdots & \rho^{(\ell+1)} \\ \vdots & \vdots & \ddots & \vdots \\ (\rho^{(1)})^\ell & (\rho^{(2)})^\ell & \cdots & (\rho^{(\ell+1)})^\ell \end{pmatrix}.$$

Note that $\boldsymbol{V}$ is invertible because $\rho^{(1)}, \ldots, \rho^{(\ell+1)}$ are different, and $\mathcal{X}$ computes $\boldsymbol{V}^{-1}$. Let $\boldsymbol{M}_{\boldsymbol{x}}$ be the $n \times (\ell + 1)$ matrix that is the column $\boldsymbol{x}$ concatenated at the left side of $\boldsymbol{M}$ and denote $\boldsymbol{W}$ as the matrix that consists of columns $(\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(\ell+1)})$. We have $\boldsymbol{W} = \boldsymbol{M}_{\boldsymbol{x}} \cdot \boldsymbol{V}$, and $\mathcal{X}$ can compute $\boldsymbol{M}_{\boldsymbol{x}} = \boldsymbol{W} \cdot \boldsymbol{V}^{-1}$. Similarly, $\mathcal{X}$ can extract $\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{d}$; hence, $\mathcal{X}$ has extracted a valid witness $\boldsymbol{M}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{d}$ for the statement.

For perfect zero-knowledge, we have to construct a simulator $\mathcal{S}$ on challenge ρ outputs the simulated argument that is indistinguishable from a real argument with challenge ρ. On challenge ρ, the simulator $\mathcal{S}$ randomly picks $\boldsymbol{w} \leftarrow_\$ \mathbb{Z}_q^n$ and $r', s', t', d' \leftarrow_\$ \mathbb{Z}_q$. $\mathcal{S}$ computes $u_0 = \mathsf{Com}_{ck}(s'; d')/(\prod_{i=1}^{\ell} u_i^{\rho^i})$, $v_0 = (\mathsf{Enc}_{pk}(s'; t') \prod_{i=1}^{n} e_i^{w_i})/(\prod_{i=1}^{\ell} v_i^{\rho^i})$ and $c_x = \mathsf{Com}_{ck}(\boldsymbol{w}; r')/(\prod_{i=1}^{\ell} c_i^{\rho^i})$. $\mathcal{S}$ outputs $\tau^* := (u_0, v_0, c_x, \rho, \boldsymbol{w}, r', s', t', d')$. Since $\boldsymbol{x}, r_0, s_0, t_0, d_0$ are uniformly random in a real argument, the distribution of simulated $\boldsymbol{w}, r', s', t', d'$ is identical to the distribution of them in a real argument. Furthermore, u_0, v_0, c_x are uniquely determined for fixed $\rho, \boldsymbol{w}, r', s', t', d'$, therefore, simulated τ^* is identical to the distribution of τ in a real argument. □

4.4 Security Analysis of Our $\mathsf{OT}^N_{k\times 1}$ Scheme

In this section, we examine the security of our $\mathsf{OT}^N_{k\times 1}$ scheme in Fig. 1. Since w_i in step 3 of the transfer phase is masked by a_i, it does not reveal information about $\boldsymbol{M}$; therefore, the receiver can only decrypt one document in each transfer phase. In our security proof of fully simulation, we don't consider the initialization phase and transfer phase as separated experiments. One may add argument of knowledge of the openings of commitment $\boldsymbol{c}$ [15] in the initialization phase in order to exact the sender's input $\boldsymbol{M}$ in the initialization phase. Note that it is the receiver's responsibility to choose correct commitment key ck to achieve the binding property. Since, the order of $\mathbb{G}$ is q, the sender only needs to check group membership of ck to guarantee that his commitments will not reveal anything information about the messages even if the receiver is cheating. Its formal security proof is given in App. A.

4.5 Implementation and Efficiency

In terms of communicational efficiency, it is clear that the proposed $\mathsf{OT}^N_{k\times 1}$ scheme (shown in Fig. 1) costs $\mathcal{O}(\sqrt{N})$ in both the initialization phase and each transfer phase. Let $k = 1$, as far as we know, our proposed OT^N_1 is the first fully simulatable OT^N_1 that achieves $\mathcal{O}(\sqrt{N})$ communication complexity. The computation complexity of our proposed $\mathsf{OT}^N_{k\times 1}$ scheme is $\mathcal{O}(N)$ in both initialization phase and each transfer phase. As mentioned before, since the scheme uses lifted ElGamal encryption, the message space should be small enough to compute discrete logarithm, e.g., $m_i \in \{0,1\}^{\xi}$, where $\xi \leq 30$.

In practical implementation, the actual complexity of our protocol is smaller. Since the protocol only uses multi-exponentiation operations in both homomorphic operations and commitments. We employ Lim's multi-exponentiation algorithm to reduce the actual computation. In [25], Lim showed how to compute a product of n exponentiations using only $\mathcal{O}(\frac{n}{\log n})$ multiplications. We implemented the proposed $\mathsf{OT}^N_{k\times 1}$ scheme on elliptic curve group over $\mathbb{F}_p$. The performance benchmark is tested with the 192-bit elliptic curve domain parameters recommended by NIST p192, where $p = 2^{192} - 2^{64} - 1$, which gives about 96-bit security level. In order to save communication bandwidth, we also used the standard point compression technique: a point on $E(\mathbb{F}_p)$ is represented by its x coordinate together with the least significant bit of its y coordinate. The code is implemented in C++, using *Multi-precision Integer and Rational Arithmetic C/C++ Library* (MIRACL) crypto SDK. All the tests are performed on a linux desktop with an Intel Core *i*5-2400 CPU running at 3.10 GHz. Table 2 depicts the sender's and receiver's running time (in seconds) as well as the communication complexity (in bytes) in both initialization phase and each transfer phase. We can see our scheme is very efficient even with relatively large database size.

Table 2. Performance Benchmark. (r.t. stands for running time. Messages are chosen from $\{0,1\}^{10}$, and the network delay is not considered.)

	Initialization phase			Each transfer phase		
DB size	**S**'s r.t. (s)	**R**'s r.t. (s)	Comm. (byte)	**S**'s r.t. (s)	**R**'s r.t. (s)	Comm. (byte)
1×10^4	0.06	0.045	4065	0.98	1.16	44320
2.5×10^5	0.29	0.565	20165	4.9	7.24	220320
1×10^6	0.59	1.975	40290	9.7	17.68	440320
2.5×10^7	2.92	43.555	201290	48.61	223.25	2200320
1×10^8	5.83	171.24	402540	96.94	786.77	4400320

5 Conclusions

In this paper, we proposed an efficient $\mathsf{OT}^N_{k\times 1}$ scheme in the plain model. It achieves fully simulatable security with $\mathcal{O}(\sqrt{N})$ communication in both the initialization phase and each transfer phase. Ideally, the scheme is dedicated to 1-out-of-N oblivious transfer, whereas it also achieves better (amortized) communication, comparing with existing schemes when $k = \mathcal{O}(N^{1/2})$, which covers majority OT usage cases. We also implemented and highly optimized the proposed scheme, and its perform benchmark shows very impressive results. When k is very large, say $\mathcal{O}(N)$, we recommend the user to adopt ORAM based two-party computation schemes, e.g. [11], so the cost of each transfer is minimum after the setup phase. We would like to further reduce the communication complexity of fully simulatable OT^N_1 in our future research.

Acknowledgements. The second author was supported by Estonian Research Council, the Tiger University Program of the Estonian Information Technology Foundation, and European Union through the European Regional Development Fund. The last author was supported in part by US National Science Foundation under grants CNS-1262277 and CNS-1116939.

References

1. Groth, J., Kiayias, A., Lipmaa, H.: Multi-query Computationally-Private Information Retrieval with Constant Communication Rate. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 107–123. Springer, Heidelberg (2010)
2. Bayer, S., Groth, J.: Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012)
3. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
4. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Short PCPs Verifiable in Polylogarithmic Time. In: CCC (2005)
5. Camenisch, J., Neven, G., Shelat, A.: Simulatable Adaptive Oblivious Transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)

6. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
7. Chaum, D.: Zero-Knowledge Undeniable Signatures (extended abstract). In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 458–464. Springer, Heidelberg (1991)
8. Damgård, I., Fujisaki, E.: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
9. Damgård, I., Goldreich, O., Okamoto, T., Wigderson, A.: Honest Verifier vs Dishonest Verifier in Public Coin Zero-Knowledge Proofs. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 325–338. Springer, Heidelberg (1995)
10. 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)
11. Gordon, S.D., Katz, J., Kolesnikov, V., Krell, F., Malkin, T., Raykova, M., Vahlis, Y.: Secure Two-party Computation in Sublinear (amortized) Time. In: CCS (2012)
12. Green, M., Hohenberger, S.: Blind Identity-Based Encryption and Simulatable Oblivious Transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
13. Green, M., Hohenberger, S.: Universally Composable Adaptive Oblivious Transfer. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 179–197. Springer, Heidelberg (2008)
14. Green, M., Hohenberger, S.: Practical Adaptive Oblivious Transfer from Simple Assumptions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 347–363. Springer, Heidelberg (2011)
15. Groth, J.: Linear Algebra with Sub-linear Zero-knowledge Arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 192–208. Springer, Heidelberg (2009)
16. Groth, J.: A Verifiable Secret Shuffle of Homomorphic Encryptions. Journal of Cryptology 23, 546–579 (2010)
17. Groth, J.: hort Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
18. Ishai, Y., Paskin, A.: Evaluating Branching Programs on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
19. Jarecki, S., Liu, X.: Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)
20. Kurosawa, K., Nojima, R.: Simple Adaptive Oblivious Transfer Without Random Oracle. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 334–346. Springer, Heidelberg (2009)
21. Kurosawa, K., Nojima, R., Phong, L.T.: Efficiency-improved fully simulatable adaptive OT under the DDH assumption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 172–181. Springer, Heidelberg (2010)
22. Kurosawa, K., Nojima, R., Phong, L.T.: Generic Fully Simulatable Adaptive Oblivious Transfer. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 274–291. Springer, Heidelberg (2011)
23. Kushilevitz, E., Ostrovsky, R.: Replication is NOT Needed: Single Database, Computationally-Private Information Retrieval. In: FOCS (1997)

24. Laur, S., Lipmaa, H.: On the Feasibility of Consistent Computations. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 88–106. Springer, Heidelberg (2010)
25. Lim, C.H.: Efficient Multi-exponentiation and Application to Batch Verification of Digital Signatures (2000), online Tech. Report: `http://dasan.sejong.ac.kr/~chlim/pub/multiexp.ps`
26. 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)
27. Lipmaa, H.: First CPIR Protocol with Data-Dependent Computation. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 193–210. Springer, Heidelberg (2010)
28. Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (2012)
29. Liskova, L., Stanek, M.: Efficient Simultaneous Contract Signing. In: Deswarte, Y., Cuppens, F., Jajodia, S., Wang, L. (eds.) Security and Protection in Information Processing Systems. IFIP, vol. 147, pp. 440–455. Springer, Boston (2004)
30. Naor, M., Pinkas, B.: Oblivious Transfer with Adaptive Queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
31. Naor, M., Pinkas, B.: Computationally Secure Oblivious Transfer. Journal of Cryptology 18, 1–35 (2005), `http://dx.doi.org/10.1007/s00145-004-0102-6`
32. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
33. Rial, A., Kohlweiss, M., Preneel, B.: Universally Composable Adaptive Priced Oblivious Transfer. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 231–247. Springer, Heidelberg (2009)
34. Yao, A.: Protocols for Secure Computations (Extended Abstract). In: FOCS (1982)
35. Zhang, B.: Simulatable Adaptive Oblivious Transfer with Statistical Receiver's Privacy. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 52–67. Springer, Heidelberg (2011)

A Security Proof of Our $\mathsf{OT}^N_{k\times 1}$ Scheme

Theorem 4. *The proposed $\mathsf{OT}^N_{k\times 1}$ scheme (as shown in Fig. 1) is secure against the sender corruption under the DDH assumption.*

Proof. We show that for every real-world cheating p.p.t. sender $\hat{\mathbf{S}}$ there exists an ideal-world cheating p.p.t. sender $\hat{\mathbf{S}}'$ such that for every distinguisher $\mathbf{D}$:

$$\mathbf{Real}_{\hat{\mathbf{S}},\mathbf{R}}(N,k,m_1,\ldots,m_N,\mathcal{I}) \quad \overset{c}{\approx} \quad \mathbf{Ideal}_{\hat{\mathbf{S}}',\mathbf{R}'}(N,k,m_1,\ldots,m_N,\mathcal{I})$$

Considering a sequence of games $G_0,\ldots,G_4$, where Game $G_0 = \mathbf{Real}_{\hat{\mathbf{S}},\mathbf{R}}$ and Game $G_4 = \mathbf{Ideal}_{\hat{\mathbf{S}}',\mathbf{R}'}$. We define

$$\mathsf{Adv}[\mathbf{D}] = \left|\Pr[\mathbf{D}(X)=1 : X \overset{\$}{\leftarrow} \mathbf{Ideal}_{\hat{\mathbf{S}}',\mathbf{R}'}] - \Pr[\mathbf{D}(X)=1 : X \overset{\$}{\leftarrow} \mathbf{Real}_{\hat{\mathbf{S}},\mathbf{R}}]\right| \ .$$

Game G_0: The real-world experiment $\mathbf{Real}_{\hat{\mathbf{S}},\mathbf{R}}$. By definition, $\Pr[\mathbf{D}(X)=1 : X \overset{\$}{\leftarrow} G_0] = Pr[\mathbf{D}(X)=1 : X \overset{\$}{\leftarrow} \mathbf{Real}_{\hat{\mathbf{S}},\mathbf{R}}]$.

Game G_1: Game G_1 is the same as Game G_0 except the following. In the first transfer phase, the receiver uses the witness-extended emulator of the *masked multi-exponentiation batch AoK* to extract $\boldsymbol{M^*}, \boldsymbol{r^*}$ that is committed in $\boldsymbol{c}$. If extraction fails, then the protocol aborts. The failure probability is negligible. Furthermore, if the server can open the commitments to a different set $\boldsymbol{M'}, \boldsymbol{r'}$ from what is extracted, then we have broken the blinding property of generalized Pedersen Commitment; namely, the discrete logarithm assumption does not hold, neither does the DDH assumption. Assume the DDH problem is hard over $\mathbb{G}$, we have $\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_1] \approx \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_0]$.

Game G_2: Game G_2 is the same as Game G_1 except the following. In the initialization phase, the receiver randomly picks pk such that he does not know the discrete logarithm of pk. In the ℓ-th transfer phase, the receiver skips all the decryption steps, and returns $m^*_{i_\ell}$ according $\boldsymbol{M^*}$ that is extracted in Game G_1. Since all the zero-knowledge arguments and proofs are sound, we have $\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_2] \approx \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_1]$.

Game G_3: Game G_3 is the same as Game G_2 except the following. In the ℓ-th transfer phase, the receiver picks two random unit vectors $\boldsymbol{u}, \boldsymbol{v}$, regardless i_ℓ. Since ElGamal encryption is IND-CPA secure under the DDH assumption, we have $\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_3] \approx \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_2]$.

Game G_4: The ideal-world experiment $\mathbf{Ideal}_{\hat{\mathbf{S}}', \mathbf{R}'}$ in which an ideal-world sender $\hat{\mathbf{S}}'$ uses the real-world sender $\hat{\mathbf{S}}$ as a black-box as follows.

1. After receiving $(m_1, \ldots, m_N)$, $\hat{\mathbf{S}}'$ forwards them to $\hat{\mathbf{S}}$.
2. $\hat{\mathbf{S}}'$ acts as the receiver and plays Game G_3 with $\hat{\mathbf{S}}$.
3. In the first transfer phase, $\hat{\mathbf{S}}'$ sends $(m^*_1, \ldots, m^*_N)$ that is extracted in Game G_1 to $\mathcal{F}_{OT}^{n\times 1}$ (for the initialization phase). [2]
4. In the ℓ-th transfer phase, if $\hat{\mathbf{S}}$ behaved in an acceptable way, then $\hat{\mathbf{S}}'$ sends $b_\ell = 1$ to $\mathcal{F}_{OT}^{n\times 1}$. Otherwise, $\hat{\mathbf{S}}'$ sends $b_\ell = 0$ to $\mathcal{F}_{OT}^{n\times 1}$.

To sum up, it is easy to see that

$$\mathsf{Adv}(\mathbf{D}) = \left|\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_4] - \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_0]\right| \leq \epsilon(\lambda) \ ,$$

where $\epsilon(\cdot)$ is a negligible function. □

Theorem 5. *The proposed $\mathsf{OT}^N_{k\times 1}$ scheme (as shown in Fig. 1) is statistically secure against the receiver corruption.*

Proof. We show that for every real-world cheating p.p.t. receiver $\hat{\mathbf{R}}$ there exists an ideal-world cheating p.p.t. receiver $\hat{\mathbf{R}}'$ such that for every distinguisher $\mathbf{D}$:

$$\mathbf{Real}_{\mathbf{S},\hat{\mathbf{R}}}(N, k, m_1, \ldots, m_N, \mathcal{I}) \quad \overset{c}{\approx} \quad \mathbf{Ideal}_{\mathbf{S}',\hat{\mathbf{R}}'}(N, k, m_1, \ldots, m_N, \mathcal{I})$$

[2] Remark: the experiments do not separate initialization phase and transfer phase.

Again, we consider a series of hybrid games $G_0, \ldots, G_4$, where Game $G_0 = \mathbf{Real}_{\mathbf{S},\hat{\mathbf{R}}}$ and Game $G_4 = \mathbf{Ideal}_{\mathbf{S}',\hat{\mathbf{R}}'}$. We define

$$\mathsf{Adv}[\mathbf{D}] = \left|\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} \mathbf{Ideal}_{\mathbf{S}',\hat{\mathbf{R}}'}] - \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} \mathbf{Real}_{\mathbf{S},\hat{\mathbf{R}}}]\right| \ .$$

Game G_0: The real-world experiment $\mathbf{Real}_{\mathbf{S},\hat{\mathbf{R}}}$. By definition, $\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_0] = \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} \mathbf{Real}_{\mathbf{S},\hat{\mathbf{R}}}]$.

Game G_1: Game G_1 is the same as Game G_0 except the following. In the ℓ-th transfer phase, the sender uses the knowledge extractor of *unit vector PoK* to extract the plaintext and randomizers of each ciphertext, i.e., those two unit vectors $\boldsymbol{u}^*, \boldsymbol{v}^*$. Subsequently, we know the index i_ℓ^*. If extraction fails, then the protocol aborts. Since the failure probability is negligible, $\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_1] \approx \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_0]$.

Game G_2: Game G_2 is the same as Game G_1 except the following. In each transfer phase, the sender uses the simulator of the *masked multi-exponentiation batch AoK* to prove that $\boldsymbol{w}$ is computed correctly without using $\boldsymbol{M}$. If simulation fails, then the protocol aborts. Since the failure probability is negligible, we have $\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_2] \approx \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_1]$.

Game G_3: Game G_3 is the same as Game G_2 except the following. In the initialization phase, the sender randomly picks $\boldsymbol{\alpha} \leftarrow_\$ \mathbb{Z}_q^n$ and sets $c_i = g^{\alpha_i}$ as fail commitments. Since the distribution of $\boldsymbol{c}$ is unchanged, $\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_3] = \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_2]$.

Game G_4: The ideal-world experiment $\mathbf{Ideal}_{\mathbf{S}',\hat{\mathbf{R}}'}$in which an ideal-world receiver $\hat{\mathbf{R}}'$ uses the real-world receiver $\hat{\mathbf{R}}$ as a black-box as follows.

1. $\hat{\mathbf{R}}'$ acts as the sender and plays Game G_3 with $\hat{\mathbf{R}}$.
2. In the ℓ-th transfer phase, $\hat{\mathbf{R}}'$ sends i_ℓ^* that is extracted in Game G_1 to $\mathcal{F}_{OT}^{n\times 1}$ and fetches $m_{i_\ell^*}$ from $\mathcal{F}_{OT}^{n\times 1}$. $\hat{\mathbf{R}}'$ prepares $\boldsymbol{M'}$ such that $m'_{i_\ell^*} = m_{i_\ell^*}$ and $\forall j \neq i_\ell^* : m'_j = 0$.
3. Compute $\boldsymbol{w}$ according to $\boldsymbol{M'}$ and complete the rest of the protocol as described in Game G_3.

To sum up, it is easy to see that

$$\mathsf{Adv}(\mathbf{D}) = \left|\Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_4] - \Pr[\mathbf{D}(X) = 1 : X \xleftarrow{\$} G_0]\right| \leq \epsilon(\lambda) \ ,$$

where $\epsilon(\cdot)$ is a negligible function. □

Theorem 6. *The proposed $\mathsf{OT}_{k\times 1}^N$ scheme (as shown in Fig. 1) is fully simulatable secure under the DDH assumption.*

Proof. By Definition 1, the proposed $\mathsf{OT}_{k\times 1}^N$ framework is fully simulatable secure due to both Theorem. 4 and Theorem. 5.

Unconditionally-Secure Robust Secret Sharing with Minimum Share Size

Mahabir Prasad Jhanwar and Reihaneh Safavi-Naini

Department of Computer Science
University of Calgary, Canada

Abstract. An n-player (t, δ)-secure threshold robust secret sharing scheme is a (t, n)-threshold secret sharing scheme with the additional property that the secret can be recovered, with probability at least $1-\delta$, from the set of *all* shares even if up to t players provide incorrect shares. The existing constructions of threshold robust secret sharing schemes for the range $n/3 \leq t < n/2$ have the share size larger than the secret size. An important goal in this area is to minimize the share size. In the paper, we propose a new unconditionally-secure threshold robust secret sharing scheme for the case $n \geq 2t + 2$ with share size equal to the secret size. This is the minimum possible size as dictated by the perfect secrecy of the scheme.

Keywords: Shamir secret sharing, robust secret sharing, secret sharing with cheating detection.

1 Introduction

Secret Sharing is one of the most important tools in modern cryptography. The concept and the first realization of secret sharing were presented independently in [24] and in [3]. In a secret sharing scheme, there exists a dealer, n participants, and possibly a reconstructor. The dealer splits a secret $s \in S$, into n pieces, called shares, and sends one share to each participant over a private point-to-point channel. An access structure is the set of subsets of participants that are qualified to recover the secret. In a (t, n)-threshold access structure, where $1 \leq t < n$, any $t + 1$ or more participants can reconstruct the secret, and the knowledge of t or less shares leaves the secret s indeterminate. A (t, n)-threshold secret sharing scheme is said to be *perfect* if no subset of t or less shares can leak any information about the secret s where the leakage is in information theoretic sense and without assuming any limit on the computational resources of the adversary.

In its basic form, secret sharing assumes that the corrupted participants are passive (or semi-honest) and follow the protocol during the reconstruction phase. Extensions of this basic model considers cases that the corrupted participants deviate from the protocol [19,27,22,4,6,20]. In these extensions different requirements such as cheater detection [4], cheater identification [20] and untrusted

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 96–110, 2013.

dealer [8] have been considered. A minimal robust requirement when participants are allowed to submit incorrect shares, is that the set of all shares, some possibly corrupted, can recover the correct secret. Perfect secret sharing schemes that satisfy this additional property are called *robust secret sharing schemes.*

Threshold robust secret sharing schemes provide a powerful tool for building secure and reliable distributed data storage systems. Users' data (files) can be broken into pieces (shares) and stored on multiple servers such that privacy of data against servers is provided, and the system ensures recovery of the data when a subset of servers corrupt their stored shares, accidentally or intentionally. In recent years, systems and architectures based on this primitive have emerged [16,28,11] which shows importance of robust secret sharing in practice. Threshold robust secret sharing has also direct application to Secure Message Transmission (SMT). In an unconditionally secure SMT [10,12,13], a sender is connected to a receiver through n wires such that up to t of which are controlled by an adversary. The goal of an SMT protocol is to ensure that the message sent by the sender is received correctly by the receiver, and no information about the message is leaked to the adversary. Good threshold robust secret sharing schemes lead to good secure message transmission schemes [18]. Robust secret sharing schemes may also be seen as an stepping stone towards the construction of verifiable secret sharing (VSS) schemes [8], in which, in addition to the corrupted players, the dealer is dishonest and may hand out inconsistent shares. Finally robust secret sharing is an important primitive for secure multi-party computation.

1.1 Motivation

In perfect threshold robust secret sharing schemes, in addition to the requirement of perfect threshold secret sharing schemes,

- any $t+1$ shares reconstruct the secret, and any t shares give no information about the secret,

it is also required that,

- the secret can be reconstructed with high probability from the set of all shares, even if up to t shares are incorrect.

The reconstruction may be with, or without, a reconstructor, and may include one or more rounds of communication [9,7]. Also, reconstruction failure may be defined differently [9,7]. These variations of the model needs careful considerations in comparing schemes and their performances. A framework for considering robust secret sharing is provided in [23], which includes schemes in both the information-theoretic and computationally secure settings.

In this paper we shall follow the model of [7], where during the reconstruction, all the n players communicate their shares to a trusted third party called the reconstructor. Based on the received shares (some of them are incorrect), the reconstructor then produces an output s', which with a probability at least $1-\delta$ is the same as the original secret s.

Important efficiency measures for robust secret sharing schemes are the communication cost (number of communicated bits) of reconstruction [9] and the share size measured by the number of bits required to represent a share. In this paper we focus on the latter measure. It is well known that in any perfect secret sharing scheme, the length of a share σ_i is at least the length of the secret, that is $\log |S|$ [25]. Secret sharing schemes that meet this lower bound are called ideal [25]. Shamir secret sharing meets this lower bound and is ideal. This lower bound also holds for robust secret sharing scheme which are perfect secret sharing schemes. So a natural question is how much *redundancy*, that is extra share length compared to the secret length, is needed for robustness.

It follows from the theory of Reed-Solomon error correcting codes that Shamir secret sharing scheme is robust if (and only if) $t < n/3$ [1] and so no increase in the share size is needed to obtain robustness. On the other hand, the robust secret sharing is impossible if $t \geq n/2$: the (t, n)-threshold access structure requires at least $(t+1)$ correct shares for the recovery of the original secret. The interesting range is $n/3 \leq t < n/2$. In this range, all existing schemes have share sizes that are strictly larger than the secret size. The problem is naturally more difficult when t is maximal in the range $n/3 \leq t < n/2$ i.e., $n = 2t + 1$ (when n is odd) and $n = 2t + 2$ (when n is even). In particular, for the range $n/3 \leq t < n/2$, there is no known threshold robust secret sharing scheme such that,

- the maximum length of individual share size of a participant is the *same* as the secret size (thus no increase in the share size), and the *probability of correctly recovering the secret from the set of all shares* is at least $1 - \delta$, where δ is a negligible value.

This is irrespective of the computational complexity of reconstruction which can be exponential in n. The result of this paper shows that it is possible to keep the share size same as the secret size when $n \geq 2t + 2$.

1.2 Our Contribution

We consider the model of [7] in which reconstruction is by a trusted reconstructor, and propose a new construction of robust secret sharing which is based on Shamir's secret sharing and has share size equal to the secret size, which is the minimum required by perfect secret sharing schemes. That is, the extra robustness property is obtained without increasing the share size. The system's public parameter, in addition to what is required by Shamir's scheme, includes a structured matrix that has $O(n)$ random elements. This matrix can be distributed during share distribution, or stored on an authenticated and publicly accessible storage. The system works for $n \geq 2t + 2$ and effectively uses the share of the extra participants as the verification information. We note that if n is even, then $n = 2t + 2$ is the minimum required number of participants.

The reconstruction is one round and requires participants to send their shares to the reconstructor. The reconstruction is secure against a non-rushing adversary (this is properly defined in Sect. 2.2), and the reconstruction procedure may output an incorrect secret with a negligible probability. The reconstruction

algorithm however is inefficient and requires that all subsets of size $t+1$ of n shares be considered. Construction of schemes with the above properties and efficient reconstruction, remains an open problem.

1.3 Related Work

Cheating detection and providing robustness against cheaters, is an important problem in secret sharing schemes. Different models and constructions have been proposed for this problem over the year [22,4,6,19,27,20]. Robustness in the sense of recoverability of the secret when some shares are wrong is a basic property that ensures the secret is not lost because of the share corruption. Robust secret sharing with unconditional security was first considered by McEliece and Sarwate [19] where they pointed out the close relationship between Shamir secret sharing scheme and Reed-Solomon coding. Little is known about robust secret sharing for the range $n/3 \leq t < n/2$. The first scheme for the range $n/3 \leq t < n/2$ is due to Rabin and BenOr [22]. Their scheme consists of Shamir secret sharing, but enhanced by means of an unconditionally secure message authentication code. The constructions in [9,7] represent the two main approaches to this problem and provide the best performances in terms of trade-off results between share size and the reconstruction complexity (see Sect. 5 for their description). In [9], the redundancy of the share size is exactly two field elements (here $n = 2t+1$, secert size is one field element, and the reconstruction model is different). The reconstruction time is however exponential in n. The scheme in [7] has the smallest share size (see Sect. 6) among schemes with efficient (polynomial) reconstruction.

2 Preliminaries

We begin by formally defining the model of threshold robust secret sharing. The "definitions" are taken verbatim from [7].

2.1 Robust Secret Sharing

A threshold robust secret sharing scheme can be described by two interactive protocols, Share and Rec, where Share involves a dealer D and n players $P_1, \ldots, P_n$, and Rec involves the n players and a reconstructor R. The dealer is connected to every player by a secure, untappable channel. There is also a broadcast channel that can be used by everyone in the system. An n-player threshold robust secret sharing scheme for a secret space $\mathcal{S}$ consists of two phases, the *sharing* and the *reconstruction* phase, specified by two protocols Share and Rec respectively, described below. Let $[n] = \{1, \ldots, n\}$.

- Share: The dealer D takes as input a secret $s \in \mathcal{S}$, locally computes shares $\sigma_1, \ldots, \sigma_n$, and for every $i \in [n]$, sends the i-th share σ_i privately to player P_i.

– Rec: During reconstruction, player P_i, $i \in [n]$, communicates, possibly by means of several synchronous communication rounds, σ_i to the reconstructor R. The reconstructor R uses the received shares to produce an output s', which is supposed to be the original secret s.

2.2 Adversarial Capabilities

We now specify the capabilities (and limitations) of the adversary who has unbounded computing power. The goal of the adversary is to make the reconstructor output a value different from the original secret s.

– During the sharing phase, the adversary remains inactive, and does not learn any information about the secret as the shares are distributed using private channels between players and the dealer.
– After the sharing phase, the adversary can adaptively corrupt up to t of the players P_i, where t **is the threshold parameter**. The corruption can be done between communication rounds and continue as long as the total number of corrupted players does not exceed t. D or R are assumed incorruptible. Once a player P_i is corrupted, the adversary learns P_i's share σ_i, and from then on, the adversary has full control over P_i. The corruptions being adaptive means that after each corruption, the adversary can decide on whom to corrupt next depending on the shares he has seen so far.
– During the reconstruction phase, the adversary sees the communications between players P_i and the reconstructor R. Furthermore, he controls the information that the dishonest players send to the reconstructor R. Reconstruction in general has multiple rounds. In every communication round, for every corrupted player, the adversary decides what this player should send to R. A **rushing** adversary can choose these values after observing what honest players send to R in the current round. A **non-rushing** adversary selects the corrupted shares before the start of the reconstruction phase.

2.3 Security

An n-player robust secret sharing scheme (Share, Rec) is (t, δ)-secure if the following properties hold for any distribution of $s \in S$ and for any adversary as specified above:

1. Privacy: Before Rec starts, the adversary has no more information about the shared secret s than he had before the execution of Share.
2. Reconstructability: At the end of Rec, the reconstructor R outputs $s' = s$ with probability at least $1 - \delta$.

It is well known that in any perfect secret sharing scheme, the bit-size of a share σ_i is at least the same as the bit-size of secret, that is $\log |S|$ [25]. Much research

effort focused on finding the least required redundancy to achieve robustness. Let σ_i denotes the share for player P_i. The redundancy (also known as overhead) is measured by the quantity $\max_i\{\log \sigma_i\} - \log |S|$. For $t < n/3$, one can use Reed-Solomon error correcting codes to construct a robust secret sharing scheme with efficient reconstruction algorithm and no redundancy in the share size i.e., the share size is the same as the secret size [1]. On the other hand, for $t \geq n/2$ there is no solution to the problem (the (t, n)-threshold access structure requires at least $t + 1$ correct shares for the recovery of the original secret). In this work we construct a robust secret sharing scheme for $n = 2t + 2$ with no redundancy in the share size. The construction works for any t in the range $\frac{n}{3} \leq t \leq \frac{n}{2} - 1$.

3 The Proposed Scheme

The scheme of [9] can be understood as being obtained from a secret sharing scheme that allows error detection, i.e., that detects if a set of $t + 1$ shares contains some incorrect ones (but can not necessarily tell which ones). It was analyzed in [15] that any secret sharing scheme with error detection [5,21,27] can be transformed into a robust secret sharing scheme by looping over all sets of size $t+1$. This line of thinking has provided schemes with low share redundancy and the work in [9] represents the best so far. It is apparent that any such scheme will suffer from the same exponential complexity and our new proposal, being constructed on this line, is no exception, but what is interesting is that the proposed scheme employs a technique that leverage some extra public values to eliminate redundancy in the shares.

3.1 The Scheme

Let t and n are positive integers such that $n = 2t+2$. Let $\mathbb{F}_q$ be a finite field with q elements, where q is a prime power with $q > n$. We now present an n-player robust secret sharing scheme over $\mathbb{F}_q$ which is (t, δ) secure and individual share size is same as secret size.

- Share:
  - Let $s \in \mathbb{F}_q$ be a secret.
  - The dealer randomly chooses a polynomial $f(x) \in \mathbb{F}_q[x]$ of degree at most t such that $f(0) = s$ and computes $s_i = f(i)$ for all $i \in [t + 1]$.
  - D choose n vectors of length $t + 1$, $(r_{i1}, \ldots, r_{i(t+1)}) \in (\mathbb{F}_q)^n$, $1 \leq i \leq n$ such that any $t + 1$ of them are linearly independent (see below for such a selection) and for every $i \in [n]$, he computes $\sigma_i = \sum_{j=1}^{t+1} r_{ij} s_j \in \mathbb{F}_q$.
  - For every $i \in [n]$, the dealer D sends to player P_i the share σ_i (just one field element). The n vectors $\{(r_{i1}, \ldots, r_{i(t+1)})\}_{1 \leq i \leq n}$ are part of system's public parameters. The dealer can send the public parameters to users, using the broadcast channel. Alternatively the public parameters can be stored on a publicly accessible authenticated bulletin board.

– Rec:
  - Every player sends σ_i to the reconstructor R.
  - To reconstruct the secret, the reconstructor does the following for *every subset* of $t+1$ players.
    * He reconstructs $(s'_1, s'_2, \ldots, s'_{t+1})$ using $t+1$ shares by solving $t+1$ equations in $t+1$ variables.
    * He checks if $\sum_{j=1}^{t+1} r_{ij} s'_j = \sigma_i$ for *at least one of the remaining* $t+1$ shares, and halts if it holds.
  - R then computes (using Lagrange interpolation) a polynomial $f(x) \in \mathbb{F}_q[x]$ of degree at most t and outputs $s = f(0)$.

3.2 Remarks

Standard methods are available to choose n vectors of length $t+1$ over $\mathbb{F}_q$ with the property that any $t+1$ of them are linearly independent. For completeness we describe some of them here. Let $z_1, \ldots, z_n, w_1, \ldots, w_{t+1} \in \mathbb{F}_q$ be such that the z_i's are distinct, the w_j's are distinct, and $z_i + w_j \neq 0$ for all i, j. Define

$$r_i = \left(\frac{1}{z_i + w_1}, \ldots, \frac{1}{z_i + w_{t+1}} \right), \ 1 \leq i \leq n \ .$$

One can check that any $t+1$ vectors chosen among these n row vectors are linearly independent as the matrix so formed has non-zero determinant (see Ch 11, [17]). In particular let M denote the matrix with rows $r_1, \ldots, r_{t+1}$, then

$$\det(M) = \frac{\prod_{i<j}(z_j - z_i)(w_j - w_i)}{\prod_{i,j}(z_i + w_j)} \ .$$

The number of field elements that are distributed publicly is equal to $n+t+1$. Another way of selection is to choose an $n \times (t+1)$ Vandermonde matrix which also has the property that any $t+1$ rows are independent. A Vandermonde matrix of size $n \times (t+1)$ can be described by n elements and in this case only n field elements are distributed publicly.

4 Security

4.1 Perfect Secrecy

The secret is the constant term of a random polynomial of degree at most t. The $t+1$ evaluations of the polynomial, $\{s_1, \ldots, s_{t+1}\}$, are independent and are needed to reconstruct the secret. We will show that any group $\mathcal{CP} = \{P_{i_1}, \ldots, P_{i_t}\}$ of corrupted participants will be completely uncertain about at least one value from $\{s_1, \ldots, s_{t+1}\}$. This is true because the group $\mathcal{CP}$ has t shares $\{\sigma_{i_1}, \ldots, \sigma_{i_t}\}$ and these shares correspond to t equations,

$$M \cdot (s_1, \ldots, s_{t+1})^T = (\sigma_{i_1}, \ldots, \sigma_{i_t})^T \ ,$$

where M is the $t \times (t+1)$ matrix consisting of the t row vectors $r_{i_1}, \ldots r_{i_t}$ associated with the corrupted users, and 'T' denotes matrix transpose.

Let $M_1, \ldots, M_{t+1}$ be the column vectors of M. As M_i's constitute a set of $t+1$ t-dimensional vectors, they are linearly dependent. Thus there exists at least one column vector, without loss of generality say M_1, such that M_1 belongs to the subspace $\langle M_2, \ldots, M_{t+1} \rangle$. So there exists a $(t+1)$-dimensional vector $b = (b_1, \ldots, b_{t+1})$ such that $Mb^T = 0$ and $b_1 \neq 0$. Thus, we have

$$(\sigma_{i_1}, \ldots, \sigma_{i_t})^T = M \cdot (s_1, \ldots, s_{t+1})^T = M \cdot ((s_1, \ldots, s_{t+1})^T + \alpha(b_1, \ldots, b_{t+1})^T)$$

for all $\alpha \in \mathbb{F}_q$. Hence, given any $\beta_1 \in \mathbb{F}_q$, there exists $(\beta_1, \ldots, \beta_{t+1}) \in (\mathbb{F}_q)^{t+1}$ such that $M \cdot (\beta_1, \ldots, \beta_{t+1})^T = (\sigma_{i_1}, \ldots, \sigma_{i_t})^T$. Therefore, the participants in $\mathcal{CP}$ cannot rule out any element of $\mathbb{F}_q$ as a possibility for s_1. Thus, there exists q values for s_1 and distinct values for s_1 leads to distinct polynomials. This makes $f(0)$ indeterminate. □

4.2 Reliability

Theorem 1. *Let k be a security parameter. For any positive integer n and t such that $n = 2t + 2$, and any finite field $\mathbb{F}_q$ with $k = \lceil \log_2 q \rceil$, the pair* (Share, Rec) *forms an n-player (t, δ)-robust secret sharing for message space $\mathbb{F}_q$ with*

$$\delta \leq \frac{\sqrt{t+1}}{2^{k-n}} \ .$$

Proof. Consider the state of the reconstruction phase right before the reconstructor R has received the shares from the players. We may assume that at this stage the adversary has corrupted t players. Thus R has now n shares of which at most t are corrupted. To reconstruct the secret, R does the following for *every subset* of $t+1$ players.

(a) He computes $(s'_1, s'_2, \ldots, s'_{t+1})$ using $t+1$ chosen shares $(\sigma'_{i_1}, \sigma'_{i_2}, \ldots, \sigma'_{i_{t+1}})$ (by solving $t+1$ equations in $t+1$ variables).
(b) He then checks if $\sum_{j=1}^{t+1} r_{ij} s'_j = \sigma_i$ for *at least one of the remaining* $t+1$ shares, and halts if it holds.

Consider an arbitrary set $A = \{\sigma'_{i_1}, \ldots, \sigma'_{i_{t+1}}\}$ of $t+1$ shares submitted during the reconstruction phase. Let us assume that j $(0 \leq j \leq t)$ of them are corrupted. Let M be the matrix with rows $r_{i_1}, \ldots, r_{i_{t+1}}$ such that,

$$M \cdot (s'_1, \ldots, s'_{t+1})^T = (\sigma'_{i_1}, \ldots, \sigma'_{i_{t+1}})^T.$$

Then $(s'_1, \ldots, s'_{t+1})^T = \sigma'_{i_1} \tilde{M}_1 + \cdots + \sigma'_{i_{t+1}} \tilde{M}_{t+1}$, where the $\tilde{M}_i$'s are the columns of the inverse matrix M^{-1}. For a fix set of values of the j corrupted shares, there are q^{t+1-j} solution vectors to the above equality. Therefore the probability that the solution vector is a solution to one of the remaining equations is $\frac{t+1}{q^{t+1-j}}$, the maximum value is $\frac{t+1}{q}$ when $j = t$. Thus, taking into account union

bound over all subsets of size $t+1$ leaves us with the failure probability $\leq \frac{\sqrt{t+1}}{2^{k-n}}$ $\left(\text{as } (t+1) \cdot \binom{n}{t+1} \leq \sqrt{t+1} \cdot 2^n \text{ when } n = 2t+2 \text{ and } k = \lceil \log_2 q \rceil \right)$. □

The efficiency comparison for the proposed scheme with the known schemes (described below) is given in Sect. 6.

5 Known Schemes and Possible Extensions

Previous works on robust secret sharing schemes with unconditional security for the range $n/3 \leq t < n/2$ can be broadly divided into two classes. We now briefly recall the best scheme from each class.

The first one is due to Cramer et al. [9], based on an idea by [5]. The scheme works as follows. Using standard Shamir secret sharing, the dealer shares independently the actual secret $s \in \mathbb{F}_q$, a randomly chosen field element $r \in \mathbb{F}_q$, and their product $p = s \cdot r$. To reconstruct the secret, the reconstructor does the following: for every subset of $t+1$ players, he reconstructs s', r' and p' and checks if $s' \cdot r' = p'$, and halts and outputs s' if it is the case. One can show that for any subset of $t+1$ players: if $s' \neq s$ then $s' \cdot r' \neq p'$ except with probability $1/q$. Thus for a field of size 2^k, taking into account union bound over all subsets of size t + 1, gives a robust secret sharing scheme with failure probability 2^{k-n} and shares of size $3k$ bits (consisting of three field elements).

The second scheme is given by Cevellos, Fehr, Ostrovsky and Rabani [7], and is based on the scheme of Rabin and BenOr [22] with an elegant twist to its reconstruction algorithm. This scheme's description is given below.

– Share:
  - Choose a random polynomial $f(x) \in \mathbb{F}_q[X]$ with degree at most t such that $f(0) = s$.
  - Compute the Shamir shares $s_1 = f(x_1), \ldots, s_n = f(x_n)$, where x_i's are distinct points in $\mathbb{F}_q$.
  - For every pair $i, j \in [n]$, choose a random key $key_{ij} \in \mathcal{K}$ and compute $\tau_{ij} = MAC(key_{ji}, s_i)$, where $MAC : \mathbb{F}_q \times \mathcal{K} \to \mathcal{T}$ be an ϵ-secure MAC [29,30,7] with message space $\mathbb{F}_q$.
  - For every $i \in [n]$, the player P_i is given the share

$$\sigma_i = (s_i, \tau_{i1}, \ldots, \tau_{in}, key_{i1}, \ldots, key_{in}).$$

– Rec:
  - **First Round:** Every player P_i sends s_i and $\tau_{i1}, \ldots, \tau_{in}$. to the reconstructor $\mathcal{R}$.
  - **Second Round:** Every player P_i sends $key_{i1}, \ldots, key_{in}$ to $\mathcal{R}$.
  - **Local Computation:**
    * For every pair $i, j \in [n]$, $\mathcal{R}$ sets ν_{ij} to be 1 if the share s_i of player P_i is accepted by the corresponding key of player P_j, i.e., if $\tau_{ij} = \text{MAC}(key_{ji}, s_i)$, and else to 0.

* $\mathcal{R}$ computes the largest set $\mathcal{I} \subseteq [n]$ with the property that

$$\forall i \in \mathcal{I} : |\{j \in \mathcal{I} | \nu_{ij} = 1\}| = \sum_{j \in \mathcal{I}} \nu_{ij} \geq t + 1 \ ;$$

in other words, every share of a player in $\mathcal{I}$ is accepted by at least $t+1$ players in $\mathcal{I}$. Clearly $\mathcal{I}$ contains all honest players. Let $c = |\mathcal{I}| - (t+1)$ be the maximum number of corrupt players in $\mathcal{I}$.

* Use the Berlekamp-Welch algorithm [2,14] to compute a polynomial $f(x) \in \mathbb{F}[X]$ of degree at most t such that $f(x_i) = s_i$ for *at least* $(t+1) + \frac{c}{2}$ players i in $\mathcal{I}$. If no such polynomial exists then $\mathcal{R}$ outputs $\perp$; otherwise, he outputs $s = f(0)$.

The Share algorithm of this scheme is the same as the well-known scheme of Rabin and Ben-Or [22] which relies on message authentication. The redundancy in share size for Rabin and Ben-Or scheme consists of $3n$ elements from the field where the secret is drawn from. The scheme uses a message authentication code with *short* tags and keys and with the resulting weak security. The short tags and keys result in the required saving (improvement over Rabin and Ben-Or scheme) in the share size. The weakened security of authentication (and so higher chance of forging) is compensated with a more sophisticated reconstruction procedure which runs in polynomial time and results in an exponentially small failure probability. The overhead of the share size depends directly on the exponent of the failure probability.

Assuming the same share distribution as Rabin and Ben-Or's scheme [22], one may consider further reduction in authentication information and improvement in the reconstruction, to obtain shorter share sizes. In Sect. 5.1 we explore one such possibility by employing list decoding algorithm for Reed-Solomon codes [26] in the reconstruction algorithm of [7]. Our goal is to reduce δ, the error probability of the decoder, which will translate into smaller share size. Our analysis shows that this modification does not reduce δ and so the share size cannot be further reduced.

5.1 Using List Decoding to Improve Decoding Error in [7]

We begin by describing a natural modification to the Cevallos et al.'s Scheme.

- Share: Same
- Rec:
  - **First Round:** Same
  - **Second Round:** Same
  - **Local Computation:** Step 1 and 2 are the same as above. Recall that $c = |\mathcal{I}| - (t+1)$ is the maximum number of corrupt players in $\mathcal{I}$.
    * **Step 3 of Cevallos et al. scheme:** Use Berlekamp-Welch to compute a polynomial $f(x) \in \mathbb{F}[X]$ of degree at most t such that $f(x_i) \neq s_i$ for *at most* $\frac{c}{2}$ players in $\mathcal{I}$.

* **Modification:** Use list decoding algorithm for $[n = 2t+1, k = t+1, d = n-k+1 = t+1]$ Reed-Solomon codes [26] that corrects up to $n-\sqrt{nt}$ of errors, to compute a (list of) polynomial(s) $f(x) \in \mathbb{F}[X]$ of degree at most t such that $f(x_i) \neq s_i$ for at most $\frac{1}{1+\sqrt{\frac{t}{t+1+c}}}(c+1)$ players in $\mathcal{I}$.
* Find correct f from the decoding list and output $s = f(0)$.

5.2 Robustness

The analysis is similar to [7]. Define the following sets: $\mathcal{A} \subset [n]$ is the set of corrupted players that have handed in modified Shamir shares, and $\mathcal{P} \subset [n]$ is the set of corrupted players that have handed in the correct Shamir shares. It holds that $|\mathcal{A}| + |\mathcal{P}| = t$. The set $\mathcal{H} = [n] \backslash (\mathcal{A} \cup \mathcal{P})$ is the set of uncorrupted players.

The set $\mathcal{I}$ computed during reconstruction contains $\mathcal{H}$ and $\mathcal{P}$ with certainty. Thus, the reconstruction procedure is guaranteed to output the correct secret if at most $\frac{1}{\theta} \cdot (p+1)$ players $i \in \mathcal{A}$ end up in $\mathcal{I}$, where $p = |\mathcal{P}|$ and $\theta = \sqrt{\frac{t}{n}}$. Indeed, if $|\mathcal{A} \cap \mathcal{I}| \leq \frac{1}{\theta} \cdot (p+1)$, then the requirement for list-decoding is satisfied ($|\mathcal{I}| = t+1+c = t+1+p+e$ where $e = |\mathcal{A} \cap \mathcal{I}| \leq \frac{1}{\theta} \cdot (p+1)$ and thus $e = \frac{1}{\theta} \cdot (p+1) = \frac{1}{1+\theta}(p + \frac{1}{\theta}(p+1)+1) = \frac{1}{1+\theta}(c+1)$).

We need to find the probability $P[|\mathcal{A} \cap \mathcal{I}| > \frac{1}{\theta} \cdot (p+1)]$. It is sufficient to consider the case $p \leq \frac{\theta}{1+\theta} \cdot t$; indeed if $p > \frac{\theta}{1+\theta} \cdot t$ and thus $p \geq \frac{\theta}{1+\theta} \cdot (t-1)$ then obviously $|\mathcal{A}| \leq t - \frac{\theta}{1+\theta} \cdot (t-1) = \frac{1}{1+\theta} \cdot (t+1)$ and hence $P[|\mathcal{A} \cap \mathcal{P}| \leq \frac{1}{\theta} \cdot (p+1)]$. Thus

$$\delta = \sum_{p=1}^{\frac{\theta}{1+\theta} \cdot t} P[|\mathcal{A} \cap \mathcal{I}| > \frac{1}{\theta} \cdot (p+1)] \ .$$

For any p in the range $1 \leq p \leq \frac{\theta}{1+\theta} \cdot t$, we first compute $P[|\mathcal{A} \cap \mathcal{I}| > \frac{1}{\theta} \cdot (p+1)]$. Let us assume that $\frac{1}{\theta} \cdot (p+1)$ is an integer. In order to bound the above probability, it is convenient to introduce the following random variables:

- For every pair $i, j \in [n]$, we define the binary random variable V_{ij} that specifies if the player P_i's share and his submitted tag associated with player P_j are accepted by player P_j's key. Note that, all the V_{ij} with $i \in [n]$ and $j \in \mathcal{H}$ are independent. Further $P[V_{ij} = 1] \leq \epsilon$ for all $i \in \mathcal{A}$ and $j \in \mathcal{H}$.
- For every $i \in \mathcal{A}$ the random variable

$$N_i = \sum_{j \in \mathcal{H}} V_{ij} = |\{j \in \mathcal{H} | V_{ij} = 1\}| \ ,$$

  i.e., the number of honest players that accept P_i's incorrect share. Note that since the V_{ij}'s are independent for all $i \in [n]$ and $j \in \mathcal{H}$, so are all the N_i's.

$$
\begin{aligned}
P[|\mathcal{A}\cap\mathcal{I}| > \frac{1}{\theta}\cdot(p+1)] &= P[\Big(|\mathcal{A}\cap\mathcal{I}| = \frac{1}{\theta}\cdot(p+1)+1\Big)\cup\cdots\cup\Big(|\mathcal{A}\cap\mathcal{I}| = t-\frac{1}{\theta}\cdot(p+1)\Big)] \\
&\leq P[|\mathcal{A}\cap\mathcal{I}| = t-\frac{1}{\theta}\cdot(p+1)] \text{ (best strategy for adversary)} \\
&= P[\cap_{i\in\mathcal{A}\setminus\mathcal{P}}(N_i=1)] \\
&= \prod_{i\in\mathcal{A}\setminus\mathcal{P}} P[N_i=1] \\
&= \prod_{i\in\mathcal{A}\setminus\mathcal{P}} P[\exists\mathcal{H}_0\subseteq\mathcal{H}:(|\mathcal{H}_0|=1)\wedge(\forall j\in\mathcal{H}_0:V_{ij}=1)] \\
&\leq \prod_{i\in\mathcal{A}\setminus\mathcal{P}}\left(\sum_{\mathcal{H}_0\subseteq\mathcal{H}:|\mathcal{H}_0|=1} P[\forall j\in\mathcal{H}_0:V_{ij}=1)]\right) \\
&\leq \prod_{i\in\mathcal{A}\setminus\mathcal{P}}((t+1)\cdot\epsilon) \\
&= ((t+1)\cdot\epsilon)^{t-\frac{1}{\theta}\cdot(p+1)}
\end{aligned}
$$

We can now compute the robustness probability as follows:

$$
\begin{aligned}
\delta &= \sum_{p=1}^{\frac{\theta}{1+\theta}\cdot t} P[|\mathcal{A}\cap\mathcal{I}| > \frac{1}{\theta}\cdot(p+1)] \\
&\leq \sum_{p=1}^{\frac{\theta}{1+\theta}\cdot t} ((t+1)\cdot\epsilon)^{t-\frac{1}{\theta}\cdot(p+1)} \\
&\leq ((t+1)\cdot\epsilon)^{\frac{\theta}{1+\theta}\cdot t-\frac{1}{\theta}}(1+((t+1)\cdot\epsilon)^{\frac{1}{\theta}}+((t+1)\cdot\epsilon)^{\frac{2}{\theta}}+\cdots) \text{ [assuming, } \epsilon\leq\frac{1}{t+1}] \\
&\leq 2((t+1)\cdot\epsilon)^{\frac{\theta}{1+\theta}\cdot t-\frac{1}{\theta}}
\end{aligned}
$$

Note that the bound on δ is similar to the bound while considering the Berlekamp-Welch setting. Also note that we have not included the analysis for the required probability to find the correct polynomial from the list of polynomials output by the list decoding algorithm.

6 Efficiency Comparison

In this section we compare the efficiency of our scheme, in terms of relation among the following three parameters: secret size, the share size and the reliability in the reconstruction, with the schemes of Cramer et al. [9] and Cevallos et al. [7]. Note that our scheme works for $n \geq 2t+2$ while the other two schemes work for $n \geq 2t+1$. To share a k-bit secret among the n players using our proposed scheme, the failure probability is at most $\frac{\sqrt{t+1}}{2^{k-n}}$, and for the Cramer et al. scheme it is $\frac{1}{2^{k-n}}$. The share size for the two schemes are k bits and $3k$ bits, respectively.

Understanding the relation for [7] is more subtle. Here the failure probability depends on an extra parameter. Let λ be a parameter that can be chosen independent of the secret size k. The two parameters λ, k are used in the following MAC function which has been used in [7]:

$$MAC : GF(2^k) \times (GF(2^{k/\lambda}))^2 \to GF(2^{k/\lambda}) \ .$$

Sharing a k-bit secret among the n players using the scheme [7], results in the failure probability of at most $\frac{1}{2^{n\frac{k}{\lambda} - n\log(n\cdot\lambda)}}$. For $\lambda \leq n$, the failure probability is less than for the other schemes. The share size for [7] is $k + 3n\frac{k}{\lambda}$. Clearly [7] has efficient reconstruction complexity and improved failure probability. However the share size is higher than the other two schemes. In the table, the secret size and share size are given in bits.

Table 1. Comparison Table

Scheme	Secret size	Share size	Rec Complexity	δ	Public Parameters
[9]	k	$3k$	Exp. in n	$2^{-(k-n)}$	Nil
[7]	k	$k + 3n\frac{k}{\lambda}$	Poly. in n	$2^{-(n\frac{k}{\lambda} - n\log(n\cdot\lambda))}$	Nil
Proposed Scheme	k	k	Exp. in n	$\sqrt{t+1}\cdot 2^{-(k-n)}$	n field elements

The last column for public parameters represents the elements that are required in addition to the interpolating points for Shamir's secert sharing scheme.

7 Conclusion

The problem of the minimum share size for threshold robust secret sharing has received considerable attention in recent years. In this paper, we proposed and analyzed a new threshold robust secret sharing scheme for which the share size of participants is the same as the secret size. This is the minimum possible value for the share size of a perfect secret sharing scheme and hence also the least possible share size for threshold robust secret sharing. The result is interesting as it means that the extra robustness property can be obtained with no extra cost on the share size. However the scheme works only for $n \geq 2t + 2$ and effectively uses the share of one extra honest participant as the verification information. The reconstruction algorithm is exponential in the number of players. Construction of schemes with efficient reconstruction in our setting remains an open problem.

Acknowledgments. Financial support for this research was provided in part by Alberta Innovates - Technology Futures, in the Province of Alberta in Canada. The authors would also like to thank Pengwei Wang for many useful discussions.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon, J. (ed.) STOC 1988, pp. 1–10. ACM (1988)
2. Berlekamp, E.R., Welch, L.R.: Error correction of algebraic block codes. U.S. patent number 4.633.470 (1986)
3. Blakley, G.: Safeguarding cryptographic keys. In: AFIPS National Computer Conference, vol. 48, pp. 313–317 (1979)
4. Brickell, E.F., Stinson, D.R.: The detection of cheaters in threshold schemes. SIAM J. Discrete Math. 4(4), 502–510 (1991)
5. Cabello, S., Padró, C., Sáez, G.: Secret sharing schemes with detection of cheaters for a general access structure. In: Ciobanu, G., Păun, G. (eds.) FCT 1999. LNCS, vol. 1684, pp. 185–194. Springer, Heidelberg (1999)
6. Carpentieri, M., De Santis, A., Vaccaro, U.: Size of shares and probability of cheating in threshold schemes. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 118–125. Springer, Heidelberg (1994)
7. Cevallos, A., Fehr, S., Ostrovsky, R., Rabani, Y.: Unconditionally-secure robust secret sharing with compact shares. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 195–208. Springer, Heidelberg (2012)
8. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: FOCS 1985, pp. 383–395. IEEE Computer Society (1985)
9. Cramer, R., Damgård, I., Fehr, S.: On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 503–523. Springer, Heidelberg (2001)
10. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. In: FOCS 1990, pp. 36–45. IEEE Computer Society (1990)
11. Ganger, G.R., Khosla, P.K., Bakkaloglu, M., Bigrigg, M.W., Goodson, G.R., Oguz, S., Pandurangan, V., Soules, C.A.N., Strunk, J.D., Wylie, J.J.: Survivable storage systems. In: Proceedings of DARPA Information Survivability Conference & Exposition II, DISCEX 2001, vol. 2, pp. 184–195 (2001)
12. Garay, J.A., Givens, C., Ostrovsky, R.: Secure message transmission with small public discussion. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 177–196. Springer, Heidelberg (2010)
13. Garay, J., Givens, C., Ostrovsky, R.: Secure message transmission by public discussion: A brief survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 126–141. Springer, Heidelberg (2011)
14. Gemmell, P., Sudan, M.: Highly resilient correctors for polynomials. Inf. Process. Lett. 43(4), 169–174 (1992)
15. Kurosawa, K., Suzuki, K.: Almost secure (1-round, n-channel) message transmission scheme. IEICE Transactions 92-A(1), 105–112 (2009)
16. Lakshmanan, S., Ahamad, M., Venkateswaran, H.: Responsive security for stored data. IEEE Trans. Parallel Distrib. Syst. 14(9), 818–828 (2003)
17. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes. North-Holland publishing company
18. Martin, K.M., Paterson, M.B., Stinson, D.R.: Error decodable secret sharing and one-round perfectly secure message transmission for general adversary structures. Cryptography and Communications 3(2), 65–86 (2011)

19. McEliece, R.J., Sarwate, D.V.: On sharing secrets and Reed-Solomon codes. Commun. ACM 24(9), 583–584 (1981)
20. Obana, S.: Almost optimum t-cheater identifiable secret sharing schemes. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 284–302. Springer, Heidelberg (2011)
21. Ogata, W., Kurosawa, K., Stinson, D.R.: Optimum secret sharing scheme secure against cheating. SIAM J. Discrete Math. 20(1), 79–95 (2006)
22. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: Johnson, D.S. (ed.) STOC 1989, pp. 73–85. ACM (1989)
23. Rogaway, P., Bellare, M.: Robust computational secret sharing and a unified account of classical secret-sharing goals. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 172–184. ACM (2007)
24. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
25. Stinson, D.R.: An explication of secret sharing schemes. Des. Codes Cryptography 2(4), 357–390 (1992)
26. Sudan, M.: Decoding of Reed-Solomon codes beyond the error-correction bound. J. Complexity 13(1), 180–193 (1997)
27. Tompa, M., Woll, H.: How to share a secret with cheaters. J. Cryptology 1(2), 133–138 (1988)
28. Waldman, M., Rubin, A.D., Cranor, L.F.: The architecture of robust publishing systems. ACM Trans. Internet Techn. 1(2), 199–230 (2001)
29. Wegman, M.N., Carter, L.: New classes and applications of hash functions. In: FOCS 1979, pp. 175–182. IEEE Computer Society (1979)
30. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. 22(3), 265–279 (1981)

A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data

Marc Joye and Benoît Libert

Technicolor
975 avenue des Champs Blancs, 35576 Cesson-Sévigné Cedex, France
{marc.joye,benoit.libert}@technicolor.com

Abstract. Suppose that a set of multiple users uploads in every time period encrypted values of some data. The considered problem is how an *untrusted* data aggregator can compute the sum of all users' values but nothing more. A solution was recently given by Shi *et al.* (NDSS 2011). However, as advocated by the authors, the proposed encryption scheme suffers from some limitations. In particular, its usage is restricted to small plaintext spaces. This paper presents a practical scheme which, advantageously, can accommodate large plaintext spaces. Somewhat surprisingly, it comes with an efficient security reduction, regardless of the number of users. Furthermore, the proposed scheme requires a minimal number of interactions, is efficient for both encryption and decryption/aggregation and can operate in an off-line/on-line mode.

Keywords: Private aggregation, smart metering, homomorphic encryption, large data sets.

1 Introduction

A fundamental problem is that of private data analysis where a third party has to compute some aggregate statistics over some sensitive data held by individuals. This problem finds concrete applications in a number of situations. When the third party, called hereafter *aggregator*, is trusted an easy solution would be to ask the users to encrypt their data using the aggregator's public key. Upon receiving the ciphertexts the aggregator applies its private key to recover the data in clear and then compute statistics. The problem becomes much more challenging in the case of an *untrusted* aggregator. This is the setting we are dealing with in this paper.

As an illustrative example, consider the case of smart energy metering (e.g., gas, electricity or water). Frequent aggregates of consumption over a population of users is very useful to finely tune the service by adapting the load or forecasting the supply, which results in better prices. It also reveals useful to rapidly detect anomalies on the grid in the case of accidental leakage. While the computation of aggregate statistics is beneficial to the consumers, it may legitimately raise privacy concerns. Electricity smart meters typically report the electricity usage every 15 minutes. Such data can be used to infer information

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 111–125, 2013.

about users' behaviors (e.g., when they are at home and what they do). Other examples include collecting medical data for disease monitoring or developing new drugs, collecting users' preferences for recommendation systems, etc.

The previous examples clearly highlight the need of allowing users to privately share their data while at the same time allowing a [non-necessarily trusted] aggregator to carry out aggregate statistics. Different types of statistics can be computed over private data. Sums and averages are widespread examples.

Privacy-preserving protocols. Several approaches can be found in the literature (see [18] for a comprehensive survey of these) to address the problem of computing the sum of time-series data, in a privacy-preserving fashion.[1]

Borrowing the taxonomy of [18], general privacy-preserving protocols can be characterized from the following dimensions:

- *Aggregate function:* This is the function evaluated by the aggregator. In our case, the function is the sum of the users' private inputs.
- *User synchronization:* A protocol is said asynchronous if the users can report their data independently of each other to the aggregator and at any time. Following the terminology of [18], our protocol is synchronous as we consider time-series data aggregation and so users should report their data at approximatively the same time (*i.e.*, the time when aggregation is computed). However, it does *not* require interaction among users. In that sense, the protocol we propose in this paper is synchronous yet non-interactive.
- *Fault tolerance:* This notion captures the capability of coping with failures. Our basic protocol assumes that there are no failures: all users report their data. This basic protocol can nevertheless be adapted to support failures; see [8,17].
- *Communication model:* The communication between the users and the aggregator can be unidirectional or bidirectional. The bidirectional setting requires a return channel from the aggregator to the users or among users. Our protocol merely requires an unidirectional communication link to the aggregator.
- *Privacy notions:* There are different flavors of privacy. Our protocol provably achieves the strongest notion of aggregator-obliviousness: The aggregator learns nothing beyond the output of the aggregate function. See Section 3.
- *Aggregate error:* The output of the aggregate function can be exact or noisy. Our basic protocol returns the exact sum of all users' data. It can however be modified to return a noisy sum through differential-privacy techniques, similarly to [25].
- *Group management:* This last notion indicates if the protocol is static or dynamic. Dynamic protocols allow users to join or leave the group without requiring a new set-up phase. As described, our protocol is static. Techniques for dealing with dynamic settings can be found in [8,17].

[1] We note that very efficient solutions are known, based on symmetric-key cryptography, in the case of a *trusted* aggregator. See e.g. [7].

Related work. In [24], Rastogi and Nath suggested that each user encrypts her private data using an additively homomorphic encryption scheme. The aggregator collects all the ciphertexts, aggregates them to get the encryption of the aggregate sum and sends this value back to the users. All the users, who possess each a share of the decryption key, contribute to the decryption of the aggregate sum by sending their decryption share to the aggregator. The aggregator then combines all decryption shares received from the users to get in clear the sum of the users' private data.

Like other proposals [1,20,21], the approach of [24] requires a bidirectional communication link between the users and the aggregator. Unfortunately a return channel is not always available. To alleviate this issue, Garcia and Jacobs [15] came up with an aggregation protocol based on homomorphic encryption and additive secret sharing. Their proposal eliminates the need for a bidirectional channel. On the downside, the communication complexity is quadratic in the number of users and each user has to compute a linear number of homomorphic encryptions.

Independently, Kursawe *et al.* [19] described four protocols allowing the aggregation of users' data and comparisons among data aggregates without using bidirectional channels. Nevertheless, the first three protocols incur interaction, Diffie-Hellman key agreements or bilinear map evaluations. They thus require either user-to-user communication or somewhat costly arithmetic operations. The fourth protocol of [19] has low overhead but, like the previous ones, it requires each user to store the fixed public key of all other users.

Back in 2011, Shi, Chan, Rieffel, Chow and Song [25] described a completely non-interactive solution. They astutely suggested to split the aggregator capability, viewed as an un-blinding key s_0, into additive shares among the set of users, say $s_0 = \sum_i s_i$. Each user makes use of her secret share s_i and blinds her private data $x_{i,t}$ at time period t with a one-time mask derived from s_i and t to obtain a masked value $c_{i,t}$. At each time period, the aggregator collects all the $c_{i,t}$'s. The difficulty resides in finding a scheme such that the aggregation results in the private data adding together and the masks summing up to a value depending only on t and on $\sum_i s_i = s_0$. Using the un-blinding key s_0, the aggregator can so recover in clear the sum of the users' private data. Shi *et al.* [25] gave a solution that generically works in any prime order group where the Decision Diffie-Hellman (DDH) assumption holds. They also provided a security proof in a formally defined privacy model. See § 3.2 for a detailed description of their scheme. The authors however left open a couple of research challenges (cf. [25, Section 8]). One of them is that of user failure (fault tolerance) and efficient support of dynamic joins and leaves. Solutions to this problem have been found in [1,8,17].

Among the fault-tolerant systems [1,8,17], the construction of Ács and Castellucia [1] requires shared keys among all pairs of users, which incurs a linear storage in the size of the network at each user. Moreover, users should also be able to receive messages from the aggregator. In a recent work, Jawurek and Kerschbaum [17] managed to avoid these overheads in a protocol that further

computes *weighted* sums. On one hand, their protocol allows for an arbitrary number of missing inputs from users. It also eliminates the need for synchronization among these. On the other hand, it involves distributed key managing authorities whose task is to jointly decrypt Paillier encryptions during the protocol. As a result, these key managing authorities have to be involved—and interact with the user performing the calculation—in each aggregation operation.

Our contribution. In this paper, we reconsider the fully non-interactive construction of Shi *et al.* [25] and show how to get rid of one of its limitations. An important challenge left open in [25] (and also discussed in [8]) is that of efficiently computing sums over large plaintext spaces. This paper proposes a scheme that supports large plaintext spaces and number of users. At the same time, the decryption algorithm operates in constant time, regardless of the number of users. As a bonus, the scheme also provides a great on-line/off-line efficiency: namely, using pre-computations, the encryptor is left with a mere modular multiplication in the on-line phase (*i.e.*, when the data to be encrypted is known), which is highly desirable when computations take place on resource-limited devices.

To achieve this, we follow the research direction suggested in [25], namely investigating other algebraic settings. The natural candidate is the Paillier cryptosystem [23]. However, the proof given in [25] does not seem to readily extend to this setting due to technical difficulties arising when we want to rely on the DDH assumption. The main hurdle appears to find a way to efficiently hash to a cyclic subgroup of hidden order while remaining able—in order to properly simulate the adversary's view in the security proof—to explain hash values as being obtained by exponentiating the output of a random oracle whose range has no specific algebraic structure. While several techniques are known (e.g., [5,16]) to efficiently hash onto cyclic groups of public order, this turns out to be more complicated in DDH-hard groups of hidden order like the subgroup of squares modulo N or N^2, for some moduli $N = pq$ of unknown factorization. By trading the DDH assumption for the Decision Composite Residuosity (DCR) assumption [23], we eliminate the need to hash onto a group of hidden order. Somewhat surprisingly, we are moreover able to derive a security proof with a much tighter (*i.e.*, independent of the number of users) reduction in the random oracle model [2]. This in turn results in better parameters when the scheme is concretely instantiated.

While practical, our scheme inherits the limitations of [25] in that it does not allow partial aggregations in the presence of missing contributions. Unlike [20,21,17], it does not extend to compute weighted sums and, unlike [21,17], it requires synchronization among participants. On the other hand, our system inherits the main advantage of [25]: the data aggregator can process users' data by itself without having to interact with a distributed key-managing authority at each aggregation. Here, the trusted party that generates the Paillier public key is never involved in any aggregation and can remain off-line after the setup phase. As for its advantages over [1], our construction only requires users to store $O(1)$ values and does not require a bidirectional channel.

Like [24,25], the proposed scheme can serve as a building block for the fault-tolerant solution of [8] while enjoying the benefits of our construction. In fact, all the extensions of [25] are also possible with our system. In particular, although the focus of this paper is put on the encryption, the proposed scheme is compatible with the differential-privacy framework [13,12]. In this case, the users simply need to add some appropriately-generated noise to their data prior to encryption.

Outline of the paper. The rest of this paper is organized as follows. The next section introduces some mathematical background and related cryptographic problems. Section 3 explains what is meant by aggregator obliviousness. Section 4 is the core of the paper. It presents a new aggregator-oblivious encryption scheme, a security proof of which is provided in Section 5. Finally, Section 6 discusses the performance of the proposed scheme and some implementation issues.

2 Preliminaries

In this section we review the necessary background and introduce some notation used throughout the paper.

2.1 N-th Residues and Discrete Logarithms in $(\mathbb{Z}/N^2\mathbb{Z})^*$

Let p be a prime. Consider the ring $\mathbb{Z}/p^2\mathbb{Z} = \{0, 1, \ldots, p^2-1\}$. Its multiplicative group is given by $(\mathbb{Z}/p^2\mathbb{Z})^* = \{x \in \mathbb{Z}/p^2\mathbb{Z} \mid \gcd(x,p) = 1\}$. This group has $p(p-1)$ elements. Letting $\mathrm{ord}(x)$ denote the order of x as an element of $(\mathbb{Z}/p^2\mathbb{Z})^*$ (*i.e.*, the smallest nonnegative integer α such that $x^\alpha \equiv 1 \pmod{p^2}$), Lagrange's theorem tells that $\mathrm{ord}(x)$ divides $p(p-1)$. In particular, $(\mathbb{Z}/p^2\mathbb{Z})^*$ has a p-order subgroup Γ_p given by

$$\Gamma_p = \{x \in (\mathbb{Z}/p^2\mathbb{Z})^* \mid x^p \equiv 1 \pmod{p^2}\} \ .$$

Suppose now that $x = 1 + \beta p$ with $\beta \in \{0, \ldots, p-1\}$—observe that such an x is in $(\mathbb{Z}/p^2\mathbb{Z})^*$ since $\gcd(1+\beta p, p) = 1$. Further, the binomial identity yields

$$(1+\beta p)^t \equiv \sum_{k=0}^{t} \binom{t}{k} 1^{n-k} (\beta t)^k \equiv \binom{t}{0} + \binom{t}{1} \beta p \equiv 1 + t\beta p \pmod{p^2} \ .$$

Hence, we have $(1+\beta p)^p \equiv 1 \pmod{p^2}$ and consequently $1 + \beta p \in \Gamma_p$. From $\Gamma_p \cong (\mathbb{Z}/p\mathbb{Z})^+$ (a cyclic group), we also deduce that any element $x \neq 1 \in \Gamma_p$ generates it. In particular, $1 + p$ is a generator of Γ_p. As a result, we get

$$\Gamma_p = \{(1+p)^\beta \pmod{p^2} \mid \beta \in \{0, \ldots, p-1\}\} = \{1 + \beta p \mid \beta \in \{0, \ldots, p-1\}\} \ .$$

Let now $N = pq$ where p and q are (distinct) primes. By Chinese remaindering, we can construct the multiplicative group $(\mathbb{Z}/N^2\mathbb{Z})^* \cong (\mathbb{Z}/p^2\mathbb{Z})^* \times (\mathbb{Z}/q^2\mathbb{Z})^*$. This group has order $\#(\mathbb{Z}/N^2\mathbb{Z})^* = \#(\mathbb{Z}/p^2\mathbb{Z})^* \times \#(\mathbb{Z}/q^2\mathbb{Z})^* = p(p-1)q(q-1) =$

$N\phi(N)$ where $\phi(N) = (p-1)(q-1)$ is Euler's totient function. Similarly, we can define $\Gamma_N \cong \Gamma_p \times \Gamma_q$ with $\#\Gamma_N = \#\Gamma_p \times \#\Gamma_q = pq = N$ as

$$\begin{aligned}\Gamma_N &= \{(1+N)^\beta \pmod{N^2} \mid \beta \in \{0,\ldots,N-1\}\} \\ &= \{1+\beta N \mid \beta \in \{0,\ldots,N-1\}\} \ .\end{aligned}$$

Again, it is worth noticing that $(1+N)^\beta \bmod N^2 = 1+\beta N$; note also that β is defined modulo N. This shows that computing discrete logarithms in Γ_N is easy. Namely, given two random elements $u, v \in \Gamma_N$, it is easy to find α such that $v = u^\alpha \pmod{N^2}$:

$$\alpha = \frac{L(v)}{L(u)} \bmod N \quad \text{where } L(x) = \frac{x-1}{N} \ .$$

Proof. Since $u \in \Gamma_N$, we have $u \equiv 1 \pmod{N}$ and so we can write $u = 1+\beta N$ where $\beta = (u-1)/N = L(u)$. Likewise, we can write $v = 1 + L(v)N$. Since we have that $u^\alpha = v$, it follows that $(1+L(u)N)^\alpha \equiv 1 + L(v)N \pmod{N^2} \iff 1+\alpha L(u)N \equiv 1+L(v)N \pmod{N^2} \iff N(\alpha L(u) - L(v)) \equiv 0 \pmod{N^2}$. Since $N(\alpha L(u) - L(v)) \bmod N^2 = N[(\alpha L(u) - L(v)) \bmod N]$, we get $\alpha L(u) \equiv L(v) \pmod{N}$ and consequently $\alpha = L(v)/L(u) \bmod N$. □

Another subgroup of $(\mathbb{Z}/N^2\mathbb{Z})^*$ is the *group of N-th residues* given by

$$\mathfrak{S}_N = \{x^N \pmod{N^2} \mid x \in (\mathbb{Z}/N^2\mathbb{Z})^*\} \ .$$

This subgroup has order $\phi(N)$ and computing discrete logarithms in $\mathfrak{S}_N$ is as hard as computing discrete logarithms in $(\mathbb{Z}/N\mathbb{Z})^*$ (see [22,26]). Given two elements $u, v \in \mathfrak{S}_N$ with $v = u^\alpha \pmod{N^2}$ for some $\alpha \in \{0,\ldots,\phi(N)\}$, it turns out that $v \equiv u^\alpha \pmod{N}$. It is also worth observing that each N-th residue possesses exactly N roots of order N, of which only one is smaller than N. Indeed, if $y \equiv x^N \pmod{N}$ then so is $y \equiv (x_0 + \beta N)^N \pmod{N^2}$ with $x_0 := x \bmod N$ and $\beta \in \{0,\ldots,N-1\}$.

2.2 DCR Complexity Assumption

We need the following hardness assumption, introduced in [23], which is implied by the commonly assumed intractability of factoring.

Definition 1. *Let $N = pq$ be a product of two large primes. The* Decision Composite Residuosity (DCR) *assumption in $(\mathbb{Z}/N^2\mathbb{Z})^*$ is to distinguish among the following two distributions given only $N = pq$:*

$$D_0 = \{x^N \pmod{N^2} \mid x \in_R (\mathbb{Z}/N^2\mathbb{Z})^*\} \quad \textit{and} \quad D_1 = \{x \in_R (\mathbb{Z}/N^2\mathbb{Z})^*\} \ .$$

The DCR assumption *states that the advantage of any distinguisher $\mathcal{D}$, defined as the distance*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{DCR}}(\mathcal{D}) = \big|\Pr[\mathcal{D}(y,N) = 1 \mid y = x^N \bmod N^2, x \in_R (\mathbb{Z}/N\mathbb{Z})^*] \\ - \Pr[\mathcal{D}(y,N) = 1 \mid y \in_R (\mathbb{Z}/N^2\mathbb{Z})^*]\big|\end{aligned}$$

where probabilities are taken over all coin tosses, is a negligible function.

3 Aggregator-Oblivious Encryption

We start with the definition and then proceed with the corresponding security notion. We refer the reader to [25] for further introductory background.

Definition 2. *An* aggregator-oblivious encryption scheme *is a tuple of algorithms,* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{AggrDec})$*, defined as:*

$\mathsf{Setup}(1^\kappa)$ *On input a security parameter* κ*, a trusted dealer generates the system parameters* param*, the aggregator's private key* sk_0*, and the private key* sk_i *for each user* i *(*$1 \leq i \leq n$*);*

$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, x_{i,t})$ *At time period* t*, user* i *encrypts a value* $x_{i,t}$ *using her private encryption key* sk_i *to get* $c_{i,t} = \mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, x_{i,t})$*.*

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, c_{1,t}, \ldots, c_{n,t})$ *At time period* t*, the aggregator using* sk_0 *obtains* $X_t = \sum_{i=1}^{n} x_{i,t}$ *as* $X_t = \mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, c_{1,t}, \ldots, c_{n,t})$*.*

3.1 Aggregator Obliviousness

Basically, the security notion of *aggregator obliviousness* (AO) requires that the aggregator cannot learn, for each time period, anything more than the aggregate value X_t from the encrypted values of n (honest) users. If there are corrupted users (*i.e.*, users sharing their private information with the aggregator), the notion only requires that the aggregator gets no extra information about the values of the honest users beyond that their aggregate value. Furthermore, it is assumed that each user encrypts only one value per time period.

More formally, AO is defined by the following game between a challenger and an attacker.

Setup. The challenger runs the Setup algorithm and gives param to the attacker.

Queries. In a first phase, the attacker can submit queries that are answered by the challenger. The attacker can make two types of queries:

1. Encryption queries: The attacker submits $(i, t, x_{i,t})$ for a pair (i, t) and gets back the encryption of $x_{i,t}$ under key sk_i for time period t;
2. Compromise queries: The attacker submits i and receives the private key sk_i of user i; if $i = 0$, the attacker receives the private key of the aggregator.

Challenge. In a second phase, the attacker chooses a time period $t^\star$. Let $\mathcal{U}^\star \subseteq \{1, \ldots, n\}$ be the whole set of users for which, at the end of the game, no encryption queries have been made on time period $t^\star$ and no compromise queries have been made. The attacker chooses of a subset $\mathcal{S}^\star \subseteq \mathcal{U}^\star$ and two different series of triples

$$\langle (i, t^\star, x_{i,t^\star}^{(0)}) \rangle_{i \in \mathcal{S}^\star} \quad \text{and} \quad \langle (i, t^\star, x_{i,t^\star}^{(1)}) \rangle_{i \in \mathcal{S}^\star} ,$$

that are given to the challenger. Further, if the aggregator capability sk_0 is compromised at the end of the game and $\mathcal{S}^\star = \mathcal{U}^\star$, it is required that

$$\sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(0)} = \sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(1)} \ . \tag{1}$$

Guess. The challenger chooses at random a bit $b \in \{0,1\}$ and returns the encryption of $\langle x_{i,t^\star}^{(b)} \rangle_{i \in \mathcal{S}^\star}$ to the attacker.
More queries. The attacker can make more encryption and compromise queries.
Outcome. At the end of the game, the attacker outputs a bit b' and wins the game if and only if $b' = b$. As usual, $\mathcal{A}$'s advantage is defined to be

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) := |\Pr[b' = b] - 1/2|.$$

Remark 1. Note in the "More queries" stage that since $\mathcal{S}^\star \subseteq \mathcal{U}^\star$, the attacker cannot submit an encryption query $(i, t^\star, \cdot)$ with $i \in \mathcal{S}^\star$ or a compromise query i with $i \in \mathcal{S}^\star$.

Definition 3. *An encryption scheme is said to meet the* AO security notion *if no probabilistic polynomial-time attacker can guess correctly in the above game the bit b with a probability non-negligibly better (in the security parameter) than* $1/2$. *The probability is taken over the random coins of the game according to the distribution induced by* Setup *and over the random coins of the attacker.*

3.2 Shi *et al.*'s Scheme

In [25], Shi, Chan, Rieffel, Chow and Song consider the following encryption scheme. They show that the scheme meets the AO security notion under the DDH assumption [4], in the random oracle model.

$\mathsf{Setup}(1^\kappa)$**.** Let a group $\mathbb{G}$ of prime order q for which the DDH assumption holds, and let a random generator $g \in \mathbb{G}$. Let also a hash function $H : \mathbb{Z} \to \mathbb{G}$ viewed as a random oracle. Finally, let n random elements in $\mathbb{Z}/q\mathbb{Z}$, $s_1, \ldots, s_n$, and define $s_0 = -\sum_{i=1}^n s_i \bmod q$.
$\mathsf{param} = \{\mathbb{G}, g, H\}$; $\mathsf{sk}_i = s_i$ (for $0 \leq i \leq n$).

$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, x_{i,t})$**.** At time period t, for a private input $x_{i,t} \in \mathbb{Z}/q\mathbb{Z}$, user i produces

$$c_{i,t} = g^{x_{i,t}} H(t)^{s_i} \ .$$

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, c_{1,t}, \ldots, c_{n,t})$**.** The aggregator obtains the sum X_t for time period t by first computing $V_t := H(t)^{s_0} \prod_{i=1}^n c_{i,t} = g^{X_t}$ and next the discrete logarithm of V_t w.r.t. basis g.

Remark 2. Since g has order q, note that the so-obtained value for X_t is defined modulo q.

We see that in the AggrDec algorithm the aggregator has to compute the value of X_t from $V_t = g^{X_t}$ in $\mathbb{G}$. For known groups satisfying Shi *et al.*'s setting (*i.e.*, prime-order DDH groups), the most appropriate method is Pollard's λ algorithm (or variants thereof) and requires that the range of X_t is small. In the next section, we present a scheme where the computation of discrete logarithms can be done efficiently without such a restriction on the range, while at the same time, meeting the AO security notion.

4 New Scheme

Shi *et al.*'s scheme involves the computation of a discrete logarithm in a prime-order group for which the DDH assumption holds. We rather consider groups $\mathbb{G}$ of composite order for which there is a subgroup $\mathbb{G}_1$ [of unknown order] wherein some intractability assumption holds and another subgroup $\mathbb{G}_2$ wherein discrete logarithms are easily computable. It is crucial that the order of $\mathbb{G}_1$, $\#\mathbb{G}_1$, is only known to a trusted dealer. For anyone else (including the aggregator), $\#\mathbb{G}_1$ must remain unknown. Merely an upper bound on $\#\mathbb{G}_1$ can be derived. We present below a scheme fulfilling these requirements.

$\mathsf{Setup}(1^\kappa)$. On some input security parameter κ, the trusted dealer randomly generates a modulus $N = pq$, which is the product of two equal-size primes p, q. Note that size condition on p and q implies that $\gcd(\phi(N), N) = 1$. It also defines a hash function $H : \mathbb{Z} \to (\mathbb{Z}/N^2\mathbb{Z})^*$ that will be viewed as a random oracle in the security analysis. Letting ℓ the bit-length of N, from n randomly chosen elements in $\pm\{0,1\}^{2\ell}$, $s_1, \ldots, s_n$, it finally sets $s_0 = -\sum_{i=1}^n s_i$ and defines $\mathsf{param} = \{N, H\}$ as well as $\mathsf{sk}_i = s_i$ (for $0 \leq i \leq n$).

$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, x_{i,t})$. At time period t, for a private input $x_{i,t} \in \mathbb{Z}/N\mathbb{Z}$, user i produces

$$c_{i,t} = (1 + x_{i,t}N) \cdot H(t)^{s_i} \bmod N^2 \ .$$

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, c_{1,t}, \ldots, c_{n,t})$. The aggregator obtains the sum X_t for time period t by first computing $V_t := H(t)^{s_0} \prod_{i=1}^n c_{i,t} \bmod N^2$ and next X_t as

$$X_t = \frac{V_t - 1}{N} \ .$$

The correctness follows by observing that

$$H(t)^{s_0} \prod_{i=1}^n c_{i,t} \equiv \prod_{i=1}^n (1 + x_{i,t}N) \equiv 1 + \Big(\sum_{i=1}^n x_{i,t} \bmod N\Big)N \pmod{N^2} \ .$$

Again, observe that the value of X_t is defined modulo N. Hence, if $\sum_{i=1}^n x_{i,t} < N$, we have $X_t = \frac{V_t - 1}{N} = \sum_{i=1}^n x_{i,t}$ over the integers. The main difference with the scheme of Shi *et al.* resides in that there is no discrete logarithm to compute in a group in which the DDH assumption holds. On the contrary, the recovery of X_t from the accumulated product, V_t, is now easy. As a result, there is no longer the restriction on the size of $x_{i,t}$ or on the total number n of users, as long as $\sum_{i=1}^n x_{i,t} < N$. Typically, N is a 2048-bit value. In practice, there is therefore no restriction.

Another interesting advantage of the scheme over [25] is that it allows efficiently encrypting "on-the-fly" [14]. Namely, exponentiations $H(t)^{s_i} \bmod N^2$ can be pre-computed in such a way that, when the plaintext $x_{i,t}$ is known, the sender only has to compute a modular multiplication to get $c_{i,t}$. In applications to smart metering systems, this advantage may be crucial as computations usually take place in constrained devices.

One surprising thing about the scheme is its extreme simplicity. Indeed, somewhat unexpectedly, we do not even need to restrict the range of the random oracle H to, for example, the subgroup of quadratic residues in $(\mathbb{Z}/N^2\mathbb{Z})^*$ or the subgroup of elements with positive Jacobi symbol modulo N. If $H(t) \bmod N$ has Jacobi symbol -1 modulo N, the parity of s_i is leaked by $c_{i,t}$. However, as will be established in Section 5, this does not jeopardize the security of the scheme as long as the values $s_i \bmod N$ remain computationally hidden.

5 Security Proof

Although the scheme is very similar to the construction of Shi *et al.*, its security analysis is completely different (and actually simpler). Albeit taking place in the random oracle model [2], it bears similarities with the techniques of Cramer and Shoup [10] (see also [6]) in that it appeals to an information theoretic argument exploiting some entropy hidden in the private key.

As a result, we obtain a much tighter security reduction than in [25]. Here, the gap between the DCR distinguisher and the adversary's advantage is only proportional to the number of encryption queries. In contrast, the construction in [25] suffers from an additional degradation factor of $O(n^3)$, where n is the number of users in the system, in terms of concrete security. In contrast, our security bound is completely independent of the number of users.

Theorem 1. *The scheme provides* AO *security under the* DCR *assumption in the random oracle model. Namely, for any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a* DCR *distinguisher $\mathcal{B}$ with comparable running time and such that*

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) \leq e \cdot (q_{enc} + 1) \cdot \mathbf{Adv}^{\mathsf{DCR}}(\mathcal{B}),$$

where q_{enc} is the number of encryption queries and e is the base for the natural logarithm.

Proof. The proof proceeds with a sequence of three games. The latter begins with Game 0, which is the real game, and ends with Game 2, where even a computationally unbounded adversary has no advantage. For each $j \in \{0, 1, 2\}$, we denote by S_j the event that the challenger $\mathcal{B}$ outputs 1 in Game j. We also define $Adv_j = |\Pr[S_j] - 1/2|$.

In the sequel, we assume w.l.o.g. that the adversary $\mathcal{A}$ has always already queried the random oracle H on the input t before any encryption query for the time period t. Indeed, the challenger can always enforce this by making H-queries for itself. For simplicity, we also assume that the adversary does not query the random oracle more than once for a given t.

Game 0**:** This is the real game. Namely, the challenger performs the setup of the system by choosing $s_1, \ldots, s_n \xleftarrow{R} \pm\{0,1\}^{2\ell}$ and defining $s_0 = -\sum_{i=1}^{n} s_i$. Queries to the random oracle H are answered by returning uniformly random elements in $(\mathbb{Z}/N^2\mathbb{Z})^*$. Encryption queries $(i, t, x_{i,t})$ are answered by

returning the ciphertext $c_{i,t} = (1 + x_{i,t}N) \cdot H(t)^{s_i} \bmod N^2$. Whenever the adversary decides to corrupt some player $i \in \{0, \ldots, n\}$, the challenger reveals s_i. In the challenge phase, the adversary chooses a target time period $t^\star$, an uncorrupted subset $\mathcal{S}^\star \subseteq \mathcal{U}^\star$ and two distinct series $\langle (i, t^\star, x_{i,t^\star}^{(0)}) \rangle_{i \in \mathcal{S}^\star}$, $\langle (i, t^\star, x_{i,t^\star}^{(1)}) \rangle_{i \in \mathcal{S}^\star}$ which must satisfy Equation (1) if $\mathcal{S}^\star = \mathcal{U}^\star$ and the aggregator's private key s_0 is exposed at some point of the game (cf. § 3.1). At this stage, the challenger flips a fair binary coin $b \stackrel{R}{\leftarrow} \{0, 1\}$ and the adversary $\mathcal{A}$ receives

$$\left\{ c_{i,t^\star} = (1 + x_{i,t^\star}^{(b)} N) \cdot H(t^\star)^{s_i} \bmod N^2 \right\}_{i \in \mathcal{S}^\star} .$$

We assume that the adversary queries $H(t^\star)$ before the challenge phase. Otherwise, $\mathcal{B}$ can simply make the query for itself. In the second phase, after a second series of queries, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. We let the challenger $\mathcal{B}$ output 1 if $b' = b$ and 0 otherwise. The adversary's advantage in Game 0 is thus $Adv_0 = |\Pr[S_0] - 1/2| = \mathbf{Adv}^{\mathsf{AO}}(\mathcal{A})$.

Game 1: This game is identical to Game 0 with the following difference. For each random oracle query t, the challenger $\mathcal{B}$ flips a biased coin $\delta_t \in \{0, 1\}$ that takes the value 1 with probability $1/(q_{enc} + 1)$ and the value 0 with probability $q_{enc}/(q_{enc} + 1)$. At the end of the game, $\mathcal{B}$ considers the event E that either of the following conditions holds:

- For the target time period $t^\star$, the coin $\delta_{t^\star}$ flipped for the hash query $H(t^\star)$ was $\delta_{t^\star} = 0$.
- There exists a time period $t \neq t^\star$ such that an encryption query $(i, t, .)$ was made for some user in $i \in \mathcal{U}^\star$ but for which $\delta_t = 1$.

If event E occurs (which $\mathcal{B}$ can detect at the end of the game), $\mathcal{B}$ halts and outputs a random bit. Otherwise, it outputs 1 if and only if $b' = b$. The same analysis as that of Coron [9] shows that $\Pr[\neg E] = 1/e(q_{enc} + 1)$, where e is the base for the natural logarithm. The transition from Game 0 to Game 1 is thus a transition based on a failure event of large probability [11] and we thus have $Adv_1 = Adv_0 \cdot \Pr[\neg E] = Adv_0/e(q_{enc} + 1)$.

Game 2: In this game, we modify the distribution of random oracle outputs. Specifically, the treatment of each hash query t depends on the random coin $\delta_t \in \{0, 1\}$.

- If $\delta_t = 0$, the challenger $\mathcal{B}$ chooses a random N-th residue $z_t = {r_t}^N \bmod N^2$, with $r_t \stackrel{R}{\leftarrow} (\mathbb{Z}/N\mathbb{Z})^*$, and defines $H(t) = z_t$. Note that the resulting hash value $H(t)$ is now a N-th residue in $(\mathbb{Z}/N^2\mathbb{Z})^*$.
- If $\delta_t = 1$, $\mathcal{B}$ chooses a uniformly random $r_t \in (\mathbb{Z}/N^2\mathbb{Z})^*$ and programs H so as to have $H(t) = r_t$.

Lemma 1 below shows that Game 2 and Game 1 are computationally indistinguishable if the DCR assumption holds. It follows that

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{\mathsf{DCR}}(\mathcal{B}) .$$

In Game 2, we claim that $\Pr[S_2] = 1/2$ so that $\mathcal{A}$ has no advantage. Indeed, with probability $1/e(q_{enc}+1)$, we have $H(t^\star) \in_R (\mathbb{Z}/N^2\mathbb{Z})^*$ for the target time period

$t^\star$ (for which the challenge ciphertexts $\{c_{i,t^\star} = (1+N)^{x_{i,t^\star}} \cdot H(t^\star)^{s_i} \bmod N^2\}_{i\in\mathcal{S}^\star}$ are generated) whereas, for each $t \neq t^\star$, the hash value $H(t)$ is uniform in the subgroup of N-th residues. For each $t \neq t^\star$, $H(t)$ thus lives in a subgroup whose order is co-prime with N since $\gcd(N, \phi(N)) = 1$. By the Chinese Remainder Theorem, for each $i \in \mathcal{S}^\star$, the value $H(t)^{s_i}$ is completely independent of $s_i \bmod N$ when $t \neq t^\star$. In other words, encryption queries for periods $t \neq t^\star$ leak no information about $\{s_i \bmod N\}_{i\in\mathcal{S}^\star}$. At the same time, for period $t^\star$, $\{H(t^\star)^{s_i}\}_{i\in\mathcal{S}^\star}$ contain random components of order N which only appear in the challenge ciphertexts $\{c_{i,t^\star} = (1+N)^{x^{(b)}_{i,t^\star}} \cdot H(t^\star)^{s_i} \bmod N^2\}_{i\in\mathcal{S}^\star}$, where they completely blind $\{x^{(b)}_{i,t^\star}\}_{i\in\mathcal{S}^\star}$.

More precisely, let us consider what a computationally unbounded adversary $\mathcal{A}$ can see. We have the following two mutually exclusive situations.

- If $\mathcal{S}^\star \subsetneq \mathcal{U}^\star$ or $\mathcal{A}$ did not compromise the aggregator, then $\mathcal{A}$ has no information about $\{s_i \bmod N\}_{i\in\mathcal{S}^\star}$ whatsoever. Even the sum $\sum_{i\in\mathcal{S}^\star} s_i \bmod N$ remains hidden because there exists $\theta \in (\mathcal{U}^\star \backslash \{\mathcal{S}^\star\}) \cup \{0\}$ such that $s_\theta \bmod N$ is completely independent of $\mathcal{A}$'s view. If we consider individual ciphertexts $\{c_{i,t^\star}\}_{i\in\mathcal{S}^\star}$ and their partial aggregation

$$c_{\mathcal{S}^\star,t^\star} = \prod_{i\in\mathcal{S}^\star} c_{i,t^\star} \bmod N^2\,, \tag{2}$$

  for each value of $b \in \{0,1\}$, there exist $\{s_i \bmod N\}_{i\in\mathcal{S}^\star}$ that explain the adversary's view.
- If $\mathcal{S}^\star = \mathcal{U}^\star$ and $\mathcal{A}$ has obtained s_0 by corrupting the aggregator, we must have $\sum_{i\in\mathcal{S}^\star} x^{(0)}_{i,t^\star} = \sum_{i\in\mathcal{S}^\star} x^{(1)}_{i,t^\star}$. Since $\sum_{i=0}^n s_i = 0$, the only information that $\mathcal{A}$ obtains about $\{s_i \bmod N\}_{i\in\mathcal{S}^\star}$ is the sum $\sum_{i\in\mathcal{S}^\star} s_i \bmod N$, which an all-powerful adversary can infer via corruption queries or encryption queries during period $t^\star$. However, if consider the partially aggregated ciphertext (Eq. (2)), then $c^{\phi(N)}_{\mathcal{S}^\star,t^\star} \bmod N^2$ does not depend on the bit $b \in \{0,1\}$.

In Game 2, we can only have $b' = b$ with probability $1/2$. Putting all together, we therefore find

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) \leq e \cdot (q_{enc} + 1) \cdot \mathbf{Adv}^{\mathsf{DCR}}(\mathcal{B})\,,$$

which concludes the proof. □

Lemma 1. *Under the* DCR *assumption, Game* 2 *is computationally indistinguishable from Game* 1*.*

Proof. The proof is by contradiction and builds a DCR distinguisher $\mathcal{B}$ from an adversary $\mathcal{A}$ that has noticeably different behaviors in Game 1 and Game 2.

The reduction $\mathcal{B}$ receives as input a pair (N, z) and has to decide whether $z = r^N \bmod N^2$, for some $r \in (\mathbb{Z}/N\mathbb{Z})^*$, or $z \in_R (\mathbb{Z}/N^2\mathbb{Z})^*$. To this end, $\mathcal{B}$ begins by picking $s_1, \ldots, s_n \xleftarrow{R} \pm\{0,1\}^{2\ell}$ and sets $s_0 = -\sum_{i=1}^n s_i$, exactly as the challenger of Game 1 does. Throughout the game, $\mathcal{B}$ always answers encryption queries and

corruption queries faithfully. However, the treatment of random oracle queries $H(t)$ depends on the value of the biased coin $\delta_t \in \{0,1\}$. Namely, when $\delta_t = 0$, $\mathcal{B}$ uses the random self-reducibility of DCR and builds many instances $\{z_t\}_t$ for the same N out of z.

- If $\delta_t = 0$, $\mathcal{B}$ chooses $\alpha_t \stackrel{R}{\leftarrow} \mathbb{Z}/N\mathbb{Z}$, $\beta_t \stackrel{R}{\leftarrow} (\mathbb{Z}/N\mathbb{Z})^*$ and computes $z_t = z^{\alpha_t} \cdot {\beta_t}^N \bmod N^2$. Observe that, if z is a N-th residue (resp. a random element of $(\mathbb{Z}/N^2\mathbb{Z})^*$), z_t is uniformly distributed in the subgroup of N-th residues modulo N^2 (resp. uniformly distributed in $(\mathbb{Z}/N^2\mathbb{Z})^*$). Then, $\mathcal{B}$ programs the random oracle H to have $H(t) = z_t$.
- If $\delta_t = 1$, $\mathcal{B}$ draws $z_t \stackrel{R}{\leftarrow} (\mathbb{Z}/N^2\mathbb{Z})^*$ instead of choosing it among N-th residues. It defines $H(t) = z_t$.

When $\mathcal{A}$ terminates, $\mathcal{B}$ outputs a random bit if event E has come about during the game. Otherwise, $\mathcal{B}$ outputs 1 if $b' = b$ and 0 otherwise.

Clearly, if $z \in_R (\mathbb{Z}/N^2\mathbb{Z})^*$, $\mathcal{A}$'s view is exactly the same as in Game 1. In contrast, if z is a N-th residue, $\mathcal{B}$ is rather playing Game 2 with the adversary. □

6 Concluding Remarks

This paper presented a new scheme allowing an untrusted aggregator to evaluate the sum of user's private inputs. In contrast to prior solutions, there is no restriction on the message space or on the number of users. This results in always fast decryption and aggregation, even over large plaintext spaces and/or population of users.

Our scheme provides many other advantages over prior solutions. One of these is a much better concrete security in the random oracle model as our bound is completely independent of the number n of users. In comparison, the security proof in [25] entails a multiplicative gap proportional to n^3 in the reduction from the DDH assumption. Considering that large values of n such as $n \approx 2^{20}$ are expectable in practical applications, it seems advisable to increase the key size accordingly. In our setting, our reduction decreases the security loss to only 30 bits if we allow for $q_{enc} = 2^{30}$, as recommended in the literature [3] (note that, in this setting, this value should be seen as a rather theoretical bound since, using a counter, one can always bound q_{enc} to arbitrarily small values like, e.g., $q_{enc} \leq 3$).[2]

Finally, our scheme makes it possible to encrypt data in off-line/on-line mode through the use of coupons: once the message is known, the user only has to evaluate a single modular multiplication. It is therefore well-suited to low-power devices that are typically used in applications like smart metering or sensor networks.

[2] The proof in [25] additionally features a degradation factor of q_H, where q_H is the number of random oracle queries. We suspect that it can be reduced to q_{enc} using Coron's technique [9]. The main limiting factor in [25] is thus the cubic dependence on n.

References

1. Ács, G., Castelluccia, C.: I have a dream! (differentially private smart metering). In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 118–132. Springer, Heidelberg (2011)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., et al. (eds.) 1st ACM Conference on Computer and Communications Security, pp. 399–416. ACM Press (1993)
3. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
4. Boneh, D.: The decision Diffie-Hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
6. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
7. Castelluccia, C., Chan, A.C.-F., Mykletun, E., Tsudik, G.: Efficient and provably secure aggregation of encrypted data in wireless sensor networks. ACM Transactions on Sensor Networks 5(3), Article 20 (2009)
8. Hubert Chan, T.-H., Shi, E., Song, D.: Privacy-preserving stream aggregation with fault tolerance. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 200–214. Springer, Heidelberg (2012)
9. Coron, J.-S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
10. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
11. Dent, A.W.: A note on game-hopping proofs. Cryptology ePrint Archive: Report 2006/260 (2006)
12. Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
13. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006)
14. Even, S., Goldreich, O., Micali, S.: On-line/off-line digital schemes. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 263–275. Springer, Heidelberg (1990)
15. Garcia, F.D., Jacobs, B.: Privacy-friendly energy-metering via homomorphic encryption. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 226–238. Springer, Heidelberg (2011)
16. Icart, T.: How to hash into elliptic curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Heidelberg (2009)
17. Jawurek, M., Kerschbaum, F.: Fault-tolerant privacy-preserving statistics. In: Fischer-Hübner, S., Wright, M. (eds.) PETS 2012. LNCS, vol. 7384, pp. 221–238. Springer, Heidelberg (2012)
18. Jawurek, M., Kerschbaum, F., Danezis, G.: Privacy technologies for smart grids – A survey of options. Technical Report MSR-TR-2012-119, Microsoft Research (November 2012)

19. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-friendly aggregation for the smart-grid. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (2011)
20. Li, F., Luo, B., Liu, P.: Secure information aggregation for smart grids using homomorphic encryption. In: 2010 First IEEE International Conference on Smart Grid Communications, SmartGridComm (2010)
21. Lin, H.-Y., Tzeng, W.-G., Shen, S.-T., Lin, B.-S.P.: A practical smart metering system supporting privacy preserving billing and load monitoring. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 544–560. Springer, Heidelberg (2012)
22. McCurley, K.S.: A key distribution system equivalent to factoring. Journal of Cryptology 1(2), 95–105 (1988)
23. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
24. Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: Elmagarmid, A.K., Agrawal, D. (eds.) 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD 2010), pp. 735–746. ACM Press (2010)
25. Shi, E., Hubert Chan, T.-H., Rieffel, E.G., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: Network and Distributed System Security Symposium (NDSS 2011). The Internet Society (2011)
26. Shmuely, Z.: Composite Diffie-Hellman public key generating systems hard to break. Technical Report 356, Israel Institute of Technology, Computer Science Department, Technion (February 1985)

"Give Me Letters 2, 3 and 6!": Partial Password Implementations and Attacks

David Aspinall[1] and Mike Just[2]

[1] University of Edinburgh
david.aspinall@ed.ac.uk
[2] Glasgow Caledonian University
mike.just@gcu.ac.uk

Abstract. A *partial password* is a query of a subset of characters from a full password, posed as a challenge such as *"Give me letters 2, 3 and 6 from your password"*. Partial passwords are commonly used in the consumer financial sector, both online and in telephone banking. They provide a cheap way of providing a varying challenge that prevents eavesdroppers or intermediate systems learning a shared secret in a single step. Yet, despite widespread adoption among millions of consumers, this mechanism has had little attention in the academic literature. Answers to obvious questions are not clear, for example, how many observations are needed for an attacker to learn the complete password, or to successfully answer the next challenge? In this paper we survey a number of online banking implementations of partial passwords, and investigate the security of the mechanism. In particular, we look at *guessing attacks* with a *projection dictionary* ranked by likelihood, and *recording attacks* which use previous information collected by an attacker. The combination of these techniques yields the best attack on partial passwords.

Keywords: passwords, PINs, dictionary attack, recording attack, bank security

1 Introduction

A *partial password* is a query of a subset of characters from a full password. The mechanism is widely adopted in the financial sector; it is particularly popular in consumer online banking in the UK [1]. As far as we can tell, the idea spread from telephone banking, where it was invented to prevent an operator seeing a customer's complete password. It is now widely used in online banking web applications and by some instances of the 3DSecure system employed by Visa and Mastercard. Online, the main benefit of the mechanism is that it is a cheap way to impede attackers who can observe password entry by shoulder surfing, key logging or browser malware. This is because it allows a time-varying challenge that does not reveal complete information in a single step.

Despite widespread adoption among millions of consumers, this mechanism seems to have received only cursory attention in the academic literature so far.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 126–143, 2013.

There are some obvious questions that remain unanswered, such as: what level of security is provided by this scheme? How many observations does an attacker need to learn the complete password or the correct response to the next challenge? Is it safe to use weak passwords? What are good choices for parameters such as the challenge size or the schedule of challenges issued? Is there already an industry consensus on preferred range of parameters?

To address these questions, we studied the security of the protocol design and also examined (externally, as users) 15 different implementations across 4 countries. Most of the implementations we examined were UK-based; we believe the protocol is more commonly used in the UK than other countries at present.

Security is assessed by measuring the difficulty of attacks. Password attacks may occur *online* or *offline*. The difficulty of offline attacks, where a password database is stolen, will depend on exactly what is stored in the database. To support the partial protocol the implementation will need to either store plaintext for the password, or devise a mechanism for performing one-way checks on all combinations that might be queried (which can be a large number for long passwords). We don't investigate this attack mode here.

Our focus is on online attacks. These may be either *targeted* against a particular user, or *trawling*, working through as many accounts as possible to break a percentage of them. Trawling takes statistical advantage of large numbers of users to circumvent the rate-limiting that applies on individual accounts. There are two basic techniques the attacker can use: he can *guess* responses using background information (e.g., a dictionary) or he can *record* previous observations of the protocol and try to replay them appropriately. These techniques can be combined: recording can be used to reduce the guesses needed at each stage. In either case, the goal is to break into accounts by either responding correctly to the *next challenge* issued in the protocol, or learning the *complete password* to answer all future challenges.

Contributions and highlights. To the best of our knowledge, this is the first time a detailed study of the partial password protocol has been given. Our main contributions are: (1) a survey of current implementations of partial passwords in online banking; (2) a framework for analysing and comparing these in terms of parameter choices; (3) investigation of a family of attacks based on guessing and recording, and an analysis of their success rates.

A short partial challenge is obviously less secure than a challenge to produce the full password, even guessing randomly without background information. To see what happens with background data, we measure guessing against subsets of the widely analysed RockYou leaked password set, coalescing entries for each challenge to form what we call a *projection dictionary*. We find that with 6 guesses, an attacker can respond correctly to 2-place challenges on 6-digit PINs with a success rate of 30%. Recording up to 4 runs, an attacker can succeed over 60% of the time, or by combining guessing and recording, over 90%. Alphanumeric passwords do somewhat better: responding to 3-place challenges on 8-character alphanumeric passwords, with up to 10 guesses, the attacker can

achieve a success rate of 5.5%. Combining guessing and recording increases that to 25% with one recorded run and at least 80% with four runs.

Related work. Human identification methods attempt to mitigate against an observer by using time-varying challenges to prevent replay, and by concealing the shared secret by requiring the user to do some manipulations [2,3]. These schemes are reminiscent of partial passwords but require the user to do much more work, going well beyond what would be comfortable for most mainstream users, to achieve probabilistic security guarantees [4]. The actual mechanisms in use by industry have not been widely examined in the open literature. One paper that has examined it is the study of Goring et al [5], which considered a particular attack enabled by hardware key-loggers that build up information about responses but not challenges; our recording attacks consider more powerful attackers who can see both challenge and response. This incremental learning about a secret has similarity to PIN guessing through faulty APIs in which an attacker is able to query the API to gradually learn different PIN positions [6,7] although in the case of partial passwords, active attacks are not required. Instead, an observer can use partial information, either to answer the next challenge directly, or in combination with information about likely password distributions to make very good guesses. To show this, we adapt recent work on guessing attacks [8,9] and analysing large leaked password sets [10,11,12] to partial passwords.

Overview. The remainder of the paper is organised as follows. In Section 2 we survey 15 implementations of partial passwords, to find the parameters they use. Broadly, these split into PINs and alphanumeric passwords; we choose two typical cases for further investigation. In Section 3 we consider guessing attacks, building up the capability of the attacker: we go from random guessing, through attacks that use letter frequencies and finally to dictionaries, in particular, a *projection dictionary* attack that guesses responses based upon projected characters from likely password dictionaries. In Section 4 we analyse recording attacks, where an adversary has the capability to observe previous runs. We show how the information learned can be used to respond exactly to a fraction of next challenges and how quickly the whole password is eventually (almost certainly) discovered. We then combine recording with guessing to achieve our optimal attacks. Section 5 concludes and discusses the different attacks as well as some recommendations.

2 System Model and Survey

Fig. 1 shows some partial password challenges from real implementations. These show two different choices in the format for presenting the challenge, and for filling in the response. The overall protocol works as follows:

Registration. In the initialisation phase, the user chooses a password. Often, this is a short (e.g., 6–10 characters) password taken from a restricted alphabet (e.g., lower case letters and numbers only), and without rules enforcing

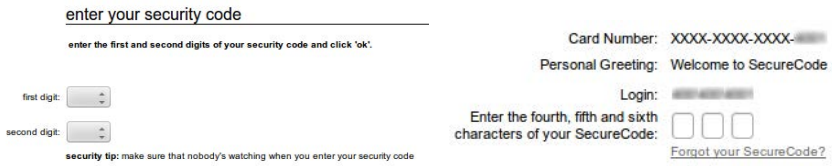


Fig. 1. Partial password implementations

special password formats, so weak (but memorable) passwords are allowed. In several implementations, short PINs are used.

Login. The user is presented with a challenge such as: *Please provide letters 2, 3 and 6 from your password*. He or she responds by projecting out the requested positions.
Here is an explicit example:

Positions:	1	2	3	4	5	6	7
User password:	a	s	h	u	f	1	0
Correct response:		s	h			1	

From our anecdotal discussions, some users perform the projection mentally, some count off letters using their fingers; others write the password down as above. If the correct response is given, the user has completed partial password authentication.

Retry. If the response is not correct, the user is challenged again, perhaps using the same challenge or perhaps changing the challenge. We observed both cases, and rate limits being enforced to restrict the number of attempts before re-registration is required.

Next time. On the next login, the user is presented with a new challenge.

This process achieves the basic requirement of not revealing the password to an observer in a single step. In almost all cases we've seen, the partial password is used in addition to another credential (e.g., a full password) in a *multi-stage authentication*, where both credentials must be entered correctly to proceed.

2.1 Survey

We conducted a survey of online web banking implementations in four countries: Canada, Germany, Ireland, and the UK. We investigated the credentials they use for authentication (password, PINs, hardware token, etc.), and the way they check those credentials (partial queries, sequentially or together, etc.). We used several collection methods, including publicly available demos and web pages (see Appendix A.1) and direct access with personal accounts. Most of the implementations were in the UK, as we were able to generate accounts there to try them out; moreover we believe that the UK is the biggest user of this form of authentication at present. The data we collected is necessarily a snapshot; banking implementations are updated periodically.

For partial password and partial PIN checks, there are several parameters which vary between implementations, summarised in Fig. 2. The parameters of interest are:

- The **password format**. This constrains the set of allowed passwords. We suppose that the password consists of n characters chosen from an N character set. For partial passwords, this set is often deliberately small, in particular typically not including non-alphanumeric symbols and not distinguishing upper and lower case letters. This may be for better usability, e.g., by reducing the size of pull-down menus for implementations that use them.
- The **challenge format**. The challenge asks for characters in m unique password positions, where $1 \leq m \leq n$. In all but one case we surveyed, the positions are queried in ascending order. This reduces the number of possible challenges but, we suppose, it may be easier for the user to step through the letters in order from left to right rather than hop around.
- The **number of guesses**. If the user responds incorrectly to a challenge, they may be allowed more attempts; we will use β to stand for the maximum number of tries allowed before the account is locked in some manner. This is also the number of guesses an online attacker may make.

We gathered values for β but they are not shown in the table as they are difficult to corroborate without owning an account with the bank in question, and even then can vary in unexpected ways from our "black box" view. Smaller limits give better security, but inconvenience the user: it has been suggested that ten attempts at password entry should be allowed to provide good usability [13]. It seems reasonable that this limit would also be applied to partial passwords, and we found some implementations do use β=10. For PINs there may be concerns about allowing too many guesses because of the smaller answer space; implementations of protocols behind ATMs typically use β=6 as an upper limit [9].

In Fig. 2 we have listed the second credential used by each bank, whether a full password, full PIN, response to a challenge question, or multi-purpose information such as a credit card number and date of birth. In several cases, the banks also offer alternatives to their customers (e.g., tokens and card readers), though the option to not use these remains. The failure behaviours differed considerably between implementations, e.g., what kind of feedback was provided when the user made a mistake, and whether failure required re-entry of other credentials. These choices affect the overall security of the system so that, for example, for a bank that uses a password as a second credential, a guessing attack would need to compromise both the full and the partial password. Some work has been undertaken to investigate the overall security in this case [14], but in this paper we focus on the security of the partial password stage.

There is another system parameter not listed in the table: the system's schedule of issued challenges. Without knowledge of the system implementation, we can only view the challenge schedule externally and make assumptions about the sequence. A plausible working assumption that fits with our observations is that challenges are chosen uniformly at random from the $\binom{n}{m}$ possible challenges.

Bank (UK based unless noted)	character set size, N	password length, n	challenge size, m	second credential
ING DiBa (Germany)	10	6	2	PIN
Cooperative	10	4	2	question
Tesco	10	6	2	password
Smile	10	6	2	question
Nationwide	10	6	3	password
AIB	10	5	3	question
Bank of Ireland (Ireland)	10	6	3	date of birth
Nat West, step 1	10	4	2	(see next row)
Nat West, step 2	36	6–20	3	(see prev row)
HBoS	36	6–15	3	password
3DSecure, B. of Ireland (UK)	36	8–15	3	credit card #
Standard Life	36	8–10	3	none
Skipton	36	8–30	3	question
First Direct	36	6–30	3	question
Barclays	52	6–8	2	PIN
HSBC (Canada)	62	8	3	question

Fig. 2. Survey of partial password parameters (as of 25 Sept 2012)

The survey motivates two typical parameter cases which we use for empirical study: PINs with N=10, n=6, m=2 and β=6 ($\binom{n}{m} = 15$) and alphanumeric passwords with N=36, n=8, m=3 and β=10 ($\binom{n}{m} = 56$).

3 Guessing Attacks

In a guessing attack, the attacker attempts to answer the next challenge, or find the whole password, by selecting from a set of possible answers, possibly using background data such as a dictionary. The plain *brute force* attack uses only knowledge of the response alphabet. Assuming a uniform distribution, the probability of guessing a randomly chosen user's complete password is $\frac{1}{N^n}$, which is 10^{-6} for the PIN case and approximately 10^{-12} for the alphanumeric. The probability of guessing the next challenge is $\frac{1}{N^m}$, so $\frac{1}{100}$ for PINs and approximately 2×10^{-5} for alphanumeric; already a reduced baseline.

The attacker usually can make more than one guess, however. The *β-success rate* [8,15] is a useful measure of the effectiveness of online attacks: it describes the (maximal) proportion of a data set that can be covered by a fixed number β of guesses. A *trawling attack* repeats guessing against many accounts; assuming these are selected randomly from the same distribution without replacement, we expect to break a fraction of them given by the success rate.

For a uniformly distributed password set we would expect to break the proportion $\frac{\beta}{N^n}$ of accounts by guessing whole passwords, or $\frac{\beta}{N^m}$ guessing partial challenges. A system security designer wants to keep the β-success acceptably low. With brute force attacks, the β-success rates goes up to 1 in 5000 for guessing the next challenge in the alphanumeric case (β=10) and 6% for PINs (β=6).

Guessing non-uniformly can change things dramatically. Real passwords are often chosen from common names, dictionary words and their variations; certain PINs are also very common, chosen from dates, or simple numeric or keypad-layout sequences. Dictionary attacks exploit this; their success depends on their coverage. For offline (or unbounded online) attacks, enormous cracking dictionaries are pre-calculated and stored [16] or generated on-the-fly from lists of words and rules for making variations by mutation, or by approximate methods [17]. For online attacks where rate-limiting applies, the attacker benefits from word ranking, so he can guess the most likely (i.e., most commonly chosen) password first, then the second most likely, and so on, up to the rate limit.

3.1 Letter Position Frequency Attack

We can improve on brute force by using letter frequencies to choose guesses. Compared with storing or generating a large dictionary, attack code with letter frequencies can be small and efficient, thus easily run in many places or on small devices. The attack could simply use overall frequency of letters, but the partial password challenges give us position information too. So we can exploit calculated letter-position frequencies to skew choices when guessing letters.

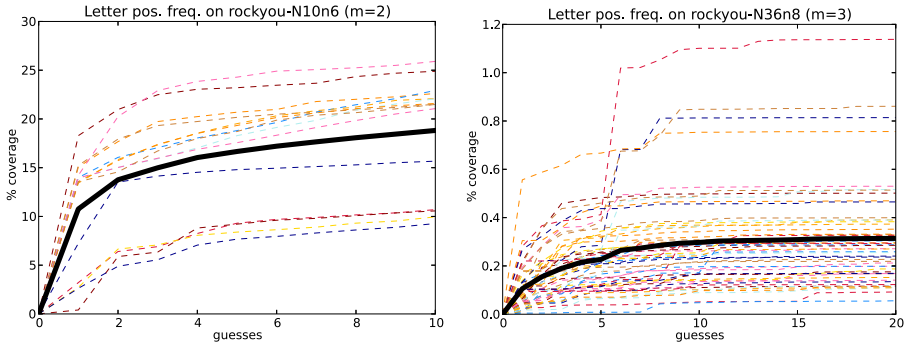


Fig. 3. Position frequency attack for RockYou PINs and passwords

It's easy to generate tables of relative letter frequencies from a dictionary. For example, in RockYou (a large set of leaked passwords described further below in Sect. 3.3), while the letter 'a' occurs 8% of the time in 8-character alphanumeric passwords, in position 2 it occurs 18% of the time. Similarly, for 6 digit PINs taken from RockYou, the digit 1 occurs on average 17% of the time, but almost 40% in position 1. (see Appendix A.2 for pictures that illustrate these numbers.) To attack using these relative frequencies we can guess characters randomly but in proportion to their frequency of occurrence by position, or, more simplistically, in a fixed order guessing the most common letters in each position in turn. For example, in the RockYou case above, the first guess for position 2 in any challenge will always be 'a'. The ideal success rate of this latter strategy is illustrated by

Fig. 3, where we measure the attack against the same sets used to generate the frequencies, and show the proportion that are broken for increasing numbers of guesses β. Each dotted line on the plot indicates success rates for a different challenge; the bold line indicates the success rates averaged over all challenges. For PINs, after 6 guesses we get an average success rate of 17% whereas for passwords, we get an average of 0.3% after 10 guesses. Because we assume that challenges are issued uniformly at random for different accounts, we take these average figures as the overall success rates.

This attack is obviously poor because it ignores correlations between letter positions; if we choose position 2 to be 'a', this changes the distributions of letters that may appear in other positions. We can improve it by using a dictionary that contains the most common answers to the projected positions.

3.2 Projection Dictionary Attacks

Using a dictionary D during the attack we could draw words at random; if the dictionary matches the target set of passwords, the success rate would be $\frac{\beta}{|D|}$. But for guessing an answer to the next challenge, there is a better strategy. Because many words may share the same projections onto the challenged positions, some responses are more frequent than others. We can pre-compute for this attack, by taking an ordinary dictionary and building a *projection dictionary* that contains combinations of positions from words and ranks the results by frequency.

For example, taking a 11,660 word dictionary of 8-letter English words[1], we find there are 2,736 possible answers to the positions 2, 3 and 6 (about 16% of the possible $26^3 = 17,576$) and 1,793 answers to challenges of the first three positions (only 10% of all possible). The projection dictionary tables give the top β guesses for each challenge, ranking them by number of occurrences, and giving the cumulative fraction covered so far:

Challenge 2 3 6:				Challenge 1 2 3:			
1.	r a i	79	0.69	1.	c o n	116	1.00
2.	r e i	77	1.34	2.	d i s	88	1.75
3.	r o i	63	1.88	3.	p r o	83	2.46
4.	l o i	59	2.38	4.	o v e	75	3.11
5.	l a i	57	2.87	5.	p r e	74	3.74

So, for example, for the challenge 1 2 3, the prefix "ove" occurs in 75 words, and the first four answers cover 3.11% of the set. The β-success rate is given by the cumulative coverage at position β. For β=10 and m=3 we get rates of 5.1% and 6.3% for the two challenges shown; these are close, but there wider variations: the most successful (weakest) case is 6 7 8 which achieves 30% after 10 guesses; this is skewed by the very common ending "ing".

Fig. 4 shows success rates of the projection dictionary attack for larger numbers of guesses. By comparison, a brute force attack would take up to $N^m = 17,576$ guesses; an ordinary (unordered) dictionary-based attack is also shown on

[1] More precisely: `/usr/share/dict/words` on Ubuntu 12.04, converted to lower case.

the graph as a diagonal line, this does better than brute force (omitting responses that do not occur), but not as well as the ordered attack using projections.

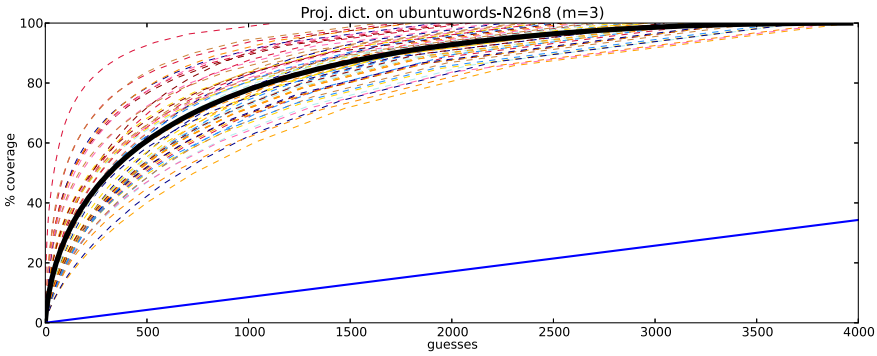


Fig. 4. Projection dictionary attack for an English dictionary

3.3 Projection Dictionaries from Password Distributions

We could use the English projection dictionary, but a better attack is possible by starting with a dictionary that more closely matches the likely target distribution of passwords, such as a real leaked password list. Many leaked password sets have been analysed by researchers, to measure the success rates of attacks that use them [10,12] and also by practitioners, to build better cracking tools [18]. The RockYou data set (leaked in 2009) is one of the most useful, with over 32 million entries. Although it is captured from a social gaming website and one might hope that users choose better passwords to secure their bank accounts than for online gaming, there is some evidence that passwords used to secure financial information are not necessarily better chosen [15]. Moreover, some bank implementations actually encourage weak choices for partial passwords by not enforcing any password composition rules and by restricting the character sets available (presumably for usability reasons).

If we take the 8-letter alphanumeric passwords from the leaked RockYou data set, including frequency counts for the words, we get a ranked dictionary:

RockYou password frequencies (N=36, n=8):

1. password	59462	1.01
2. iloveyou	49952	1.85
3. princess	33291	2.41
4. 12345678	20553	2.76
5. babygirl	15163	3.02

again, showing frequencies and cumulative percentage coverage of the whole set. There are about 5.9m 8-character alphanumeric passwords in RockYou, with 2.5m distinct, only 0.00009% of the 36^8 possible combinations; the top ten covers 3.88% of the whole set. For PINs, there are about 2.3m 6-digit passwords

appearing in RockYou, but only 390,000 are distinct. The top 6 choices cover 15.3% of the whole set, although this is very skewed by the top choice "123456" (12.8%). In trawling attacks on data sets sharing the same distributions this means 1 in 26 alphanumeric passwords could be broken by making the top 10 guesses, and 1 in 7 PINs could be broken within 6 guesses.[2]

Responding to a partial password challenge we can do even better. We build a projection dictionary taking frequencies into account; by construction, this can only improve the success rate. For example, for alphanumeric passwords:

Challenge 2 3 6:				Challenge 1 2 3:			
1.	a s o	64819	1.10	1.	i l o	76508	1.29
2.	l o y	52074	1.98	2.	p a s	66758	2.42
3.	r i e	47833	2.79	3.	m a r	58058	3.40
4.	2 3 6	24857	3.21	4.	b a b	52785	4.30
5.	a r e	21192	3.56	5.	p r i	46565	5.08

The effect of combining words to rank projections is apparent by comparing this with the previous table. For example, the most frequently occurring 8-letter password in RockYou is "password" but responding to a challenge on the first three positions, an attacker will succeed slightly more often with the letters i l o, the projection on the second most common word "iloveyou". This is because there are more variations that share those first three letters. (See Appendix A.3 for some examples with PINs.)

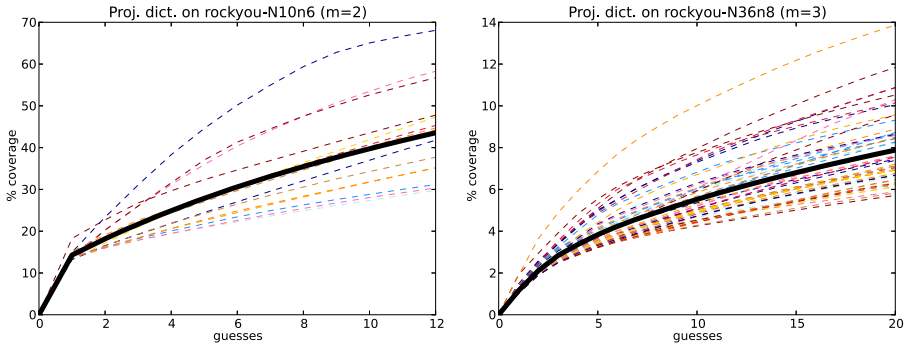


Fig. 5. Projection dictionary attacks for RockYou PINs and passwords

Fig. 5 shows the success rates of the projection dictionary attack on RockYou PINs and passwords themselves for smaller numbers of guesses. For PINs, the rates at our limit point β=6 vary from 22% to 50% with an average of 30.6%; for passwords at β=10, the rates vary from 4.2% to 10%, with the average of 5.5%. These are our best success rates for guessing the answer to the next challenge.

[2] In [9] the success rate for guessing 4 digit PINs with 6 guesses is given as 12.29%, based on almost 1.8m PINs assembled from *all* 4 digit sequences appearing anywhere in a password; the RockYou set only contains 20,661 4-digit passwords.

4 Recording Attacks

The attacks considered in the last section do not require the attacker to eavesdrop. But if an attacker can record previous challenges and responses, he will be able to gradually learn the password, as well as make better informed guesses.

4.1 Pure Recording Attacks

A *recording attack* consists of observing k previous protocol runs and using the recorded challenge-response pairings to either learn the full password, or to respond to a new challenge. And once $k > 1$ runs have been recorded, an attacker can respond to an increasing number of challenges. For example, after recording the responses corresponding to the challenges 1 3 5 and 2 4 5, new challenges such as 1 2 4 can be correctly answered.

How quickly are positions learned? The following recurrence defines the probability $p_n^m(i,k)$ of observing exactly i different positions after k runs:

$$p_n^m(i,k) = \begin{cases} \frac{1}{\binom{n}{m}} \sum_{j=0}^{m} \binom{i-j}{m-j}\binom{n-(i-j)}{j} p_n^m(i-j,k-1) & m \le i \le n,\ k \ge 1 \\ 1 & i = 0,\ k = 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

We can use Equation 1 to recursively compute a probability for run k as a function of the number of positions observed at run $k-1$, by summing over the number of fresh positions j that are learned in the kth run and considering how the remaining $m-j$ positions may be chosen from positions already observed.

Fig. 6 (left) shows the probability $p_n^m(n,k)$ of observing all n positions after k runs for different choices of n and m. For both our PIN (n=6, m=2) and alphanumeric cases (n=8, m=3), after k=6 recorded runs an attacker has a greater than 50% probability of learning the full password. This is purely recording with no guessing, so it is unaffected by the N or β parameters.

In terms of guessing the next challenge after recording k runs, if the attacker knows $m \le i \le n$ password positions, then the proportion of challenges to which they can immediately respond can be computed as a fraction $s_n^m(i)$. We can then compute the proportion of challenges learned after k runs as $\overline{s_n^m}(k)$.

$$s_n^m(i) = \frac{\binom{i}{m}}{\binom{n}{m}} \qquad \overline{s_n^m}(k) = \sum_{i=m}^{n} p_n^m(i,k)\, s_n^m(i) \quad (2)$$

The results are displayed for different choices of n and m in Fig. 6 (right). For both our PIN and alphanumeric cases, after k=4 recorded runs an attacker has a greater than 50% probability of learning the next challenge. More generally, 50% of challenges can be answered after $3 \le k \le 7$ for password lengths of $8 \le n \le 11$, and common choices of m=2 or m=3.

The attacker may be able to find out the password, or answer the next challenge, quicker still than this, by using partial recording information and then making guesses on unknown positions.

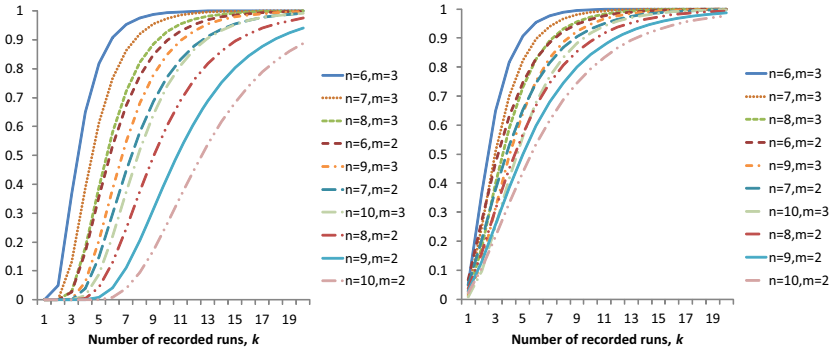


Fig. 6. Probability $p_n^m(n,k)$ of learning password (L), or $\overline{s_n^m}(k)$ next challenge (R)

4.2 Recording Plus Guessing Attacks

Let $0 \leq m' \leq m$ be the number of positions that an attacker might know in a given challenge. If an attacker knows $i \leq n$ password positions, then the proportion of challenges with m' known positions is given by $s_n^m(i,m')$ in Equation 3 (generalizing Equation 2).[3] As before, we can compute the fraction of challenges with $0 \leq m' \leq m$ known positions after $k \geq 1$ runs as $\overline{s_n^m}(k,m')$:

$$s_n^m(i,m') = \frac{\binom{i}{m'}\binom{n-i}{m-m'}}{\binom{n}{m}} \qquad \overline{s_n^m}(k,m') = \sum_{i=m}^{n} p_n^m(i,k)s_n^m(i,m') \tag{3}$$

When an attacker knows $0 \leq m' < m$ positions in a challenge after k runs, guessing can be used to determine remaining $m-m'$ response positions. We can calculate the β-success rate to give us the fraction of responses that are correctly learned after β guesses, as shown in Equation 4.

$$\sum_{j=0}^{m} \overline{s_n^m}(k,j)w_j \tag{4}$$

Here, the new term w_j represents the success rate of guesses when $j \leq m$ positions are known in a challenge, which depends on β and N and varies according to the guessing method. The sum weights w_j by the proportion of challenges with $0 \leq j \leq m$ matches to recorded positions.

First, we combine recording and brute force guessing. If m' positions are known in a given challenge, then at most $N^{m-m'}$ guesses are required to brute force guess the remaining $m-m'$ positions. Here w_j is the probability of guessing

[3] The computation for $s_n^m(i,m')$ was used by Goring et al [5] to determine the expected number of positions known in subsequent challenges.

correctly with β guesses, which is 1 when the number of guesses is less than or equal to the maximum allowed, β:

$$w_j = \begin{cases} 1 & \text{if } N^{m-j} \leq \beta \\ \frac{\beta}{N^{m-j}} & \text{otherwise} \end{cases} \tag{5}$$

The results are displayed in Fig. 7 which show different choices for n and m. For the PIN case (third line up on left graph of Fig. 7) the proportion of guessable challenges exceeds 50% after only k=2 runs, while for the alphanumeric case (third line down on right graph in Fig. 7) it takes k=3 runs.

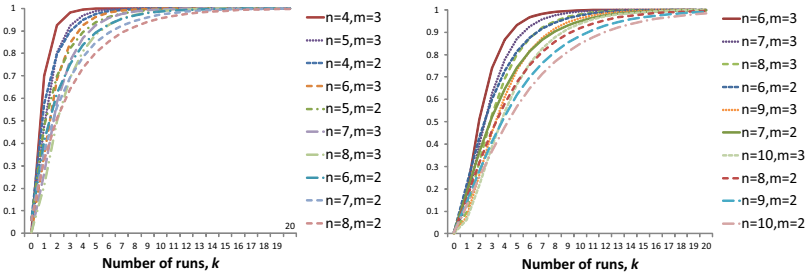


Fig. 7. Success rates for recording and brute force guesses, N=10 and N=36

We can improve the performance beyond brute force guessing by using our letter position frequency and projection dictionary attacks of Section 3. We provide a lower bound for the success rates in these cases by taking the best values for each w_j from Equation 4 for our example cases. In the alphanumeric case, w_0 represents the number of guesses when j=0 challenge positions are known; we earlier calculated a success rate of 5.5% using a projection dictionary. For w_1, guessing the two remaining positions, we take the success rate from a projection dictionary with m=2 sized challenges, which is 12% (versus 4% for the letter position frequency option). This is a rough lower bound: it treats the two remaining challenge positions as a lone challenge without taking into account any dependence upon the known password positions, which would reduce the answers possible (but would require the attacker to use more background dictionary data). Based upon a letter position frequency attack we computed a success rate of 60% for w_2 where we guess one remaining position in the challenge. Again, this success rate would be improved if knowledge of the two known positions is used. Finally, w_3=100% since all three positions are known.

For the PIN case, we similarly chose values $w_0 = 30.6\%$, $w_1 = 77.35\%$ and $w_2 = 100\%$. By plugging these values in Equation 4 we can compute the success rate for recording and best guessing attacks. The results for both k=1 and k=4 recorded runs are shown in our summary table in Fig. 8.

5 Summary

Partial passwords, introduced to prevent a telephone operator learning a user's password, have taken on a broader role in securing the online accounts of many banks. In addition to their likely susceptibility to guessing attacks, they do not appear to be adequate even to mitigate against a small number of recorded protocol runs, at least for typical choices of parameters used today.

Fig. 8 displays a summary of our attacks, showing success rates responding to challenges on partial PINs and passwords with the typical parameters sizes we found. For guessing attacks, our results are relative to data from the RockYou leak, and cannot be taken to be accurate for real PINs and passwords actually used in online banking. Nevertheless, these success rates are worryingly high, especially for banks that allow weak passwords and do not use a second credential or rely upon a second credential that may be easily obtainable such as a credit card number. Within the limited scope of our survey, Standard Life and 3DSecure for Bank of Ireland look at risk here, although, like some other banks, it appears that user ids are an additional credential for Standard Life as they are bank-issued and, we assume, unpredictable if they are not recorded.

Attack type	**parameters**	**percentage success rate**	
		PINs: β=6, N=10, n=6, m=2	alphanumeric: β=10, N=36, n=8, m=3
Brute force		6	0.002
Letter position frequency	RockYou	17.2	0.3
Dictionary	RockYou	15.3	3.9
Projection dictionary	RockYou	30.6	5.5
Recording	k=1 (k=4)	6.7 (63.1)	1.8 (59.0)
Recording + BF Guessing	k=1 (k=4)	41.1 (83.8)	9.6 (69.1)
Recording + Best Dictionary	k=1 (k=4)	60.2 (90.4)	25.2 (81.2)

Fig. 8. Summary of next-challenge attacks on partial passwords

Once recording takes place, attack success rates rapidly increase. It is claimed that half of online banking users access their account at least weekly [19,20]. Malware detection and removal, if operating, has a similar frequency, suggesting that if a key-logger is installed on a user's machine (or a public terminal), there is a good chance that at least one run of a partial password will be recorded. With k=1, only the PIN case yields a >50% success rate (k=2 for the alphanumeric case), so it can be argued that the partial mechanism provides *some* improvement over normal password authentication where an observer learns a complete password in a single step. As more runs are recorded, all success rates exceed 50%, and there is question over whether the partial scheme adds any useful security; reliance on the additional authentication mechanisms (where applicable) becomes primary.

Nonetheless, since the scheme is already widely used, it seems worthwhile for banks to enforce parameter choices that might improve the resistance to attack. There is an optimisation problem to solve here, to balance parameters both for sufficient usability and security. While a larger N improves resistance to guessing, it does not affect recording. So it may be worth first looking at the graphs in Fig. 6, which show that, unsurprisingly, larger password lengths n and smaller challenge sizes m increase the attack difficulty (the latter because it reduces the rate of revealing characters). But reducing m too far raises the success rate of guessing. Keeping m=3, but with $n = 16$, it requires recording 8 runs before the success rate exceeds 50% for an alphanumeric partial password. Longer passwords may be supportable by using passphrases (with a suitable interface) or even physical cards to store longer passwords; the latter is done with some products in a two-dimensional matrix [21].

Further work. There are several areas we want to investigate further. We have started to extend our study to the case considered by Goring et al [5] where the observer captures the response but not the corresponding challenge, which is the case when using a hardware key-logger. Varying the challenge format or sequence can have an effect. For example, repeating positions or giving positions in any order, not only ascending (one of our surveyed banks does this) greatly increases the number of challenges which would thwart the hardware key-logger case. We also want to extend our analysis to model *adaptive* projection dictionary attacks whereby guesses use the information previously learned. However, this requires the attacker to have the full dictionary, not just the top β answers for each challenge. Finally, we also plan to explore the usability of partial passwords, to explore the impact of parameter choices on users, as well as the effect of different presentation formats (e.g., drop-down menus versus text boxes) and challenge schedules. In the case of the latter, it may be interesting to examine how users react to different schedules, such as those generated maliciously in order to quickly reveal as many password positions as possible.

Acknowledgements. We're grateful to Atif Hussain who spent several months gathering the survey data used in Section 2 and to Ronald Bowes for hosting some of the password sources we used on his blog [18].

References

1. UK Consumers Association: Bank websites: How safe is yours? Which? Magazine, 24–27 (September 2011)
2. Matsumoto, T., Imai, H.: Human identification through insecure channel. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 409–421. Springer, Heidelberg (1991)
3. Li, X.Y., Teng, S.H.: Practical human-machine identification over insecure channels. Journal of Combinatorial Optimization 3(4), 347–361 (1999)
4. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)

5. Goring, S., Rabaiotti, J., Jones, A.: Anti-keylogging measures for secure internet login: An example of the law of unintended consequences. Computers & Security 26(6), 421–426 (2007)
6. Berkman, O., Ostrovsky, O.M.: The unbearable lightness of PIN cracking. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 224–238. Springer, Heidelberg (2007)
7. Focardi, R., Luccio, F.: Guessing bank PINs by winning a mastermind game. Theory of Computing Systems 50(1), 52–71 (2012)
8. Bonneau, J., Just, M., Matthews, G.: What's in a name? Evaluating statistical attacks on personal knowledge questions. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 98–113. Springer, Heidelberg (2010)
9. Bonneau, J., Preibusch, S., Anderson, R.: A birthday present every eleven wallets? The security of customer-chosen banking PINs. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 25–40. Springer, Heidelberg (2012)
10. Weir, M., et al.: Testing metrics for password creation policies by attacking large sets of revealed passwords. In: Proc. 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 162–175. ACM (2010)
11. Kelley, P.G., Komanduri, S., Mazurek, M.L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F., Lopez, J.: Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In: IEEE Symposium on Security and Privacy, pp. 523–537. IEEE Computer Society (2012)
12. Malone, D., Maher, K.: Investigating the distribution of password choices. In: WWW, pp. 301–310. ACM (2012)
13. Brostoff, S., Sasse, M.A.: "Ten strikes and you're out": Increasing the number of login attempts can improve password usability. In: Proceedings of CHI 2003 Workshop on HCI and Security Systems. John Wiley (April 2003)
14. Just, M., Aspinall, D.: On the security and usability of dual credential authentication in UK online banking. In: 7th International Conference for Internet Technology and Secured Transactions (ICITST 2012). IEEE (December 2012)
15. Bonneau, J.: The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In: IEEE Symposium on Security and Privacy, pp. 538–552. IEEE CS (2012)
16. Yan, J.J.: A note on proactive password checking. In: Proc. 2001 New Security Paradigms Workshop, NSPW 2001, pp. 127–135. ACM (2001)
17. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: Proc. of the 12th ACM CCS, pp. 364–372. ACM (2005)
18. Bowes, R.: SkullSecurity blog, passwords page, `http://www.skullsecurity.org/wiki/index.php/Passwords` (accessed September 2012)
19. Mahmood, Z.: Attitudes towards the use of e-banking: Result of a pilot study. Communications of the IBIMA 8, 170–174 (2009)
20. Thepaypers.com: UK consumers prefer online banking - survey (May 2011)
21. Voice, C.B., Chiviendacz, M., Pillman, E.: United states patent: 8060915 - Method and apparatus for providing electronic message authentication (November 2011)

A Appendix

In this appendix we provide more detail behind our calculations and data sets.

A.1 Bank Survey Resources

We gathered data directly from personal accounts and also consulted the following demo and information pages (see collected parameters in Fig. 2):

- ING DiBa: `https://www.ing-diba.de/kundenservice/banking-und-brokerage/#!01091`
- First Direct: `http://www1.firstdirect.com/1/2/banking/ways-to-bank/online-banking`
- Smile: `http://www.smile.co.uk/images/flash/smiledemo/`
- HBoS: `http://www.onlinebankingdemo.co.uk/launchbos.html`
- Nat West: `http://www.rbs.co.uk/personal/online-banking/g1/existing-customers/problems-logging-in.ashx`
- 3DSecure: `http://www.bank-of-ireland.co.uk/3dsecure/`
- AIB: `http://www.aibgb.co.uk/onlinebankingdemo/index.html`
- Bank of Ireland: `https://personalbanking.bankofireland.com/online-banking-demo/`
- Standard Life: `http://www.standardlife.co.uk/html/demo/home1.html` and `http://www.standardlife.co.uk/1/site/uk/help/faqs/online-servicing`
- Cooperative: `http://www.co-operativebank.co.uk/bankdemo/2008-06-04/IB_Demo_v2.html`
- Nationwide: `http://media.nationwide.co.uk/swfs/Demos/default.htm?DemoId=7`
- Skipton: `http://www.skipton.co.uk/demo/`
- Tesco: `http://www.tescobank.com/demos/index.html`
- Barclays: `http://www.barclays.co.uk/online/demo/?WT.ac=coukolbdemo`
- HSBC: `https://www.hsbc.ca/1/2/personal/banking`

A.2 Letter-Position Frequency Data

Fig. 9 shows the relative letter position frequencies from the RockYou set calculated on our two parameter cases: 6 digit PINs and 8 character alphanumeric passwords. These illustrate the statistics used in Sect. 3.1, demonstrating how

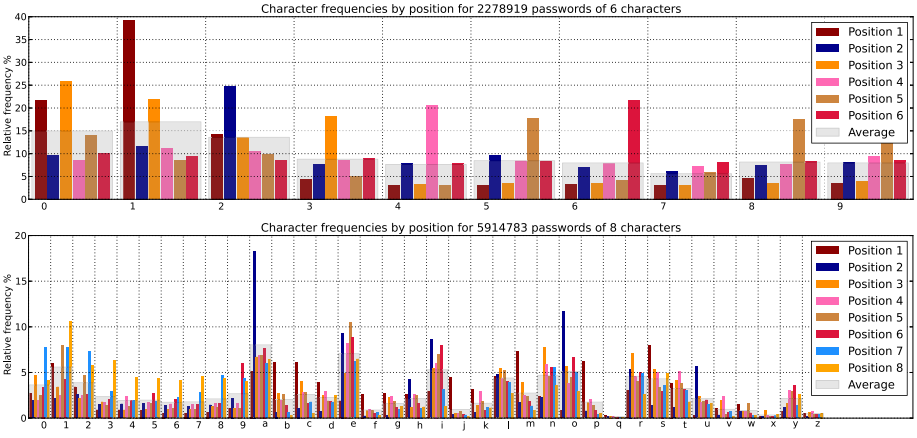


Fig. 9. Letter frequencies by position for RockYou passwords

frequencies of letters vary by position. The ranking of the top 10 characters in each position from this data is shown below, for each case:

											1	s	m	p	b	c	1	a	l	j	d
1	1	0	2	8	3	9	6	7	5	4	2	a	o	e	i	u	r	l	h	2	n
2	2	1	0	5	9	4	3	8	6	7	3	n	r	a	o	l	i	s	e	0	t
3	0	1	3	2	9	6	8	5	4	7	4	e	a	i	n	t	s	r	l	o	y
4	4	1	2	9	0	5	3	6	8	7	5	e	1	i	a	l	2	n	o	r	t
5	5	8	9	0	2	1	7	3	6	4	6	e	i	a	o	9	n	r	1	l	y
6	6	0	1	3	9	2	5	8	7	4	7	0	1	2	e	a	n	o	r	8	9
											8	1	e	a	3	2	s	7	4	8	5

So, for the challenge 2 3 6 the simplistic attack measured in Sect. 3.1 would guess in turn "a n e", "o r i", "e a a" and so on. We measure the success rate by seeing what proportion of the data set is then broken by making β of these guesses. This attack is not particularly successful, but is meant to illustrate a simple improvement using position data that improves over guessing at random.

A.3 Projection Dictionary Data

Projection dictionaries are built by taking a source dictionary, finding the answers to each of the $\binom{n}{m}$ challenges, and ranking each result by the number of times it occurs. The calculations are performed using some scripts written in Python. For PINs, an example table entry looks like this:

```
# Challenge (1, 3) has 100 distinct responses, and 2278919 total
# Challenge     Response Occurrences    Coverage %   Cum. %
  1. (1, 3)     13       340719         14.951%      14.951%
  2. (1, 3)     00       191577          8.406%      23.357%
  3. (1, 3)     11       175642          7.707%      31.065%
  4. (1, 3)     10       164900          7.236%      38.301%
  5. (1, 3)     01       143302          6.288%      44.589%
  6. (1, 3)     20       127502          5.595%      50.184%
# Metrics for Challenge (1, 3): Beta success rate: 50.18%
```

An attacker following this strategy would, given the challenge (1,3), reply with responses 1,3, then 0,0 and so on. As with the alphanumeric example, this ordering differs from the global ordering, where the top six PINs occurring as RockYou passwords were:

```
Passwords selected from data/rockyou-N10n6.txt
Selected passwords of length  6
Total of 2278919 passwords, with 390529 unique (17.14%)
Most common password '123456' occurs 290729 times
#    Word    Occurrences  Coverage %  Cum. %
  1. 123456  290729       12.757%     12.757%
  2. 654321  13984        0.614%      13.371%
  3. 111111  13272        0.582%      13.953%
  4. 000000  13028        0.572%      14.525%
  5. 123123  9516         0.418%      14.943%
  6. 666666  7419         0.326%      15.268%
# Password metrics: Beta success rate (beta = 6): 15.268%
```

Hey, You, Get Off of My Clipboard

On How Usability Trumps Security in Android Password Managers

Sascha Fahl, Marian Harbach, Marten Oltrogge,
Thomas Muders, and Matthew Smith

Distributed Computing & Security Group
Leibniz University of Hannover
Hannover, Germany
lastname@dcsec.uni-hannover.de

Abstract. Password managers aim to help users manage their ever increasing number of passwords for online authentication. Since users only have to memorise one master secret to unlock an encrypted password database or key chain storing all their (hopefully) different and strong passwords, password managers are intended to increase username/password security. With mobile Internet usage on the rise, password managers have found their way onto smartphones and tablets. In this paper, we analyse the security of password managers on Android devices. While encryption mechanisms are used to protect credentials, we will show that a usability feature of the investigated mobile password managers puts the users' usernames and passwords at risk. We demonstrate the consequences of our findings by analysing 21 popular free and paid password managers for Android. We then make recommendations how to overcome the current problems and provide an implementation of a secure and usable mobile password manager.

Keywords: Android, Security, Apps, Password Managers, Vulnerability.

1 Introduction

Today, using text-based passwords is the most prominent authentication scheme in computer systems. Although researchers have been criticising this scheme as being hard to use in a secure way, it is still the most widely adopted system for authenticating users. Previous research has shown that passwords chosen by users are often easy to compromise by attackers [1,2,3,7,10].

Another problem with the wide-spread application of passwords is password re-use. Users register with many services on the Internet, each requiring them to create a new username/password tuple. Previous research has shown that users often deal with this password overload by re-using the same or similar passwords for multiple accounts [8,13].

Multiple mechanisms have been proposed to support the user in choosing more secure passwords (cf. [12,14]), all trying to alleviate the challenges password-based authentication holds for their users.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 144–161, 2013.

However, due to the bounded cognitive abilities and motivation of users, password re-use is still commonplace. Password managers (PMs) aim to overcome this problem: they help the user to handle a large number of different passwords by storing them in encrypted form. To access the encrypted passwords, the user usually has to enter a single master secret that decrypts the password database. Password managers often include a password generator to simplify the creation of new unique and secure passwords. For convenience, login forms are pre-filled and account information for new websites is captured on the fly. Another prominent feature is the synchronisation of password databases between multiple devices.

While early password manager applications were limited to desktop computers and their browsers [11], current implementations offer mobile password manager apps, that can be synchronised over the cloud to ensure that the password database is available on all of the users' devices. While users of desktop password managers benefit from a smooth integration into browsers, password manager apps on mobile platforms offer less comfort. First, web browsers on smartphones and tablets often do not provide a plugin interface, that would allow for a smooth integration of password managers. Second, the existence of dedicated apps for many online services steadily increases the number of users that access online services through an app instead of a website in a general-purpose web browser. This circumstance requires that mobile password managers have to be able to manage passwords not only for browsers but also for apps.

Unfortunately, there is a fundamental problem with password manager apps on Android: The OS does not offer an API to integrate password managers with the browser or other apps. This has led to the adoption of a highly insecure practice to overcome this weakness: Password managers use the OS clipboard to transfer credentials from a password manager app to the browser or other apps. This method effectively broadcasts credentials to all apps installed on the smartphone.

We analysed the security of 21 password manager apps on Android, having a combined install base of 2,500,000 to 10,900,000 devices[1]. We found that apps mostly use AES, Blowfish or a combination of both to encrypt credential databases, although some apps use their own crypto implementations and do not rely on a proven open-source library. Seven apps use a key derivation function to derive the symmetric encryption key from the user's master secret to strengthen the security of encrypted credentials. We found one PM that directly inputs the user's password as the encryption key and truncates passwords longer than 16 characters. In case the password has less than 16 characters, the string '`FEDCBA9876543210`" is appended to "*strengthen*" the password. Another PM app uses an HMAC algorithm with SHA-256 and the fixed initialisation vector `''notverysecretiv''` for key derivation for every encrypted credential tuple. Three password manager apps provide their own cloud synchronisation feature to easily share passwords between multiple devices. Two of them synchronise the users' databases over a broken TLS channel (i. e. the apps are vulnerable

[1] The numbers are based on information provided by Google's Play Market. Google does not provide more fine-grained numbers for an app's install base.

due to incomplete TLS certificate validation) and hence allow a Man-In-The-Middle attacker to capture the databases. We investigated this problem in [5] in more detail and analysed 13,500 popular Android apps and found that many app developers failed to apply TLS appropriately. However, most critically, all apps use the clipboard feature to transfer credentials from the password manager to browsers or other apps. Two apps automatically copy credentials to the clipboard when the user clicks the URL for the given online account. Only one app allows a user to disable the clipboard feature.

While there is known malware on desktop computers that threatens passwords copied to the computer's clipboard[2], this circumstance has not been investigated in detail yet. Hence, to the best knowledge of the authors, this paper is the first analysis of password managers' security on mobile devices.

The remainder of the paper is organised as follows: Section 2 gives background information on password managers and discusses peculiarities of password managers on desktops and mobile devices. Section 3 describes our attack against password managers on Android and illustrates how captured credentials can be linked to online services and how stolen information can be transferred to the attacker's server without giving the user a chance to notice. Section 4 gives more details on the security of the PMs we analysed. To understand why developers added the clipboard features to their apps, Section 5 summarises open interviews with app developers. Section 6 discusses countermeasures and presents an implementation of a secure and usable password manager that overcomes the presented vulnerabilities. In Section 7, we conclude the paper and discuss future work.

2 Background

While early password managers were simply a username/password database embedded in desktop computers' browsers, the features of modern password managers are much more extensive and can be categorised as follows:

Browser-Embedded PM: Most web browsers, such as Google Chrome, Microsoft Internet Explorer, Mozilla Firefox or Apple's Safari, include an embedded password management feature. In case a user logs into a website for the first time, the PM inquires whether the login credentials should be saved to ease future logins by automatically filling the username and password into the login form. These embedded password managers traditionally store credentials locally on the user's computer, but are increasingly syncing them between multiple devices using proprietary Cloud services. Some browsers do not encrypt credentials stored locally or require the user to set a master secret to enable encryption. For example, Chrome uses the Google Account password to encrypt the synced password database by default, but offers to use a dedicated secret as an advanced feature.

[2] e. g.: `http://www.f-secure.com/v-descs/trojan_w32_qhost_je.shtml`

Browser Plugins: The majority of modern desktop web browsers provide an API to extend their functionality by allowing the user to install third-party plugins or extensions. Many password managers are hence available as browser plugins. KeePass[3], 1Password[4] and Lastpass[5] are prominent examples of plugin-based password managers. They encrypt passwords and protect them with a master secret. These third-party password managers often provide further functionality and act as encrypted storage for more than just usernames and passwords: other sensitive information such as credit card numbers, online banking information or secret notes can be stored as well.

2.1 Password Managers on Desktops

Password managers on desktop computers are generally well integrated into the users' everyday Internet-facing software, such as browsers and email clients. Regardless of whether an embedded password manager or a third-party plugin is used, when the user accesses an online account for the first time or creates a new account, the password manager automatically comes into play and offers the user to securely store the new account information. The plugin APIs of modern browsers offer a very comfortable integration of password managers into the users' workflows. When a user visits a website that requires authentication, a password manager typically auto-fills the username and password and might even automatically submit the login form.

Early password managers for desktop computers (e. g. [11]) assumed a single device environment. Nowadays users often work with multiple devices such as desktops, notebooks, smartphones and tablet PCs, which makes it necessary to synchronise password databases between multiple devices to have credentials available whenever needed. For this reason, some password managers offer to sync databases between multiple devices by storing credentials in the Cloud or by putting the database on USB drives. A popular way to sync password databases is the Dropbox service. The encrypted password database is stored in the user's Dropbox account and can be accessed from all the user's devices. 1Password for example maintains an encrypted database for sensitive information and allows users to store and share the database via their Dropbox account.

2.2 Password Managers on Mobile Devices

While password managers in desktop environments are well integrated into browsers and users' workflows, the situation for third-party password managers on mobile platforms is different. Neither of the major mobile platforms (Android, iOS and Windows Mobile) nor mobile browsers provide a plugin API comparable to desktop computers. Additionally, the paradigm shift away from the browser as a generic tool to surf the Internet towards the „there is an app for everything" approach makes integrating PMs into mobile ecosystems even harder.

[3] cf.: `http://keepass.info/`
[4] cf.: `https://agilebits.com/onepassword`
[5] cf.: `https://lastpass.com`

API limitations and the requirement to support arbitrary apps creates a different usage pattern for mobile PMs. Instead of storing new account information automatically and auto-filling authentication forms, the workflows of mobile PMs typically consist of the following steps:

1. The user has to switch to the PM app,
2. then needs to find the appropriate username/password tuple from a list of stored credentials,
3. copies the password to the clipboard,
4. switches back to the app that requires authentication, and
5. finally pastes the password into the corresponding text field before submitting the login form.
6. In case the user does not remember the username for a given service, these steps (except step 2) are repeated for the username as well.

Although this workflow's usability is far from optimal, it is the best mobile password managers can provide so far. To understand why users nevertheless use password managers on mobile devices, we analysed 2,000 user reviews in Google's Play Market. To this end, we downloaded user reviews, manually extracted factors that motivate users to use PMs on their Android device and identified the following reasons to be substantial:

Protection: Users feel that embedded PMs do not store the passwords in a way they believe to be secure (e. g. some users were angry that Android's stock browser does not encrypt stored usernames and passwords).

Confidentiality: Users do not trust embedded PMs in keeping their data confidential (e. g. users were afraid that their credentials could be sent to Google).

Features: Embedded PMs are usually limited to usernames/passwords. Users often want to store other confidential data, such as banking information.

Availability: Embedded PMs are usually limited to a single browser. Since many users need access to their information on multiple devices and browsers, a vendor-independent PM is preferred.

After having outlined background information on password managers, the next section presents our attack on PM apps on Android and illustrates its consequences for the user.

3 Password Sniffing on Android

As illustrated in Section 2.2, the workflow of mobile password managers requires the user to copy account credentials to the clipboard before switching to the target app and pasting them before actually logging in. There are some problems with this practice: On Android, writing data to or reading data from the clipboard does not require any permission. Therefore, every app currently running on an Android device can read the items stored in the clipboard at any time. To make matters worse for password managers, the Android SDK provides the

`android.content.ClipboardManager.OnPrimaryClipChangedListener` interface, which defines a listener callback that is invoked each time the primary item on the clipboard changes. This can be used by malicious apps to harvest passwords as they are passed through the clipboard. As a proof of concept we implemented a password sniffer named *PWSniff* using this mechanism. PWSniff runs as a background service and does not require any Android permission to work properly.

As long as no changes to the clipboard occur, the background service idles and therefore does not consume any CPU cycles. Directly after a new item is copied to the clipboard, the listener callback is invoked by Android and the idling PWSniff background service is notified and then reads the primary item. Next, PWSniff determines the app which is currently in the foreground. This information can also be acquired without requesting any permission. We assume that the foreground app at the time of copying is the app from which a user copied data (cf. Section 2.2 step 1). In case this app is a known password manager, we assume that the primary clipboard item is either a service URL, a username or a password (cf. Section 2.2 step 3). Whether the app is a password manager can be determined based on the app's user ID, which is assigned at install time and can be mapped to the a unique app market ID. The third step in our attack is to wait for a foreground app switch by checking the current foreground app in a loop and waiting until the user brings another app to the foreground. In case we identified the primary clipboard item as possibly confidential data (no matter if it is a username or a password) copied from a password manager, the new foreground app is assumed to be the destination of the credentials-copy-operation (cf. Section 2.2 step 4). Hence, by exploiting features of the Android SDK that require no special permissions in combination with a typical workflow in the context of using password manager apps on Android, it is easily possible to harvest (still potentially noisy) usernames and passwords from the world readable clipboard.

At this point, an attacker cannot be sure which item is the username and which the password. But, in many cases it is possible to differentiate between both items based on their structure. Usernames are often chosen to be easily memorable (e. g. an email address) while passwords, especially those which are managed with PM software, usually are more "cryptic". Even in cases where the username and password cannot be easily distinguished, an attacker could first try one combination of the sniffed items and in a second attempt the reversed order. In both cases, breaking into an account is straightforward.

Advanced Username Capture: The attack described above relies on the user copying and pasting both the username and the password. Since users might just type their usernames from memory or use browser or app autofill features to save this effort, it might become necessary to acquire the username through an alternative method. For this, PWSniff can be equipped with the GET_ACCOUNTS permission. The permission allows the app to see usernames that other apps handle on the smartphone and which are registered with the *AccountManager*[6].

[6] cf.: `http://developer.android.com/reference/android/accounts/AccountManager.html`

This also includes all email addresses used on the device. Since many online services use email addresses as usernames, this list offers a good basis from which to guess usernames for many services.

The downside of this extension is that it involves the danger of a user becoming suspicious of the app's permissions, which are presented to the user at install time. However, Felt et al. [6] demonstrated that users pay little attention to the permissions of an app and mostly do not understand the permissions' meaning. While Felt et al's. results account for Android's permission system in general, an app's permissions are also grouped and classified based on their security relevance. In this respect, the GET_ACCOUNTS permission does not rank particularly high and thus is not often shown on the first page. To ensure that the GET_ACCOUNTS permission is not shown on the first page, an attacker only needs to add more than three popular permissions such as INTERNET, LOCATION and STORAGE which are used by many apps to his malware app. The best composition of permissions to mask the GET_ACCOUNTS permission is outside the scope of this work.

Advanced Account Capture: In Section 3, we illustrated that an Android app which holds no special permissions is able to sniff online account credentials that are copied to the clipboard when working with any password manager on Android in most cases. It is also possible to learn from which app a value was copied to the clipboard and into which app the value was pasted. If the target app has a special purpose (e. g. the Skype app only logs into Skype), it is easy to guess to which online service the harvested credentials belong. However, in case the target app is a multi-purpose Internet client such as a web browser, finding the intended service is not quite as straightforward.

To learn for which account a password is used, an attacker can benefit from Android's ProcFS features. The ProcFS is an interface to the kernel and provides information about a device such as information about the CPU, memory and network details. On Linux-based systems such as Android, the ProcFS is usually mounted at `/proc`. Most entries in `/proc` and its subdirectories can be read by everyone. The `/proc/net/tcp` file contains information about all TCP connections on an Android device and is also world-readable and hence accessible by every app without requiring any permissions. Information such as source IP and port, destination IP and port and the UID of the process that created the network connection are listed there. Since Android creates a static mapping of Apps to a UID at install time, one can easily learn which app connects to which Internet hosts based on the UID entry in `/proc/net/tcp`. Having the destination IP for an app's network connection at hand allows an attacker to easily infer to which online service a credential pair is connected by logging all network connections of an app, immediately after a copy operation from a PM to another app was discovered.

Exfiltrating the Data: In [5], we found that 92.8 % of 13,500 popular Android apps request Internet access. Adding the Internet permission to PWSniff should thus not raise undue concern. With this permission, transmitting

the harvested data is of course trivial. However, if a zero permission attack app is desired, exfiltration of the harvested data can still be done using another flaw in Android's permission system. Egners et al. [4] describe a loophole in Android's permission system that we adopt for our purposes and which allows PWSniff to send gathered credential information to a remote HTTP server without requiring the Internet permission. After the account login information was gathered, the harvested data is cached until the device's display is turned off. When this happens, an HTTP URL with the following structure is built: `http://<pwsniff-master>/pw#username#service`. This URL is used to invisibly open Android's stock browser when the display is turned off by running the following code in PWSniff. We explicitly call Android's stock browser since some third-party browsers do not hand back control for unknown protocols to the Android OS, which is required to keep the attack stealthy.

The server behind the URL replies with a location header containing a custom protocol, for example: `'Location:pwsniff://all.ok'`. Since PWSniff includes an activity that previously registered for the custom `pwsniff://` protocol, the browser passes handling for the URI `pwsniff://all.ok` to PWSniff. Staying invisible, the activity then simply terminates.

After demonstrating how credentials can be sniffed when Android password managers are used, how they can be mapped to online accounts and how this information can be exfiltrated stealthily, the next section gives some relevant excerpts of our detailed security analysis of PMs on Android.

4 Security in Detail

We analysed 13 free and 8 paid Android PM apps in detail. Our intention was to analyse which apps include the clipboard feature for credential copy & paste, which encryption algorithms protect the password database, whether or not the app includes an embedded browser, whether or not the SD card is used to store the password database and whether or not the app removes itself from the recent apps view. For analysis, we installed all apps on a Samsung Galaxy Nexus with Android 4.0. We applied forensic techniques[7] to learn database and configuration files' structures of the installed password manager apps. To learn internals of the password managers, we decompiled them[8] and conducted manual static code analysis.

We also conducted static code analysis on the same dataset as in [5] and found that only two apps in this dataset registered for the clipboard change listener. We analysed both apps manually and found no malicious behaviour in the apps. 907 apps (6.7 %) in the sample access the clipboard API programatically to

[7] We used the adb tool (cf.: `http://developer.android.com/tools/help/adb.html`) for logical extraction.

[8] We used a bundle of decompilation tools: JD-GUI (cf.: `http://java.decompiler.free.fr/?q=jdgui`), apktool (cf.: `http://code.google.com/p/android-apktool/`) and dex2jar (cf.: `http://code.google.com/p/android-apktool/`)

share more complex objects than simple text strings such as images, video or audiofiles.

Table 1 in the Appendix shows an overview of the security parameters we analysed.

4.1 Encryption

One important aspect of PM security is the encryption mechanism to store credential databases. Android's stock browser does not encrypt stored passwords in any way but protects them from unauthorised access by file system permissions. Android's AccountManager mechanism provides centralised credential storage and also protects user credentials from unauthorised access by file system permissions, but the `accounts.db` database is not protected with an extra layer of encryption. This does not protect the password from forensic analysis.

All third party PMs we analysed apply some encryption mechanism to protect the data. Android supports (3)DES, RC2, and RC5[9] to encrypt data out of the box. Other encryption algorithms require the developer to add third-party libraries to their app. We decompiled the PMs to find out what kind of encryption algorithm is applied in each PM app. To provide stronger security, most password managing apps use the Advanced Encryption Standard (AES) with several key lengths. aWallet uses a combination of AES, Blowfish and 3DES.

A critical aspect of encrypting password databases is the derivation of the encryption key [9] that is directly connected to the master secret used to unlock/decrypt the password database. Seven apps use a dedicated key derivation function to derive the symmetric encryption key from the user's master secret to strengthen the security of encrypted credentials. We found one app that directly inputs the user's password as the encryption key, truncating passwords longer than 16 characters. In case the password has less than 16 characters, the string "FEDCBA9876543210" is appended to *"strengthen"* the password. Another app uses an HMAC algorithm with SHA-256 and the fixed initialization vector `''notverysecretiv''` for key derivation.

4.2 Storage

Most password managers, including Android's stock browser, store password databases in files or SQLite databases that are only accessible by the password manager app itself. Hence, other Android apps cannot access account information regardless of whether it is encrypted or not. Ten of the analysed PMs store databases on the SD card that is world readable without requiring further permissions on all devices with Android 4.0 and older. In combination with inappropriate database structures (not encrypting all information stored in the password manager), an attacker is for instance able to learn for which services a user holds accounts or for which services the same username and/or passwords are used.

[9] cf.: `http://developer.android.com/reference/javax/crypto/spec/package-summary.html`

4.3 Recent Apps

An essential feature of Android devices is an overview of the currently running apps, also called the *Recent Apps View*. The Recent Apps View shows thumbnails of current foreground activities of all running apps. While a security feature of all analysed password managers is the automatic locking of the password database either immediately after the password manager app was left or after a configurable amount of time, we found only three apps that also replace their thumbnails in the recent apps view (cf. Table 1). In case the user copied online account information (usually the location, username and password) and then leaves the password manager app to paste the information into another app, the account information is left in the recent apps view and can be seen by anyone with physical access to the user's device. Although this threat is orthogonal to our attack (cf. Section 3), it outlines a security risk for users' online credentials.

4.4 Cloud Sync

While all password managers store their databases locally and allow synchronisation of their databases, most offer a more manual functionality using Dropbox or similar services. LastPass, SecureSafe and RoboForm provide dedicated Cloud storage features to automatically synchronise all passwords remotely. In [5] we analysed popular Android apps and found that many app developers fail to apply TLS appropriately, being vulnerable to active Man-In-The-Middle attacks. Although LastPass, SecureSafe and RoboForm protect their network communication with TLS, SecureSafe and RoboForm fail to verify the cloud servers' TLS certificates. Instead, they accept all certificates. In case of SecureSafe this however has not further security implications since in addition to TLS, SecureSafe uses a session-specific symmetric key, which is set up during the SRP-login[10], to additionally encrypt password-data end-to-end. However, RoboForm leaks the users' credentials which are used for password encryption in the default case (i. e. the user did not choose an extra password for encryption). Hence, an attacker can gain access to the data in cleartext under this circumstances.

5 The Developers View

After analysing Android password managers on a technical basis, we contacted their developers via email and informed them about a possible security threat for their users. We offered them to get in contact either via email or telephone to discuss the details of the PWSniff attack. We also posed the following questions:

- Why was the C&P feature used in the password manager app?
- Were developers aware of the security threats arising from using the clipboard for username/password sharing, and, if so, why did they add the C&P feature nonetheless?

[10] cf. RFC2945.

– Which features, if any, do developers miss in Android's SDK for developing a password manager app?

15 of the 21 developers agreed to participate in the email interview and are anonymously referred to as P1, ..., P15 in the following.

5.1 Results

During the discussions with developers, we were able to identify three different reasons to add the usability-enhancing clipboard feature to PM apps. One was because the developers themselves were users of their apps and desired the feature themselves. ("As I'm a [...] program user too, I added the copy feature because I needed to transfer usernames (that are usually long email addresses) and passwords to login forms in web browsers."; P7). The second reason provided by PM developers was the wish to come as close as possible to PM functionality on the desktop, because developers believed that users would reject their apps if they were not sufficiently usable. ("Copy to clipboard has been in [...] Android from early on. [...] It was something that we knew we needed to make the application usable at all"; P4). Lastly, developers reported that users directly requested a C&P feature for their app ("The feature was highly requested by users. The most common example: users want to login to a website on their mobile device, so he/she copies credentials from [our PM] to the clipboard and then pastes them into the browser."; P15).

All but one developer were aware of security threats resulting from putting passwords into a device's clipboard. Developers who were aware of the security threat justified adding the clipboard integration, stating that they had no other choice. They described it as a tradeoff between usability and security which was decided in favour of increasing usability ("It's a balance between ease of use and security. Of course it would be much more secure to not use the clipboard, however people accept the risk of doing so; the alternative of not using a password manager is worse."; P3). One developer interestingly described his decision not as a usability-security tradeoff but as a "one type of security versus another type of security" decision, alluding to the fact that without password managers users would choose less secure passwords. Additionally, P4 stated: "On the whole, I think that password reuse [...] is currently the biggest single problem with password security today. And so, if a password manager gets people to use unique passwords for each site, the dangers of a publicly readable clipboard is a security risk that can be worthwhile. [...] What's the alternative?".

All developers criticised Android's missing support for password manager apps. A native integration into third party apps and browsers was described as the most effective countermeasure against the password sniffing security threat ("Android doesn't offer hooks into the native default browser [...] and does not allow our app to access input fields of other apps [...] which makes it necessary that password managers make heavy use of the clipboard."; P3).

5.2 Discussion

Based on the lack of API support for third-party password managers on the Android OS, developers decided to opt for the best usability they could achieve by including the clipboard feature to allow users to copy-and-paste usernames and passwords from their apps to other apps. Although all but one developer were aware of the possible security threat, they decided that better usability was more important than stronger security. A justification multiple developers offered was that *they had no other choice* and that it was necessary to add the best possible usability even if security was threatened.

6 Countermeasures

With the results of our analysis and the developers' comments in mind, we first discuss possible countermeasures to improve the security of a smartphone's clipboard facilities as a global shared memory. Additionally, we present a PM implementation for Android based on a customised soft-keyboard that provides usability features similar to desktop PMs and does not leak credentials over public channels.

Secure Clipboard Architecture: Sniffing confidential information on Android devices is currently easy since on the one hand, a proper plugin API for integrating password managers is missing and, on the other hand, the design of the current clipboard mechanism on Android is not made for sharing confidential information between apps. The current clipboard model allows an arbitrary app to access clipboard items deposited by any other app. With the assumption that both, the copy as well as the paste operation are triggered by the user, such a clipboard model does not cause security concerns. However, on Android, two other API features open the door for malicious activity: Android's background service feature for apps and the `ClipboardManager.OnPrimaryClipChangedListener` allow for stealthy harvesting of clipboard items (cf. Section 3). Therefore, we present two possible modifications to improve Android's clipboard model when it is accessed using API functionalities:

Permissions. The current clipboard model allows every app to programmatically read data from and write data to the clipboard, without requiring permission for that. While user-triggered clipboard operations can remain unchanged, we propose two new permissions for API-based access to clipboard functionality: `WRITE_TO_CLIPBOARD` and `READ_FROM_CLIPBOARD`. Although the limited effects of Android's permission model for the average app user have been discussed (cf. Felt et al. [6]), these permissions should be added for completeness. This way, at least the tech-savvy users would have a chance to see if an app is capable of accessing the clipboard programmatically and can warn the rest of the community. Since we identified only very few apps to access the clipboard programatically (cf. Section 4), the proposed changes would only impact a small number of apps. Regular, user-triggered copy-and-paste operations would not be influenced by this modification.

Targeted Clipboard. Copying a value to the clipboard on current Android smartphones is equivalent to broadcasting the information to all other apps. This is contrary to the users' intuition of using a copy-and-paste feature that is generally used to transfer information from one app to another. Therefore we propose to extend API calls to the clipboard with a "target app" parameter that the app may request from the user. Keeping usability in mind, the number of target apps should be kept to a minimum. Apps providing an API-based copy feature may let the user choose target apps from a list of all apps or suggest useful targets as well as remember previous preferences. If clipboard operations are triggered by the user, reading the clipboard's contents should only be possible through explicit user interaction as well.

The modifications to Android's current clipboard model proposed above do not only protect credentials from unwanted disclosure, but can also serve to shield any other (possibly confidential) information (such as financial or medical information), that a user might copy to the clipboard.

USecPassBoard: While the above solutions would alleviate the current security problems of PMs, they would also require modification of the Android OS itself. Additionally, these measures cannot address the usability issues of mobile PMs, i. e. that the user needs to manually select credentials, switch apps and manually paste. To offer both better security *and* usability we propose a novel password manager: USecPassBoard. To overcome the issues plaguing the traditional approach of mobile password managers, we went down a different path. We created a soft-keyboard which integrates a password manager. Since soft-keyboards are available in every app and can access a shared credential database, they integrate well with most scenarios where credentials need to be entered. A custom soft-keyboard implementation on Android replaces the default keyboard and provides a custom means to input data into user-input fields. Figure 1 in the Appendix shows the user interface of the USecPassBoard PM. Besides preserving the regular keyboard functionality, it essentially adds two operations: (1) Creating a new username/password entry and (2) inserting a username/password tuple at the user's discretion. Since USecPassBoard is a soft-keyboard, it is available in every application, including the browser and stores passwords in a master secret-protected AES-256 encrypted database[11] to protect username/password tuples from unauthorized access. This effectively avoids the use of copy-and-paste on usernames and passwords while maintaining the flexibility of all available password managers.

New Account: USecPassBoard analyses the context of user input to determine if credentials are being entered. The password context is determined by identifying which app is currently used (i. e. which app is in the foreground) and in case the foreground app is a browser, it determines the website which is displayed by reading the browser's first item cached in the history. Apps are uniquely identified

[11] We use the SQLCipher (cf.: `http://sqlcipher.net/sqlcipher-for-android/`) database.

based on their package names managed by the Android operating system[12]. USecPassBoard then caches the input of all textfields in the foreground activity. This is possible since soft-keyboards on Android are triggered when a textfield is activated by the user. Additionally, a soft-keyboard receives a reference of the `EditorInfo` class[13] which identifies an input as a text or a password field. After the user completes the input and the keyboard loses the focus on a password field, a notification is displayed in the status bar that a new dataset was created (cf. Figure 1) if there is no identical username/password tuple for the current context in the database. New username/password tuples are bound to the target app – based on its package name – and are not available for possibly malicious apps. In case the user would like to share credentials between different apps (e. g. between two Facebook client apps), we allow this in the settings menu of USecPassBoard.

Credential Insertion: In case USecPassBoard recognises a known password context (i. e. a package name of an app for which credentials are stored in the database), the user can choose to insert this information by tapping into the input field for the username or password. A popup message appears after the user tapped onto the key button (cf. Figure 1) and a list of available credentials for the given password context is displayed. Subsequently the selected username/-password tuple is inserted at the user's discretion and the login process can be started.

Security Considerations: All interactions between the USecPassBoard virtual keyboard and a target app must be initiated by the user by tapping into a text input field. This creates a communication channel between the keyboard and the target app through Android's InputMethodManager[14] which is not accessible from other third party apps. This allows the automatic storage of new account credentials and insertion of stored credentials into uniquely identifiable target apps.

Target apps are uniquely identified based on their package name that is managed by the Android OS and cannot be spoofed by malicious apps[15]. In case the target app is the browser, a password context consists of the browser's package name and a target website. We identify the target website by reading the top item from the browser's history. This is accessible with Android's READ_HISTORY_BOOKMARKS permission and gives us the currently viewed website. Hence, we can avoid that users falsely insert credentials another website.

[12] cf.: `http://developer.android.com/guide/topics/manifest/manifest-element.html`

[13] cf.: `https://developer.android.com/reference/android/view/inputmethod/EditorInfo.html`

[14] cf.: `http://developer.android.com/reference/android/view/inputmethod/InputMethodManager.html`

[15] cf.: `http://source.android.com/tech/security/`

7 Conclusions

With the rise of mobile devices, mobile password manager apps could be an integral security tool for smartphone and tablet PC users. Since Android based devices lack APIs for the integration of password managers, current solutions rely heavily on the clipboard to share credentials between the PM and other apps. We analysed 21 popular password managers on Android which all are vulnerable to credential sniffing because a device's clipboard is a publicly available storage that can be accessed from any app. We showed that, using additional context information, malware is able to link the stolen credentials to the corresponding online account in many cases. We interviewed developers of the analysed PM apps and found that the majority of them were aware of possible security threats but accepted the risk to provide better usability. Based on the analyses' findings and developers' feedback, we discuss modifications to Android's clipboard mechanism to increase security for sensitive information. Finally, we present a soft-keyboard that integrates a secure and easy-to-use password manager which prevents the leakage of usernames/passwords via the clipboard. This password manager design is the first to offer both usability and security for Android-based password managers.

Since, in addition to security, usability is crucial for a password manager, in future work we plan to conduct multiple user studies for the USecPassBoard password manager.

References

1. Bishop, M., Klein, D.V.: Improving system security via proactive password checking. Computers & Security 14(3), 233–249 (1995)
2. Bonneau, J.: The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In: 2012 IEEE Symposium on Security and Privacy (SP), pp. 538–552 (2012)
3. Dell'Amico, M., Michiardi, P., Roudier, Y.: Password Strength: An Empirical Analysis. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9 (2010)
4. Egners, A., Marschollek, B., Meyer, U.: Messing with Android's Permission Model. In: IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom 2012) (May 2012)
5. Fahl, S., Harbach, M., Muders, T., Baumgärtner, L., Freisleben, B., Smith, M.: Why eve and mallory love android: an analysis of android ssl (in)security. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, pp. 50–61. ACM, New York (2012)
6. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: user attention, comprehension, and behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, SOUPS 2012, pp. 3:1–3:14. ACM, New York (2012)
7. Florencio, D., Herley, C.: A large-scale study of web password habits. In: Proceedings of the 16th International Conference on World Wide Web, pp. 657–666 (2007)

8. Gaw, S., Felten, E.W.: Password management strategies for online accounts. In: Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS 2006, pp. 44–55. ACM, New York (2006)
9. Kaliski, B.: PKCS #5: Password-Based cryptography specification version 2.0. RFC 2898, Internet Engineering Task Force (September 2000)
10. Malone, D., Maher, K.: Investigating the distribution of password choices. In: Proceedings of the 21st international Conference on World Wide Web, WWW 2012, pp. 301–310. ACM, New York (2012)
11. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J.C.: Stronger password authentication using browser extensions. In: Proceedings of the 14th Conference on USENIX Security Symposium, SSYM 2005, vol. 14, p. 2. USENIX Association, Berkeley (2005)
12. Shay, R., Kelley, P.G., Komanduri, S., Mazurek, M.L., Ur, B., Vidas, T., Bauer, L., Christin, N., Cranor, L.F.: Correct horse battery staple: exploring the usability of system-assigned passphrases. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, SOUPS 2012, pp. 7:1–7:20. ACM, New York (2012)
13. Shay, R., Komanduri, S., Kelley, P.G., Leon, P.G., Mazurek, M.L., Bauer, L., Christin, N., Cranor, L.F.: Encountering stronger password requirements: user attitudes and behaviors. In: Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS 2010, pp. 2:1–2:20. ACM, New York (2010)
14. Ur, B., Kelley, P.G., Komanduri, S., Lee, J., Maass, M., Mazurek, M.L., Passaro, T., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F.: How does your password measure up? The effect of strength meters on password creation. In: Proceedings of the 21st USENIX Conference on Security Symposium, Security 2012, p. 5. USENIX Association, Berkeley (2012)

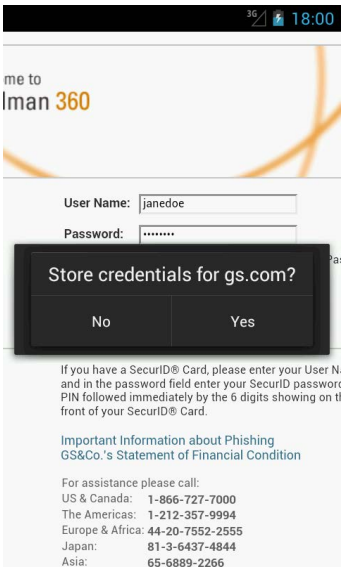
A Investigated Password Managers

Table 1. Overview of the analysed Android password manager apps. (EM=Encryption Method, KD=Key Derivation, C&P=copy&paste functionality, EB=Embedded Browser, SD=Writes database to SD card, RA=Removes itself from the recent apps view)

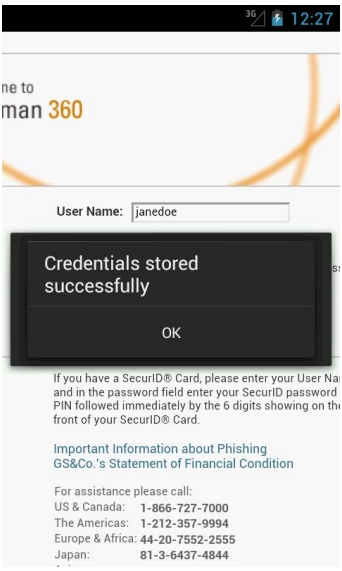
Free							
App	**Installs**[1]	**EM**	**KD**	**C&P**	**EB**	**SD**	**RA**
PassDroid	100-500k	AES	SHA-256	✓	–	Backup	✓
1Password	100-500k	AES	PBKDF2	✓	✓	Always	✓
KeePassDroid	500k-1m	AES[1]	SHA-256	✓	–	Always	✓
UPM	100-500k	AES	PBE	✓	–	Backup	✓
Pocket	100-500k	AES	PBE	✓	–	Backup	✓
NS Wallet	10-50k	AES	–	✓	–	✓	✓
LastPass	100-500k	AES	○[4]	✓	✓	–	–
PasswdSafe	10-50k	AES	–	✓	–	✓	✓
OI Safe	100-500k	AES	PBE	✓	–	✓	✓
aWallet	100-500k	AES[2]	SHA-256	✓	–	Backup	✓
Moxier Wallet	10-50k	AES	SHA-256	✓	–	–	✓
Keeper	1-5m	AES	SHA-1	✓	✓	Backup	✓
RoboForm	100-500k	○[4]	○[4]	✓	✓	Backup	✓
Paid							
mSecure	100-500k	Blowfish	SHA-256	✓	–	✓	–
Secret Safe	50-100k	AES[3]	SHA-256[3]	✓	–	Backup	✓
SafeWallet	10-50k	AES	HmacSHA1	✓	–	✓	✓
SPB Wallet	10-50k	AES	○[4]	✓	–	✓	✓
eWallet	10-50k	AES	PBE	✓	–	✓	–
Handy Safe Pro	10-50k	Blowfish	–	✓	–	–	✓
DataVault	1-5k	AES	–	✓	–	Backup	✓
Password Box	5-10k	AES	–	✓	–	✓	✓

[1] KeePassDroid combines AES and Twofish
[2] aWallet combines AES, Blowfish and 3DES
[3] Secret Safe combines AES and Twofish for encryption and multiple rounds of SHA-256 and Whirlpool for key derivation.
[4] This information could not be found by reverse engineering.

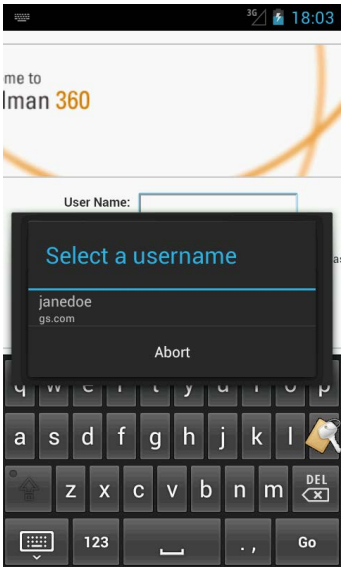
B USecPassBoard User Interface



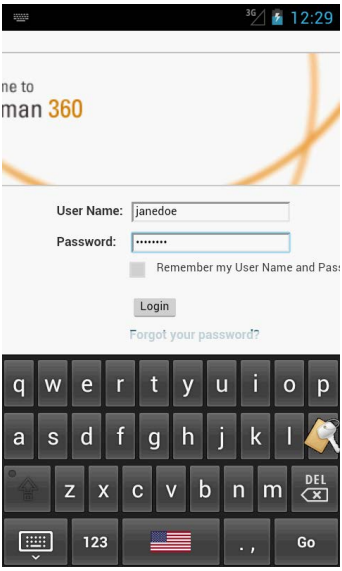
(a) Asking the user to store new credential tuple.



(b) Successfully stored new credential tuple.



(c) Selecting existing credential tuple.



(d) Credentials filled in.

Fig. 1. The USecPassBoard workflow for storing new credential tuples and filling in stored credentials

Unique Ring Signatures: A Practical Construction

Matthew Franklin and Haibin Zhang

Dept. of Computer Science, University of California, Davis, California 95616, USA
{franklin,hbzhang}@cs.ucdavis.edu

Abstract. We propose unique ring signatures that simplify and capture the spirit of linkable ring signatures. We use new techniques to provide an instantiation which can be *tightly* related to the DDH problem in the random oracle model, leading to the most efficient linkable/unique ring signature.

Keywords: anonymity, authentication, e-voting system, provable security, ring signature, tight reduction, unique signature, verifiable random function.

1 Introduction

Ring signatures [23] are very useful tools for many privacy-preserving applications. However, they are not adequate in settings where some degree of privacy for users must be balanced against limited access. For example, a service provider might have the list of public keys that correspond to all users that have purchased a single access to some confidential service for that day (requiring anonymous authentication). For this kind of application, a number of restricted-use ring signatures are proposed. Notable examples include *linkable ring signatures* [19,26,20,25,1,7] and *traceable ring signatures* [13,14].

Linkable ring signature asks that if a user signs any two messages (same or different) with respect to the same ring, then an efficient public procedure can verify that the signer was the same (although the user's identity is not revealed).

Traceable ring signature is a ring signature scheme where each message is signed not only with respect to a list of ring members, but also with respect to an *issue* (e.g., identifying label of a specific election or survey). If a user signs any two different messages with respect to the same list of ring members *and* the same issue label, then the user's identity is revealed by an efficient public procedure. If a user signs the same message twice with respect to the same list of ring members *and* the same issue label, then the two signed messages can be determined to have come from the same signer by an efficient public procedure (although the signer's identity remains concealed).

Both linkable ring signatures and traceable ring signatures admit interesting applications such as various *e-voting systems* and *e-token systems*, and so on. Notably, the e-voting schemes *directly* from linkable or traceable ring signatures

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 162–170, 2013.

do *not* need any central authorities, a unique and desirable property in sharp contrast to all the schemes from other methods.

Unique ring signatures. We define *unique ring signatures* that capture the essence of linkable ring signatures and traceable ring signatures without identity revelation. We may say a ring signature scheme *unique* if whenever a signer produces two different ring signatures of the *same message* with respect to the same ring, such that both will pass the verification procedure, then these two ring signatures will always have a large common component (hereinafter *unique identifier*). For all the applications introduced in this paper, we further need a *non-colliding* property for a unique ring signature. Call a unique ring signature non-colliding if two different signers of the same message, almost never produce ring signatures with the same unique identifier.

Our contributions. We provide an efficient instantiation of unique ring signature in the random oracle model (ROM). Security of the scheme can be *tightly* reduced to the DDH problem (where, by "tight," it means that the success probability of some adversary in some time is roughly equal to the probability of solving some hard problem within almost the same period of time). Despite the similarities with the linkable ring signature due to Liu, Wei, and Wong [19], our construction employs a proof technique fundamentally different from the Cramer-Damgård-Schoemaker (CDS) type of ring signatures [8,17] which rely on "rewinding". Namely, our proof does not require *proof of knowledge* but heavily relies on zero-knowledge proof of *membership*. Tight reduction usually comes at a cost, but it turns out that our scheme has a tight reduction without sacrificing on efficiency. In toto, this scheme gives the most efficient linkable/unique ring signature in the ROM, in terms of key generation, signing, and verification algorithms.

Typically, one evaluates provably secure signature schemes from *three* perspectives: *efficiency*, indicating how fast the scheme can be implemented, which has an immediate impact on its genuine utility; *concrete security reduction*, which gives explicit bounds on success probability of the adversary, enabling meaningful comparisons for a given level of provable security; and *cryptographic assumptions*, preferably being simple, standard, and well-studied, on which the security of the scheme relies. A *desirable* provably secure cryptographic signature, commonly recognized, whether in the random oracle standard or the standard model, should be *at first* efficient, and could be *as well* tightly related to a reasonable assumption. Of course, it is also desirable to consider various tradeoffs among the three factors, provided that the scheme is still sufficiently efficient.

For signature schemes based on discrete logarithm problems, the most efficient scheme is the Schnorr signature [24] that is proven secure in the ROM under the DL assumption by Pointcheval and Stern [22]. The technique used is the Forking Lemma: by *rewinding* the forger $\mathcal{O}(q_h/\varepsilon)$ times, where q_h denotes the number of the forger makes to the random oracles and ε denotes its success probability one can compute the discrete logarithm of the public key. The reduction is unfortunately too loose. To obtain tight security reductions for the DL-based signature schemes, a number of constructions that are less efficient or/and under *stronger*

assumptions are proposed, including the EDL scheme by Goh and Jarecki [16] (under the CDH assumption), subsequent work by Chevallier-Mames [6] (under the CDH assumption), two schemes by Katz and Wang [18] (from the CDH and DDH problems respectively), and Fischlin's scheme [10] (that relies on the DL assumption but is relatively inefficient).

Turning to the DL-type ring signature schemes, tight reductions are more challenging to achieve. This is due, first, to the fact that all the DL based ones, to the best of our knowledge, follow the CDS paradigm [8] whose security seems to inevitably rely on the (generalized) rewinding technique (see, e.g., [17]). This is further due to the fact that the ring signature runs in the multi-user setting such that the reduction might *naturally* lose a factor of n which denotes the number of users in the ring. Last, we emphasize that ring signatures (in general) have multiple security notions such as unforgeability, anonymity, and possibly some others (see [3]). Tight reductions (to possibly different assumptions) here should be satisfied for *all* the required security notions. To put it differently, the security notion with the loosest reduction and the strongest assumption is the benchmark against which the security of the system can be measured.

The linkable ring signature [19] from the DDH assumption inherit the CDS framework and its analysis for ordinary ring signatures. In particular, if we let ε be an upper bound on the probability that the DL problem can be solved, then the success probability of any adversaries attacking the unforgeability is roughly $nq_h\varepsilon$, but for anonymity one has to rely on the potentially stronger DDH assumption. Similar results hold for the traceable ring signature [13], where Fujisaki and Suzuki therefore consider using Fischlin's technique [10, Remark 5.7] to improve the reduction tightness at a notable cost.

Instead, our random oracle based scheme has security tightly reduced to the DDH problem for *each* of the security notions, which implies that the scheme is *as secure as* the DDH problem. One main reason our scheme has tight reductions is the use of NIZK proof of membership, instead of the conventional proof of knowledge such that one has to rewind the forger for sufficient times.

For standard signature and ring signature schemes, to obtain tighter security, they necessarily become less efficient or rely on stronger assumptions. In contrast, our unique ring signature scheme is as efficient as the previous scheme [19] with a loose reduction. Notice that the PRF part not only enables NIZK proof of membership but *happens* to serve as the unique identifier.

2 Unique Ring Signature Model

We begin by recalling the definition of a *ring signature* scheme $\mathcal{RS} = (\mathsf{RK}, \mathsf{RS}, \mathsf{RV})$ that consists of three algorithms:

- $\mathsf{RK}(1^\lambda)$. The randomized *user key generation* algorithm takes as input the security parameter λ and outputs a public key pk and a secret key sk.
- $\mathsf{RS}(sk, R, m)$. The probabilistic *ring signing* algorithm takes as input a user secret key sk, a ring R that is a set of public keys (such that $pk \in R$), and a message m to return a signature σ on m with respect to the ring R.

- $\mathsf{RV}(R, m, \sigma)$. The deterministic *ring verification* algorithm takes as input a ring R, a message m, and a signature σ for m to return a single bit b.

The following correctness condition is required: for any security parameter λ, any integer n, any $\{(pk_i, sk_i)\}_1^n \xleftarrow{\$} \mathsf{RK}(1^\lambda)$ (where now $R = \{pk_i\}_1^n$), any $i \in [n]$, and any m, it holds that $\mathsf{RV}(R, m, \mathsf{RS}(R, sk_i, m)) = 1$.

We consider *unique ring signature* where the signature should have the form of $(R, m, \sigma) = (R, m, \tau, \pi)$ where τ is the *unique identifier* for some message m and some signer i, and π is the rest of the signature. For our constructions, one may simply consider that τ is *the* signature, and π is the corresponding (maybe probabilistic) proof of correctness. Following the recent formulation for ring signature due to Bender, Katz, and Morselli [3], we define for unique ring signature three security requirements: uniqueness, anonymity, and unforgeability. The way we define uniqueness property largely follows from that for unique group signature [11], where the uniqueness security is coupled to a non-colliding property. The formalization of the definitions of security can be found in [12].

3 Unique Ring Signature in Random Oracle Model

We start by describing our basic underlying signature/VRF scheme, and then give the construction of unique ring signature. Notice that our proof techniques do not require *proof of knowledge* but heavily rely on zero-knowledge proof of *membership*, which is one of the main reasons our signature enjoys tight security reductions and admits an improvement in efficiency for a given level of security.

The Underlying VRF Scheme. The signature we shall describe is first predicated on a (well-known) observation that given a random public group element $y = g^x$, the function $F(m) := H(m)^x$ is a PRF, if we model the hash function $H(\cdot)$ as a random oracle.

Our scheme is furthermore based on a well-known zero-knowledge proof system for equality of discrete logarithm due to Chaum and Pederson [5]:

A prover and a verifier both know (g, h, y_1, y_2) with $g, h \neq 1$ and $y_1 = g^x$ and $y_2 = h^x$ for an exponent $x \in \mathbb{Z}_q$. A prover also knows the exponent x. They run the following protocol:

1. The prover chooses $r \xleftarrow{\$} \mathbb{Z}_q$ and sends $a \leftarrow g^r$, $b \leftarrow h^r$ to the verifier.
2. The verifier sends a challenge $c \xleftarrow{\$} \mathbb{Z}_q$ to the prover.
3. The prover sends $t \leftarrow r - cx \bmod q$ to the verifier.
4. The verifier accepts iff $a = g^t y_1^c$ and $b = h^t y_2^c$.

The above protocol is a *sound* proof system but also *honest-verifier zero-knowledge* (HVZK). By using Fiat-Shamir transformation [13], it becomes a NIZK proof system if we model the hash function as a random oracle. Given the above PRF and NIZK proof system, we apply the Bellare-Goldwasser (BG) paradigm [2] to obtain a VRF scheme depicted in Figure 1. (The scheme is in fact a PRF with a NIZK proof and of course a secure signature scheme.) Note that the function that maps x

to g^x is not a commitment scheme: the binding property is satisfied while the hiding property is not. This prevents us from following the general BG construction's proof strategy exactly. However, under the DDH assumption, this can be proven secure with a similar proof to that of BG signature.

$\mathsf{Setup}(1^\lambda)$.

The setup algorithm takes as input the security parameter λ and outputs a multiplicative group $\mathbb{G}$ of prime order q and a randomly chosen generator g of $\mathbb{G}$. It also provides two hash functions $H\colon \{0,1\}^* \to \mathbb{G}$ and $H'\colon \{0,1\}^* \to \mathbb{Z}_q$. It outputs the public parameters as

$$\mathsf{pp} = (\lambda, q, \mathbb{G}, H, H').$$

$\mathsf{Gen}(1^\lambda, \mathsf{pp})$.

The key generation algorithm takes as input the parameter pp and chooses $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $y \leftarrow g^x$. It outputs the public key as $pk = y$ and the secret key as $sk = x$.

$\mathsf{Sig}(sk, m)$.

To sign the message m, the signer selects $r \xleftarrow{\$} \mathbb{Z}_q$ and computes

$$(m, H(m)^x, c, t),$$

where $c \leftarrow H'(m, g^r, H(m)^r)$ and $t \leftarrow r - cx \bmod q$.

$\mathsf{Vrf}(sk, m, \sigma)$.

The verification algorithm first parses σ as (m, τ, c, t) and checks if

$$c = H'(m, g^t y^c, H(m)^t \tau^c).$$

Fig. 1. Efficient Signature/VRF from the DDH assumption in the random oracle model. The algorithms are described in the context of digital signature. It is also a VRF scheme, where $\mathcal{VRF}.\mathsf{Eva}(sk, m) = H(m)^x$, $\mathcal{VRF}.\mathsf{Prove}(sk, m) = (c, t)$, and $\mathcal{VRF}.\mathsf{Ver}(m, \sigma) = \mathcal{DS}.\mathsf{Vrf}(m, \sigma)$.

Extending the underlying proof system. We now extend the underlying NIZK proof to an "or" language—a proof system that a unique identifier τ (for a message m and a ring R) has the same logarithm with respect to base $H(m||R)$ as one of the public keys $y_j := g^{x_j}$ $(j \in [n])$ with respect to base g. Assume, without loss of generality, $\log_{H(m||R)} \tau = \log_g y_i$ and the prover knows x_i. In particular, we use the proof system between a prover and a verifier.

1. For $j \in [n]$ and $j \neq i$, the prover selects $c_j, t_j \xleftarrow{\$} \mathbb{Z}_q$ and computes $a_j \leftarrow g^{t_j} y_j^{c_j}$ and $b_j \leftarrow H(m)^{t_j} (H(m)^{x_i})^{c_j}$; for $j = i$, the prover selects $r_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $a_i \leftarrow g^{r_i}$ and $b_i \leftarrow H(m)^{r_i}$. It sends $\{a_j, b_j\}_1^n$ to the verifier.
2. The verifier sends a challenge $c \xleftarrow{\$} \mathbb{Z}_q$ to the prover.
3. The prover computes $c_i \leftarrow c - \sum_{j \neq i} c_j$ and $t \leftarrow r - c_i x_i \bmod q$, and sends $c_1, t_1, \cdots, c_n, t_n$ to the verifier.
4. The verifier accepts iff $a_j = g^{t_j} y_j^{c_j}$ and $b_j = H(m)^{t_j} \tau^{c_j}$ for every $j \in [n]$.

The above protocol combines the Chaum-Pederson (CP) technique for proving the equality of two discrete logarithms of [5] and Cramer-Damgård-Schoenmakers (CDS) transformation [8]. Since both of the conversions "preserve" the properties of Σ-protocols, the above system is a sound proof system,[1] and also an interactive honest-verifier zero-knowledge of membership. However, as far as we are concerned, its soundness property has never been used in any signature schemes related to the above proof system. (This is perhaps due to the fact no one needs this property in these schemes anyway.) We now prove that the above proof system is sound;[2] in particular, even an arbitrarily malicious prover P^* cannot convince the verifier to accept a false statement.

Proof. The goal is to show that if $\log_{H(m)} \tau \neq \log_g y_j$ for every $j \in [n]$, then given any $\{a_j, b_j\}_1^n$ sent by P^* there is at most one value c for which P^* can respond correctly. Recall above that we let x_0 denote $\log_{H(m)} \tau$ and x_j denote $\log_g y_j$ for every $j \in [n]$. In this case, we have that $x_0 \neq x_j$ $(j \in [n])$. Given any $\{a_j, b_j\}_1^n$ (where we assume $a_j = g^{r_j}$ and $b_j = H(m)^{r'_j}$) sent to the verifier by a cheating prover, we have the following: if the verifier is to accept, then we must have that

$$c = \sum_1^n c_j, \tag{1}$$

and for every $j \in [n]$,

$$a_j = g^{t_1} y_j^{c_j}, \tag{2}$$

$$b_j = H(m)^{t_j} \tau^{c_j}. \tag{3}$$

By (2) and (3) we obtain that for every $j \in [n]$,

$$r_j = t_j + x_j c_j, \tag{4}$$

$$r'_j = t_j + x_0 c_j. \tag{5}$$

Noting that $x_0 \neq x_j$ for every $j \in [n]$, we have $c_j \leftarrow (r_j - r'_j)(x_o - x_j)^{-1}$ mod q. According to equation (1), we can now conclude that there is at most one challenge which the cheating prover can respond to. Therefore, the verifier generates this challenge with probability $1/q$ and the proof now follows.

If we turn the above system into a NIZK proof system by following Fiat-Shamir transformation through a hash function H' then one can check that the soundness property is bounded by q_h/q, where q_h denotes the number of times the adversary makes to the random oracle H'. Indeed, in this case, for any $\{a_j, b_j\}_1^n$ and any query $H(m, \{a_j, b_j\}_1^n)$ made by an adversary P^*, it follows from the

[1] Strictly speaking, Σ-protocols can be divided into two categories: Σ-protocols for proof of knowledge, and Σ-protocols for proof of membership. In particular, we can formally show, in the setting of proof of membership, the special soundness property implies that a Σ-protocol is always an interactive proof system.

[2] This is needed, since we shall be providing the exact bound on the soundness property in the random oracle model which appears in our full paper [12].

above proof that there is at most one possible value of c satisfying the verification equations. The unique ring signature (from the DDH assumption in the ROM) is described in Figure 2.

$\mathsf{Setup}(1^\lambda)$.

The setup algorithm takes as input the security parameter λ and outputs a multiplicative group $\mathbb{G}$ of prime order q and a randomly chosen generator g of $\mathbb{G}$. It also provides two hash functions $H\colon \{0,1\}^* \to \mathbb{G}$ and $H'\colon \{0,1\}^* \to \mathbb{Z}_q$. It outputs the public parameters as

$$\mathsf{pp} = (\lambda, q, \mathbb{G}, H, H').$$

$\mathsf{RG}(1^\lambda, \mathsf{pp})$.

The key generation algorithm for user i takes as input the parameter pp and selects an element $x_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $y_i \leftarrow g^{x_i}$. It outputs the public key as $pk_i = (\mathsf{pp}, y_i)$ and the secret key as $sk_i = (\mathsf{pp}, x_i)$.

$\mathsf{RS}(sk_i, R, m)$.

To sign the message m in the ring $R = (pk_1, ..., pk_n)$, the signer i with the secret key $sk_i = x_i$ generates the signature in the following way:

1. (Simulation step.) For $j \in [n]$ and $j \neq i$, select $c_j, t_j \xleftarrow{\$} \mathbb{Z}_q$ and compute $a_j \leftarrow g^{t_j} y_j^{c_j}$ and $b_j \leftarrow H(m||R)^{t_j} (H(m||R)^{x_i})^{c_j}$.
2. For $j = i$, select $r_i \xleftarrow{\$} \mathbb{Z}_q$ and compute $a_i \leftarrow g^{r_i}$ and $b_i \leftarrow H(m||R)^{r_i}$.
3. Let $c_i \leftarrow H'(m, R, \{a_j, b_j\}_1^n) - \sum_{j \neq i} c_j \bmod q$ and $t_i \leftarrow r_i - c_i x_i \bmod q$.
4. Return $(R, m, H(m||R)^{x_i}, c_1, t_1, \cdots, c_n, t_n)$.

$\mathsf{RV}(R, m, \sigma)$.

On receiving the signature (R, m, σ), the verification algorithm first parses σ as $(\tau, c_1, t_1, \cdots, c_n, t_n)$ and checks if

$$\textstyle\sum_1^n c_j = H'(m, R, \{g^{t_j} y_j^{c_j}, H(m||R)^{t_j} \tau^{c_j}\}_1^n).$$

Fig. 2. Unique ring signature from the DDH assumption in the ROM

The following theorem establishes the security of the above scheme (with proof and exact security bounds in our full paper [12]).

Theorem 1. *The scheme presented in this section is a unique ring signature in the random oracle model under the DDH assumption.* ∎

Acknowledgments. The authors thank Tsz Hon Yuen and anonymous reviewers for comments.

References

1. Au, M., Chow, S., Susilo, W., Tsang, P.: Short linkable ring signatures revisited. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 101–115. Springer, Heidelberg (2006)
2. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, Heidelberg (1990)

3. Bender, A., Katz, J., Morselli, R.: Ring signatures: stronger definitions, and constructions without random oracles. Journal of Cryptology 22(1), 114–138 (2009)
4. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
5. Chaum, D., Pedersen, T.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
6. Chevallier-Mames, B.: An efficient CDH-based signature scheme with a tight security reduction. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 511–526. Springer, Heidelberg (2005)
7. Chow, S.S.M., Susilo, W., Yuen, T.H.: Escrowed linkability of ring signatures and its applications. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 175–192. Springer, Heidelberg (2006)
8. Cramer, R., Damgård, I., 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)
9. 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)
10. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
11. Franklin, M., Zhang, H.: Unique group signatures. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 643–660. Springer, Heidelberg (2012); Full version in Cryptology ePrint Archive: Report 2012/204, `http://eprint.iacr.org`
12. Franklin, M., Zhang, H.: A Framework for Unique Ring Signatures. Cryptology ePrint Archive: Report 2012/577, `http://eprint.iacr.org`
13. Fujisaki, E., Suzuki, K.: Traceable ring signature. IEICE Transactions 91-A(1), 83–93 (2008)
14. Fujisaki, E.: Sub-linear size traceable ring signatures without random oracles. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 393–415. Springer, Heidelberg (2011)
15. Goh, E., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight security reductions to the Diffie-Hellman problems. J. of Cryptology 20(4), 493–514 (2007)
16. Goh, E., Jarecki, S.: A signature scheme as secure as the Diffie-Hellman problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 401–415. Springer, Heidelberg (2003)
17. Herranz, J., Sáez, G.: Forking lemmas for ring signature schemes. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 266–279. Springer, Heidelberg (2003)
18. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: CCS 2003, pp. 155–164. ACM Press (2003)
19. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004)
20. Liu, J.K., Wong, D.S.: Linkable ring signatures: Security models and new schemes. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005)

21. Micali, S., Reyzin, L.: Improving the exact security of digital signature schemes. J. Cryptology 15(1), 1–18 (2002)
22. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology 13(3), 361–396 (2000)
23. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret: Theory and applications of ring signatures. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science. LNCS, vol. 3895, pp. 164–186. Springer, Heidelberg (2006)
24. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
25. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for E-voting, E-cash and attestation. In: Deng, R.H., Bao, F., Pang, H., Zhou, J. (eds.) ISPEC 2005. LNCS, vol. 3439, pp. 48–60. Springer, Heidelberg (2005)
26. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 384–398. Springer, Heidelberg (2004)

Aggregating CL-Signatures Revisited: Extended Functionality and Better Efficiency

Kwangsu Lee[1,2,⋆], Dong Hoon Lee[1,⋆⋆], and Moti Yung[2,3]

[1] CIST, Korea University, Korea
{guspin,donghlee}@korea.ac.kr
[2] Columbia University, USA
{kwangsu,moti}@cs.columbia.edu
[3] Google Inc., USA

Abstract. Aggregate signature is public-key signature that allows anyone to aggregate different signatures generated by different signers on different messages into a short (called aggregate) signature. The notion has many applications where compressing the signature space is important: in infrastructure: secure routing protocols, in security: compressed certificate chain signature, in signing incrementally changed data: such as software module authentications, and in transaction systems: like in secure high-scale repositories and logs, typical in financial transactions. In spite of its importance, the state of the art of the primitive is such that it has not been easy to devise a suitable aggregate signature scheme that satisfies the conditions of real applications, with reasonable parameters: short public key size, short aggregate signatures size, and efficient aggregate signing/verification. In this paper, we propose two aggregate signature schemes based on the Camenisch-Lysyanskaya (CL) signature scheme whose security is reduced to that of CL signature (i.e., secure under the LRSW assumption) which substantially improve efficiency conditions for real applications. The first scheme is an "efficient sequential aggregate signature" scheme with the shortest size public key, to date, and very efficient aggregate verification. The second scheme is an "efficient synchronized aggregate signature" scheme with a very short public key size, and with the shortest (to date) size of aggregate signatures among synchronized aggregate signature schemes. Signing and aggregate verification are very efficient. Furthermore, our schemes are compatible: a signer of our aggregate signature schemes can dynamically use two modes of aggregation "sequential" and "synchronized," employing the same private/public key.

Keywords: Public-key signature, Aggregate information applications, Aggregate signature, CL signature, Bilinear map.

⋆ Supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the C-ITRC (Convergence Information Technology Research Center) support program (NIPA-2013-H0301-13-3007) supervised by the NIPA (National IT Industry Promotion Agency).

⋆⋆ Supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2010-0029121).

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 171–188, 2013.

1 Introduction

Public-key signature (PKS) is a central cryptographic primitive with numerous applications. However, constructing a PKS scheme that is efficient, secure, and flexible enough for a range of possible applications is not easy. Among such schemes, CL signature, proposed by Camenisch and Lysyanskaya [12], is one of the pairing-based signature schemes [8, 10, 12, 24] that satisfies these conditions. It was widely used as a basic component in various cryptosystems such as anonymous credential systems, group signature, RFID encryption, batch verification signature, ring signature [2, 3, 5, 11, 12], as well as in aggregate signature [22].

Pubic-key aggregate signature (PKAS), introduced by Boneh, Gentry, Lynn, and Shacham [9], is a special type of PKS that enables anyone to aggregate different signatures generated by different signers on different messages into a short aggregate signature. Boneh et al. proposed the first full aggregate signature scheme in bilinear groups and proved its security in the random oracle model under the CDH assumption. After the introduction of aggregate signatures, various types of aggregate signatures such as sequential aggregate signatures [6, 15, 17–19] and synchronized aggregate signatures [1, 14] were proposed. PKAS has numerous applications. In network and infrastructure: secure routing protocols, public-key infrastructure systems (signing certificate chains), sensor network systems, proxy signatures, as well as in applications: dynamically changing document composition (in particular, secure updating of software modules), secure transaction signing, secure work flow, and secure logs and repositories [1, 6, 7, 9]. In all these applications, compressing the space consumed by signatures is the major advantage. Note that in the area of financial transactions, in particular, logs and repositories are very large due to regulatory requirements to hold records for long time periods. The effect of compressing signatures by aggregation increases with the number of data items; thus it is quite plausible that the financial sector may find variations of aggregate signature most useful.

Though PKAS can reduce the size of signers' signatures by using the aggregation technique, it cannot reduce the size of signers' public keys since the public keys are not aggregated. Thus, the total information the verifier needs to access is still proportional to the number of signers in the aggregate signature, since the verifier should retrieve all public keys of signers from a certificate storage. Therefore, it is very important to reduce the size of public keys. An ideal solution for this problem is to use identity-based aggregate signature (IBAS) that represents the public key of a signer as an identity string. However, IBAS requires a trust structure different from public key infrastructure, namely, the existence of an additional trusted authority, (the current IBAS schemes are in [6, 14, 15] and are all secure in the random oracle model.) To construct a PKAS scheme with short public keys, Schröder proposed a sequential aggregate signature scheme with short public keys based on the CL signature scheme [22]. In the scheme of Schröder, the public key consists of two group elements and the aggregate signature consists of four group elements, but the aggregate verification algorithm requires l pairing operations and l exponentiations where l is the number of signers in the aggregate signature. Therefore, this work, while nicely pointing at the

CL signature as a source of efficiency for the context of aggregate signatures, still leaves out desired properties to build upon while exploiting the flexibility of the CL signature: can we make the public key shorter? can we require substantially less work in verification? and, can we build other modes of aggregate signatures? While asking such questions, we revisit the subject of aggregate signature based on CL signatures.

1.1 Our Contributions

In this paper, we indeed solve the problem of constructing a PKAS scheme that has short public keys, short aggregate signatures, and an efficient aggregate verification algorithm. We first propose an efficient sequential aggregate signature scheme based on the CL signature scheme and prove its security based on that of CL signature (i.e., the LRSW assumption) without random oracles. A sequential aggregate signature assumes that the aggregation mode is done in linear order: signed message after signed message. In this scheme, the public key consists of just one group element and the aggregate signature consists of just three group element. The size of the public key is the shortest among all sequential aggregate schemes to date (except IBAS schemes). The aggregate verification algorithm of our scheme is quite efficient since it just requires five pairing operations and l exponentiations (or multi-exponentiations). Therefore our scheme simultaneously satisfies the conditions of short public keys, short aggregate signatures, and efficient aggregate verification.

Next, we propose an efficient synchronized aggregate signature scheme based on the CL signature scheme and prove its security based on the CL signature security in the random oracle model (the random oracle can be removed if the number of messages is restricted to be polynomial). Synchronized aggregate signature is a mode where the signers of messages to be aggregated are synchronized, but aggregation can take any order. In this scheme, the public key consists of just one group element and the aggregate signature consists of one group element and one integer. The size of the aggregate signature is the shortest among all synchronized aggregate signature schemes to date. The aggregate verification algorithm of this scheme is also quite efficient since it just requires three pairing operations and l exponentiations (or multi-exponentiations).

Additionally, our two aggregate signature schemes can be combined to give a new notion of *aggregate "multi-modal" signature scheme*: A scheme which supports, both, sequential aggregation or synchronized aggregation, since the public key and the private key of two schemes are the same. This property can increase the utility and flexibility of the suggested scheme(s).

1.2 Related Work

Given the importance of aggregation to saving signature space, much work has been invested in the various notions allowing aggregation.

Full Aggregation. The notion of public-key aggregate signature (PKAS) was introduced by Boneh, Gentry, Lynn, and Shacham [9]. They proposed the first

PKAS scheme in bilinear groups that supports full aggregation such that anyone can freely aggregate different signatures signed by different signers on different messages into a short aggregate signature [9]. The PKAS scheme of Boneh et al. requires l number of pairing operations in the aggregate verification algorithm where l is the number of signers in the aggregate signature. Bellare et al. modified the PKAS scheme of Boneh et al. to remove the restriction such that the message should be different by hashing a message with the public key of a signer [4].

Sequential Aggregation. The concept of sequential aggregate signature was introduced by Lysyanskaya, Micali, Reyzin, and Shacham [19]. In sequential aggregate signature, a signer can generate an aggregate signature by adding his signature to the previous aggregate signature that was received from a previous signer. Lysyanskaya et al. proposed a sequential PKAS scheme using certified trapdoor permutations, and they proved its security in random oracle models [19]. Boldyreva et al. proposed an identity-based sequential aggregate signature (IBSAS) scheme (in the trust model of identity-based schemes with a trusted private keys authority), in bilinear groups and proved its security in the random oracle model under an interactive assumption [6]. Recently, Gerbush et al. showed that a modified IBSAS scheme of Boldyreva et al. in composite order bilinear groups can be secure in the random oracle model under static assumptions [15].

The first sequential PKAS scheme without random oracles was proposed by Lu et al. [18]. They constructed a sequential PKAS scheme based on the PKS scheme of Waters and proved its security without random oracles under the CDH assumption. However, this sequential PKAS scheme has a disadvantage such that the size of public keys is very long. To reduce the size of pubic keys in PKAS schemes, Schröder proposed the CL signature based scheme discussed above [22]. Recently, Lee et al. proposed an efficient sequential PKAS scheme with short public keys and proved its security without random oracles under static assumptions [17].

Synchronized Aggregation. The concept of synchronized aggregate signature was introduced by Gentry and Ramzan [14]. In synchronized aggregate signature, all signers have synchronized time information and individual signatures generated by different signers within the same time period can be aggregated into a short aggregate signature. They proposed an identity-based synchronized aggregate signature scheme in bilinear groups and proved its security in the random oracle model under the CDH assumption [14]. Ahn et al. proposed an efficient synchronized PKAS scheme based on the PKS scheme of Hohenberger and Waters and proved its security without random oracles under the CDH assumption [1].

2 Preliminaries

In this section, we define bilinear groups, and introduce the LRSW assumption which is associated with the security of the CL signature scheme, which is, then, presented as well.

2.1 Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $\exists g$ such that $e(g, g)$ has order p, that is, $e(g, g)$ is a generator of $\mathbb{G}_T$.

We say that $\mathbb{G}, \mathbb{G}_T$ are bilinear groups if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable.

2.2 Complexity Assumption

The security of our aggregate signature schemes is based on the following LRSW assumption. The LRSW assumption was introduced by Lysyanskaya et al. [20] and it is secure under the generic group model defined by Shoup [23] (and adapted to bilinear groups in [12]).

Assumption 1 (LRSW). *Let $\mathcal{G}$ be an algorithm that on input the security parameter 1^λ, outputs the parameters for a bilinear group as $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Let $X, Y \in \mathbb{G}$ such that $X = g^x, Y = g^y$ for some $x, y \in \mathbb{Z}_p$. Let $O_{X,Y}(\cdot)$ be an oracle that on input a value $M \in \mathbb{Z}_p$ outputs a triple (a, a^y, a^{x+Mxy}) for a randomly chosen $a \in \mathbb{G}$. Then for all probabilistic polynomial time adversaries $\mathcal{A}$,*

$$\begin{aligned}\Pr[&(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^\lambda), x \leftarrow \mathbb{Z}_p, y \leftarrow \mathbb{Z}_p, X = g^x, Y = g^y,\\ &(M, a, b, c) \leftarrow \mathcal{A}^{O_{X,Y}(\cdot)}(p, \mathbb{G}, \mathbb{G}_T, e, g, X, Y) :\\ &M \notin Q \wedge M \in \mathbb{Z}_p^* \wedge a \in \mathbb{G} \wedge b = a^y \wedge c = a^{x+Mxy}] < 1/poly(\lambda)\end{aligned}$$

where Q is the set of queries that $\mathcal{A}$ made to $O_{X,Y}(\cdot)$.

2.3 The CL Signature Scheme

The CL signature scheme is a public-key signature scheme that was proposed by Camenisch and Lysyanskaya [12] and the security was proven without random oracles under the LRSW assumption. Although the security of the CL signature scheme is based on this interactive assumption, it is flexible and widely used for the constructions of various cryptosystems [5, 11, 12, 20, 22] (this is so, perhaps due to its relatively elegant and simple algebraic structure).

PKS.KeyGen(1^λ): The key generation algorithm first generates the bilinear groups $\mathbb{G}, \mathbb{G}_T$ of prime order p of bit size $\Theta(\lambda)$. Let g be the generator of $\mathbb{G}$. It selects two random exponents $x, y \in \mathbb{Z}_p$ and sets $X = g^x, Y = g^y$. It outputs a private key as $SK = (x, y)$ and a public key as $PK = (p, \mathbb{G}, \mathbb{G}_T, e, g, X, Y)$.

PKS.Sign(M, SK): The signing algorithm takes as input a message $M \in \mathbb{Z}_p^*$ and a private key $SK = (x, y)$. It selects a random element $A \in \mathbb{G}$ and computes $B = A^y,\ C = A^x B^{xM}$. It outputs a signature as $\sigma = (A, B, C)$.

PKS.Verify(σ, M, PK)**:** The verification algorithm takes as input a signature $\sigma = (A, B, C)$ on a message $M \in \mathbb{Z}_p^*$ under a public key $PK = (p, \mathbb{G}, \mathbb{G}_T, e, g, X, Y)$. It verifies that $e(A, Y) \stackrel{?}{=} e(B, g)$ and $e(C, g) \stackrel{?}{=} e(A, X) \cdot e(B, X)^M$. If these equations hold, then it outputs 1. Otherwise, it outputs 0.

Theorem 2 ([12]). *The CL signature scheme is existentially unforgeable under a chosen message attack if the LRSW assumption holds.*

3 Sequential Aggregate Signature

In this section, we propose an efficient sequential aggregate signature (SeqAS) scheme based on the CL signature scheme, and prove its security without random oracles.

3.1 Definitions

Sequential aggregate signature (SeqAS) is a special type of public-key aggregate signature (PKAS) that allows each signer to sequentially add his signature on a different message to the aggregate signature [19]. That is, a signer with an index i receives an aggregate signature σ'_Σ from the signer of an index $i-1$, and he generates a new aggregate signature σ_Σ by aggregating his signature on a message M to the received aggregate signature. The resulting aggregate signature has the same size of the previous aggregate signature.

Formally, a SeqAS scheme consists of four PPT algorithms **Setup**, **KeyGen**, **AggSign**, and **AggVerify**, which are defined as follows:

- **Setup**(1^λ). The setup algorithm takes as input a security parameter 1^λ and outputs public parameters PP.
- **KeyGen**(PP). The key generation algorithm takes as input the public parameters PP, and outputs a public key PK and a private key SK.
- **AggSign**$(\sigma'_\Sigma, \mathbf{M}, \mathbf{PK}, M, SK, PP)$. The aggregate signing algorithm takes as input an aggregate-so-far σ'_Σ on messages $\mathbf{M} = (M_1, \ldots, M_k)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_k)$, a message M, and a private key SK with PP, and outputs a new aggregate signature σ_Σ.
- **AggVerify**$(\sigma_\Sigma, \mathbf{M}, \mathbf{PK}, PP)$. The aggregate verification algorithm takes as input an aggregate signature σ_Σ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$ and the public parameters PP, and outputs either 1 or 0 depending on the validity of the aggregate signature.

The correctness requirement is that for each PP output by **Setup**, for all (PK, SK) output by **KeyGen**, any M, we have that **AggVerify**(**AggSign**$(\sigma'_\Sigma, \mathbf{M}', \mathbf{PK}', M, SK, PK, PP), \mathbf{M}'||M, \mathbf{PK}'||PK, PP) = 1$ where σ'_Σ is a valid aggregate-so-far signature on messages $\mathbf{M}'$ under public keys $\mathbf{PK}'$.

The security model of SeqAS was introduced by Lysyanskaya et al. [19]. In this paper, we follow the security model that was proposed by Lu et al. [18]. The security model of Lu et al. is a more restricted model that requires the

adversary to correctly generate other signers' public keys and private keys except the challenge signer's key. To ensure the correct generation of public keys and private keys, the adversary should submit the corresponding private keys of the public keys to the challenger before using the public keys. A realistic solution of this is for the signer to prove that he knows the corresponding private key of the public key by using zero-knowledge proofs when he requests the certification of his public key.

Formally, the security notion of existential unforgeability under a chosen message attack is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

Setup: $\mathcal{C}$ first initializes a key-pair list $KeyList$ as empty. Next, it runs **Setup** to obtain public parameters PP and **KeyGen** to obtain a key pair (PK, SK), and gives PK to $\mathcal{A}$.

Certification Query: $\mathcal{A}$ adaptively requests the certification of a public key by providing a key pair (PK, SK). Then $\mathcal{C}$ adds the key pair (PK, SK) to $KeyList$ if the key pair is a valid one.

Signature Query: $\mathcal{A}$ adaptively requests a sequential aggregate signature (by providing an aggregate-so-far σ'_{Σ} on messages $\mathbf{M}'$ under public keys $\mathbf{PK}'$), on a message M to sign under the challenge public key PK, and receives a sequential aggregate signature σ_{Σ}.

Output: Finally (after a sequence of the above queries), $\mathcal{A}$ outputs a forged sequential aggregate signature σ^*_{Σ} on messages $\mathbf{M}^*$ under public keys $\mathbf{PK}^*$. $\mathcal{C}$ outputs 1 if the forged signature satisfies the following three conditions, or outputs 0 otherwise: 1) $\mathbf{AggVerify}(\sigma^*_{\Sigma}, \mathbf{M}^*, \mathbf{PK}^*, PP) = 1$, 2) The challenge public key PK must exist in $\mathbf{PK}^*$ and each public key in $\mathbf{PK}^*$ except the challenge public key must be in $KeyList$, and 3) The corresponding message M in $\mathbf{M}^*$ of the challenge public key PK must not have been queried by $\mathcal{A}$ to the sequential aggregate signing oracle.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{SeqAS}_{\mathcal{A}} = \Pr[\mathcal{C} = 1]$ where the probability is taken over all the randomness of the experiment. A SeqAS scheme is existentially unforgeable under a chosen message attack if all PPT adversaries have at most a negligible advantage (for large enough security parameter) in the above experiment.

3.2 Construction

We first describe the design idea of our SeqAS scheme. To construct a SeqAS scheme, we use the "public key sharing" technique such that the element Y in the public key of the original CL signature scheme can be shared with all signers. The modified CL signature scheme that shares the element Y of the public key is described as follows: The setup algorithm publishes the public parameters that contain the description of bilinear groups and an element Y. Each signer generates a private key $x \in \mathbb{Z}_p$ and a public key $X = g^x$. A signer who has the private key x of the public key X can generate an original CL signature

$\sigma = (A, B, C)$ on a message M just using the private key x and a random r as $A = g^r, B = Y^r$, and $C = A^x B^{xM}$ since the element Y is given in the public parameters.

We construct a SeqAS scheme based on the modified CL signature scheme that supports "public key sharing" by using the "randomness re-use" technique of Lu et al. [18]. It is easy to sequentially aggregate signatures if the element Y is shared with all signers since we only need to consider the aggregation of the $\{X_i\}$ values of signers instead of the $\{X_i, Y_i\}$ values of signers. For instance, the first signer who has a private key x_1 generates a signature $\sigma_1 = (A_1, B_1, C_1)$ on a message M_1 as $A_1 = g^{r_1}, B_1 = Y^{r_1}$, and $C_1 = (g^{r_1})^{x_1}(Y^{r_1})^{x_1 M_1}$. The second signer with a private key x_2 generates a sequential aggregate signature $\sigma_2 = (A_2, B_2, C_2)$ on a message M_2 as $A_2 = A_1, B_2 = B_1$, and $C_2 = C_1(A_1)^{x_2}(B_1)^{x_2 M_2}$ by using the "randomness re-use" technique. Therefore a sequential aggregate signature of signers is formed as $\sigma_\Sigma = (A = g^r, B = Y^r, C = A^{\sum x_i} B^{\sum x_i M_i})$. Additionally, each signer should re-randomize the aggregate signature to prevent a simple attack.

Our SeqAS scheme is described as follows:

SeqAS.Setup(1^λ): This algorithm first generates the bilinear groups $\mathbb{G}, \mathbb{G}_T$ of prime order p of bit size $\Theta(\lambda)$. Let g be the generator of $\mathbb{G}$. It chooses a random element $Y \in \mathbb{G}$ and outputs public parameters as $PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, Y)$.

SeqAS.KeyGen(PP): This algorithm takes as input the public parameters PP. It selects a random exponent $x \in \mathbb{Z}_p$ and sets $X = g^x$. Then it outputs a private key as $SK = x$ and a public key as $PK = X$.

SeqAS.AggSign($\sigma'_\Sigma, \mathbf{M}', \mathbf{PK}', M, SK, PP$): This algorithm takes as input an aggregate-so-far $\sigma'_\Sigma = (A', B', C')$ on messages $\mathbf{M}' = (M_1, \ldots, M_k)$ under public keys $\mathbf{PK}' = (PK_1, \ldots, PK_k)$ where $PK_i = X_i$, a message $M \in \mathbb{Z}_p^*$, and a private key $SK = x$ with PP. It first checks the validity of σ'_Σ by calling **AggVerify**$(\sigma'_\Sigma, \mathbf{M}', \mathbf{PK}', PP)$. If σ'_Σ is not valid, then it halts. It checks that the public key PK of SK does not already exist in $\mathbf{PK}'$. If the public key already exists, then it halts. Note that if $k = 0$, then $\sigma'_\Sigma = (g, Y, 1)$. It selects a random exponent $r \in \mathbb{Z}_p$ and computes

$$A = (A')^r, \ B = (B')^r, \ C = \left(C' \cdot (A')^x \cdot (B')^{xM}\right)^r.$$

It outputs an aggregate signature as $\sigma_\Sigma = (A, B, C)$.

SeqAS.AggVerify($\sigma_\Sigma, \mathbf{M}, \mathbf{PK}, PP$): This algorithm takes as input an aggregate signature $\sigma_\Sigma = (A, B, C)$ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$ where $PK_i = X_i$. It first checks that any M_i is in $\mathbb{Z}_p^*$, any public key does not appear twice in $\mathbf{PK}$, and any public key in $\mathbf{PK}$ has been certified. If these checks fail, then it outputs 0. If $l = 0$, then it outputs 1 if $\sigma_\Sigma = (1, Y, 1)$, 0 otherwise. Next, it verifies that

$$e(A, Y) \stackrel{?}{=} e(B, g) \ \text{ and } \ e(C, g) \stackrel{?}{=} e(A, \prod_{i=1}^{l} X_i) \cdot e(B, \prod_{i=1}^{l} X_i^{M_i}).$$

If these equations hold, then it outputs 1. Otherwise, it outputs 0.

A sequential aggregate signature $\sigma_\Sigma = (A, B, C)$ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$ has the following form

$$A = g^r,\ B = Y^r,\ C = \left(g^r\right)^{\sum_{i=1}^{l} x_i} \left(Y^r\right)^{\sum_{i=1}^{l} x_i M_i}$$

where $PK_i = X_i = g^{x_i}$.

3.3 Security Analysis

We prove the security of our SeqAS scheme based on the security of the CL signature scheme without random oracles.

Theorem 3. *The above SeqAS scheme is existentially unforgeable under a chosen message attack if the CL signature scheme is existentially unforgeable under a chosen message attack.*

Proof. The main idea of the security proof is that the aggregated signature of our SeqAS scheme is independent of the order of aggregation, and the simulator of the SeqAS scheme possesses the private keys of all signers except the private key of the challenge public key. That is, if the adversary requests a sequential aggregate signature, then the simulator first obtains a CL signature from the target scheme's signing oracle and runs the aggregate signing algorithm to generate a sequential aggregate signature. If the adversary finally outputs a forged sequential aggregate signature that is non-trivial, then the simulator extracts the CL signature of the challenge public key from the forged aggregate signature by using the private keys of other signers.

Suppose there exists an adversary $\mathcal{A}$ that forges the above SeqAS scheme with non-negligible advantage ϵ. A simulator $\mathcal{B}$ that forges the CL signature scheme is first given: a challenge public key $PK_{CL} = (p, \mathbb{G}, \mathbb{G}_T, e, g, X, Y)$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first constructs $PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, Y)$ and $PK^* = X$ from PK_{CL}. Next, it initializes a key-pair list $KeyList$ as an empty one and gives PP and PK^* to $\mathcal{A}$.

Certification Query: $\mathcal{A}$ adaptively requests the certification of a public key by providing a public key $PK_i = X_i$ and its private key $SK_i = x_i$. $\mathcal{B}$ checks the private key and adds the key pair (PK_i, SK_i) to $KeyList$.

Signature Query: $\mathcal{A}$ adaptively requests a sequential aggregate signature by providing an aggregate-so-far σ'_Σ on messages $\mathbf{M}' = (M_1, \ldots, M_k)$ under public keys $\mathbf{PK}' = (PK_1, \ldots, PK_k)$, and a message M to sign under the challenge private key of PK^*. $\mathcal{B}$ proceeds the aggregate signature query as follows:

1. It first checks that the signature σ'_Σ is valid by calling **SeqAS.AggVerify** and that each public key in $\mathbf{PK}'$ exits in $KeyList$.
2. It queries its signing oracle that simulates **PKS.Sign** on the message M for the challenge public key PK^* and obtains a signature σ.

3. For each $1 \leq i \leq k$, it constructs an aggregate signature on message M_i using **SeqAS.AggSign** since it knows the private key that corresponds to PK_i. The resulting signature is an aggregate signature for messages $\mathbf{M}'||M$ under public keys $\mathbf{PK}'||PK^*$ since this scheme does not check the order of aggregation. It gives the result signature σ_Σ to $\mathcal{A}$.

Output: $\mathcal{A}$ outputs a forged aggregate signature $\sigma^*_\Sigma = (A^*, B^*, C^*)$ on messages $\mathbf{M}^* = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK}^* = (PK_1, \ldots, PK_l)$ for some l. Without loss of generality, we assume that $PK_1 = PK^*$. $\mathcal{B}$ proceeds as follows:

1. It first checks the validity of σ^*_Σ by calling **SeqAS.AggVerify**. Additionally, the forged signature should not be trivial: the challenge public key PK^* must be in $\mathbf{PK}^*$, and the message M_1 must not be queried by $\mathcal{A}$ to the signature query oracle.
2. For each $2 \leq i \leq l$, it parses $PK_i = X_i$ from $\mathbf{PK}^*$, and it retrieves the private key $SK_i = x_i$ of PK_i from $KeyList$. It then computes

$$A = A^*,\ B = B^*,\ C = C^* \cdot \left((A^*)^{\sum_{i=2}^{l} x_i} (B^*)^{\sum_{i=2}^{l} x_i M_i} \right)^{-1}.$$

3. It outputs $\sigma^* = (A, B, C)$ on a message $M^* = M_1$ as a non-trivial forgery of the CL signature scheme since it did not make a signing query on M_1.

To finish the proof, we first show that the distribution of the simulation is correct. It is obvious that the public parameters and the public key are correctly distributed. The distribution of the sequential aggregate signatures is correct since this scheme does not check the order of aggregation. Finally, we can show that the resulting signature $\sigma^* = (A, B, C)$ of the simulator is a valid signature for the CL signature scheme on the message M_1 under the public key PK^* since it satisfies the following equation:

$$\begin{aligned} e(C, g) &= e(C^* \cdot \left((A^*)^{\sum_{i=2}^{l} x_i} (B^*)^{\sum_{i=2}^{l} x_i M_i} \right)^{-1}, g) \\ &= e((A^*)^{\sum_{i=1}^{l} x_i} (B^*)^{\sum_{i=1}^{l} x_i M_i} \cdot (A^*)^{-\sum_{i=2}^{l} x_i} (B^*)^{-\sum_{i=2}^{l} x_i M_i}, g) \\ &= e((A^*)^{x_1} (B^*)^{x_1 M_1}, g) = e(A^*, g^{x_1}) \cdot e(B^*, g^{x_1 M_1}) \\ &= e(A, X) \cdot e(B, X^{M^*}). \end{aligned}$$

This completes our proof. □

3.4 Discussions

Efficiency. The public key of our SeqAS scheme consists of just one group element and the aggregate signature consists of three group elements, since the public key element Y of the CL signature scheme is moved to the public parameters of our scheme. The aggregate signing algorithm requires one aggregate verification and five exponentiations, and the aggregate verification algorithm requires five pairing operations and l exponentiations where l is the number of signers in the aggregate signature. In the SeqAS scheme of Schröder [22], the

public key consists of two group elements, the aggregate signature consists of four group elements, and the aggregate verification algorithm requires l pairing operations and l exponentiations.

4 Synchronized Aggregate Signature

In this section, we propose an efficient synchronized aggregate signature (SyncAS) scheme based on the CL signature scheme, and prove its security in the random oracle model.

4.1 Definitions

Synchronized aggregate signature (SyncAS) is a special type of public-key aggregate signature (PKAS) that allows anyone to aggregate signer's signatures on different messages with a same time period into a short aggregate signature if all signers have the synchronized time period information like a clock [1,14]. In SyncAS scheme, each signer has a synchronized time period or has an access to public time information. Each signer can generate an individual signature on a message M and a time period w. Note that the signer can generate just one signature per one time period. After that, anyone can aggregate individual signatures of other signers into a short aggregate signature σ_Σ if the individual signatures are generated on the same time period w. The resulting aggregate signature has the same size of the individual signature.

Formally, a SyncAS scheme consists of six PPT algorithms **Setup**, **KeyGen**, **Sign**, **Verify**, **Aggregate**, and **AggVerify**, which are defined as follows:

- **Setup**(1^λ). The setup algorithm takes as input a security parameter 1^λ and outputs public parameters PP.
- **KeyGen**(PP). The key generation algorithm takes as input the public parameters PP, and outputs a public key PK and a private key SK.
- **Sign**(M, w, SK, PP). The signing algorithm takes as input a message M, a time period w, and a private key SK with PP, and outputs an individual signature σ.
- **Verify**(σ, M, PK, PP). The verification algorithm takes as input a signature σ on a message M under a public key PK, and outputs either 1 or 0 depending on the validity of the signature.
- **Aggregate**$(\mathbf{S}, \mathbf{M}, \mathbf{PK})$. The aggregation algorithm takes as input individual signatures $\mathbf{S} = (\sigma_1, \ldots, \sigma_l)$ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$, and outputs an aggregate signature σ_Σ.
- **AggVerify**$(\sigma_\Sigma, \mathbf{M}, \mathbf{PK}, PP)$. The aggregate verification algorithm takes as input an aggregate signature σ_Σ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$, and outputs either 1 or 0 depending on the validity of the aggregate signature.

The correctness requirement is that for each PP output by **Setup**, for all (PK, SK) output by **KeyGen**, any M, we have that **AggVerify**(**Aggregate**(**S**,

$\mathbf{M}, \mathbf{PK}), \mathbf{M}, \mathbf{PK}, PP) = 1$ where $\mathbf{S}$ is individual signatures on messages $\mathbf{M}$ under public keys $\mathbf{PK}$.

The security model of SyncAS was introduced by Gentry and Ramzan [14]. In this paper, we follow the security model that was proposed by Ahn et al. [1]. The security model of Ahn et al. is a more restricted model that requires the adversary to correctly generate other signers' public keys and private keys except the challenge signer's key. To ensure the correct generation of public keys and private keys, the adversary should submit the private key of the public key, or he should prove that he knows the corresponding private key by using zero-knowledge proofs.

Formally, the security notion of existential unforgeability under a chosen message attack is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

Setup: $\mathcal{C}$ first initializes a key-pair list $KeyList$ as empty. Next, it runs **Setup** to obtain public parameters PP and **KeyGen** to obtain a key pair (PK, SK), and gives PK to $\mathcal{A}$.

Certification Query: $\mathcal{A}$ adaptively requests the certification of a public key by providing a key pair (PK, SK). Then $\mathcal{C}$ adds the key pair (PK, SK) to $KeyList$ if the key pair is a valid one.

Hash Query: $\mathcal{A}$ adaptively requests a hash on a string for various hash functions, and receives a hash value.

Signature Query: $\mathcal{A}$ adaptively requests a signature on a message M and a time period w that was not used before to sign under the challenge public key PK, and receives an individual signature σ.

Output: Finally (after a sequence of the above queries), $\mathcal{A}$ outputs a forged synchronized aggregate signature σ^*_Σ on messages $\mathbf{M}^*$ under public keys $\mathbf{PK}^*$. $\mathcal{C}$ outputs 1 if the forged signature satisfies the following three conditions, or outputs 0 otherwise: 1) $\mathbf{AggVerify}(\sigma^*_\Sigma, \mathbf{M}^*, \mathbf{PK}^*, PP) = 1$, 2) The challenge public key PK must exist in $\mathbf{PK}^*$ and each public key in $\mathbf{PK}^*$ except the challenge public key must be in $KeyList$, and 3) The corresponding message M in $\mathbf{M}^*$ of the challenge public key PK must not have been queried by $\mathcal{A}$ to the signing oracle.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{SyncAS}_{\mathcal{A}} = \Pr[\mathcal{C} = 1]$ where the probability is taken over all the randomness of the experiment. A SyncAS scheme is existentially unforgeable under a chosen message attack if all PPT adversaries have at most a negligible advantage (for large enough security parameter) in the above experiment.

4.2 Construction

We first describe the design idea of our SyncAS scheme. In the previous section, we proposed a modified CL signature scheme that shares the element Y in the public parameters. The signature of this modified CL signature scheme is formed as $\sigma = (A = g^r, B = Y^r, C = A^x B^{xM})$. If we can force signers to use the same $A = g^r$ and $B = Y^r$ in signatures, then we easily obtain an aggregate signature

as $\sigma_\Sigma = (A = g^r, B = Y^r, C = A^{\sum x_i} B^{\sum x_i M_i})$ by just multiplying individual signatures of signers. In synchronized aggregate signatures, it is possible to force signers to use the same A and B since all signers have the same time period w. Therefore, each signer first sets $A = H(0||w)$ and $B = H(1||w)$ using the hash function H and the time period w, and then he generates an individual signature $\sigma = (C = A^x B^{xM}, w)$. We need to hash a message for the proof of security.

Let $\mathcal{W}$ be a set of time periods where $|\mathcal{W}|$ is fixed polynomial in the security parameter[1]. Our SyncAS scheme is described as follows:

SyncAS.Setup(1^λ): This algorithm first generates the bilinear groups $\mathbb{G}, \mathbb{G}_T$ of prime order p of bit size $\Theta(\lambda)$. Let g be the generator of $\mathbb{G}$. It chooses two hash functions $H_1 : \{0,1\} \times \mathcal{W} \to \mathbb{G}$ and $H_2 : \{0,1\}^* \times \mathcal{W} \to \mathbb{Z}_p^*$. It outputs public parameters as $PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, H_1, H_2)$.

SyncAS.KeyGen(PP): This algorithm takes as input the public parameters PP. It selects a random exponent $x \in \mathbb{Z}_p$ and sets $X = g^x$. Then it outputs a private key as $SK = x$ and a public key as $PK = X$.

SyncAS.Sign(M, w, SK, PP): This algorithm takes as input a message $M \in \{0,1\}^*$, a time period $w \in \mathcal{W}$, and a private key $SK = x$ with PP. It first sets $A = H_1(0||w), B = H_1(1||w), h = H_2(M||w)$ and computes $C = A^x B^{xh}$. It outputs a signature as $\sigma = (C, w)$.

SyncAS.Verify(σ, M, PK, PP): This algorithm takes as input a signature $\sigma = (C, w)$ on a message M under a public key $PK = X$. It first checks that the public key has been certified. If these checks fail, then it outputs 0. Next, it sets $A = H_1(0||w), B = H_1(1||w), h = H_2(M||w)$ and verifies that $e(C, g) \stackrel{?}{=} e(AB^h, X)$. If this equation holds, then it outputs 1. Otherwise, it outputs 0.

SyncAS.Aggregate($\mathbf{S}, \mathbf{M}, \mathbf{PK}, PP$): This algorithm takes as input signatures $\mathbf{S} = (\sigma_1, \ldots, \sigma_l)$ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$ where $\sigma_i = (C_i', w_i')$ and $PK_i = X_i$. It first checks that that w_1' is equal to w_i' for $i = 2$ to l. If it fails, it halts. Next, it sets $w = w_1'$ and computes $C = \prod_{i=1}^{l} C_i'$. It outputs an aggregate signature as $\sigma_\Sigma = (C, w)$.

SyncAS.AggVerify($\sigma_\Sigma, \mathbf{M}, \mathbf{PK}, PP$): This algorithm takes as input an aggregate signature $\sigma_\Sigma = (C, w)$ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$ where $PK_i = X_i$. It first checks that any public key does not appear twice in $\mathbf{PK}$ and any public key in $\mathbf{PK}$ has been certified. If these checks fail, then it outputs 0. Next, it sets $A = H_1(0||w), B = H_1(1||w)$, $h_i = H_2(M_i||w)$ for all $1 \le i \le l$ and verifies that

[1] The set $\mathcal{W}$ does not need to be included in PP since an integer w in the range $[1, T]$ can be used where T is fixed polynomial in the security parameter. In practice, we can set $T = 2^{32}$ if the maximum time period of certificates is 10 years and a signer generates a signature per each second. The previous SyncAS schemes [1,14] support exponential size of time periods while our SyncAS scheme supports polynomial size of time periods.

$$e(C, g) \stackrel{?}{=} e(A, \prod_{i=1}^{l} X_i) \cdot e(B, \prod_{i=1}^{l} X_i^{h_i}).$$

If this equation holds, then it outputs 1. Otherwise, it outputs 0.

A synchronized aggregate signature $\sigma_\Sigma = (C, w)$ on messages $\mathbf{M} = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK} = (PK_1, \ldots, PK_l)$ has the following form

$$C = H_1(0||w)^{\sum_{i=1}^{l} x_i} H_1(1||w)^{\sum_{i=1}^{l} x_i H_2(M_i||w)}$$

where $PK_i = X_i = g^{x_i}$.

4.3 Security Analysis

We prove the security of our SyncAS scheme based on the security of the CL signature scheme in the random oracle model.

Theorem 4. *The above SyncAS scheme is existentially unforgeable under a chosen message attack if the CL signature scheme is existentially unforgeable under a chosen message attack.*

Proof. The main idea of the security proof is that the random oracle model supports the programmability of hash functions, the adversary can request just one signature per one time period in this security model, and the simulator possesses the private keys of all signers except the private key of the challenge public key. In the proof, the simulator first guesses the time period w' of the forged synchronized aggregate signature and selects a random query index k of the hash function H_2. After that, if the adversary requests a signature on a message M and a time period w such that $w \neq w'$, then he can easily generate the signature by using the programmability of the random oracle model. If the adversary requests a signature for the time period $w = w'$, then he can generate the signature if the query index i is equal to the index k. Otherwise, the simulator should abort the simulation. Finally, if the adversary outputs a forged synchronized aggregate signature that is non-trivial on the time period w', then the simulator extracts the CL signature of the challenge public key from the forged aggregate signature by using the private keys of other signers.

Suppose there exists an adversary $\mathcal{A}$ that forges the above SyncAS scheme with non-negligible advantage ϵ. A simulator $\mathcal{B}$ that forges the CL signature scheme is first given: a challenge public key $PK_{CL} = (p, \mathbb{G}, \mathbb{G}_T, e, g, X, Y)$. Then $\mathcal{B}$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}$ first constructs $PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, H_1, H_2)$ and $PK^* = X$ from PK_{CL}. It chooses a random value $h' \in \mathbb{Z}_p^*$ and queries its signing oracle **PKS.Sign** to obtain $\sigma' = (A', B', C')$. Let q_{H_1} and q_{H_2} be the maximum number of H_1 and H_2 hash queries respectively. It chooses a random index k such that $1 \leq k \leq q_{H_2}$ and guesses a random time period $w' \in \mathcal{W}$ of the forged signature. Next, it initializes a key-pair list $KeyList$, hash lists $H_1\text{-}List, H_2\text{-}List$ as an empty one and gives PP and PK^* to $\mathcal{A}$.

Certification Query: $\mathcal{A}$ adaptively requests the certification of a public key by providing a public key $PK_i = X_i$ and its private key $SK_i = x_i$. $\mathcal{B}$ checks the private key and adds the key-pair (PK_i, SK_i) to $KeyList$.

Hash Query: $\mathcal{A}$ adaptively requests a hash value for H_1 and H_2 respectively. If this is a H_1 hash query on a bit $b \in \{0,1\}$ and a time period w_i, then $\mathcal{B}$ treats the query as follows:

- If $b = 0$ and $w_i \neq w'$, then it selects a random exponent $r_{0,i} \in \mathbb{Z}_p$ and sets $H_1(0||w_i) = g^{r_{0,i}}$.
- If $b = 0$ and $w_i = w'$, then it sets $H_1(0||w_i) = A'$.
- If $b = 1$ and $w_i \neq w'$, then it selects a random exponent $r_{1,i} \in \mathbb{Z}_p$ and sets $H_1(1||w_i) = g^{r_{1,i}}$.
- If $b = 1$ and $w_i = w'$, then it sets $H_1(1||w_i) = B'$.

If this is a H_2 hash query on a message M_i and a time period w_j, then $\mathcal{B}$ treats the query as follows:

- If $i \neq k$ or $w_j \neq w'$, then it selects a random value $h_{i,j} \in \mathbb{Z}_p$ and sets $H_2(M_i||w_j) = h_{i,j}$.
- If $i = k$ and $w_j = w'$, then it sets $H_2(M_i||w_j) = h'$.

Note that $\mathcal{B}$ keeps the tuple $(b, w_i, r_{b,i}, H_1(b||w_i))$ in H_1-$List$ and the tuple $(M_i, w_j, h_{i,j})$ in H_2-$List$.

Signature Query: $\mathcal{A}$ adaptively requests a signature by providing a message M_i and a time period w_j to sign under the challenge private key of PK^*. $\mathcal{B}$ proceeds the signature query as follows:

- If $w_i \neq w'$, then it responds $\sigma_{i,j} = (X^{r_{0,i}} X^{r_{1,i} h_{i,j}}, w_j)$ where $r_{0,i}, r_{1,i}$, and $h_{i,j}$ are retrieved from the H_1-$List$ and H_2-$List$.
- If $w_i = w'$ and $i = k$, then it responds $\sigma_{i,j} = (C', w_j)$.
- If $w_i = w'$ and $i \neq k$, it aborts the simulation.

Output: $\mathcal{A}$ outputs a forged aggregate signature $\sigma^*_\Sigma = (C^*, w^*)$ on messages $\mathbf{M}^* = (M_1, \ldots, M_l)$ under public keys $\mathbf{PK}^* = (PK_1, \ldots, PK_l)$ for some l. Without loss of generality, we assume that $PK_1 = PK^*$. $\mathcal{B}$ proceeds as follows:

1. It checks the validity of σ^*_Σ by calling **SyncAS.AggVerify**. Additionally, the forged signature should not be trivial: the challenge public key PK^* must be in $\mathbf{PK}^*$, and the message M_1 must not be queried by $\mathcal{A}$ to the signature query oracle.
2. If $w^* \neq w'$, then it aborts the simulation since it fails to guess the forged time period.
3. For each $2 \leq i \leq l$, it retrieves the private key $SK_i = x_i$ of PK_i from $KeyList$ and sets $h_{i,*} = H_2(M_i||w^*)$. Next, it computes

$$A = A',\ B = B',\ C = C^* \cdot \left((A')^{\sum_{i=2}^{l} x_i} (B')^{\sum_{i=2}^{l} x_i h_{i,*}} \right)^{-1}.$$

4. If $H_2(M_1||w^*) = h'$, then it also aborts the simulation.
5. It outputs $\sigma^* = (A, B, C)$ on a message $h_{1,*}$ as a non-trivial forgery of the CL signature scheme since $h_{1,*} \neq h'$ where $h_{1,*} = H_2(M_1||w^*)$.

To finish the proof, we first show that the distribution of the simulation is correct. It is obvious that the public parameters and the public key are correctly distributed. The distribution of the signatures is also correct. Next, we show that the resulting signature $\sigma^* = (A, B, C)$ of the simulator is a valid signature for the CL signature scheme on the message $h_{1,*} \neq h'$ under the public key PK^* since it satisfies the following equation:

$$\begin{aligned} e(C, g) &= e(C^* \cdot \left((A')^{\sum_{i=2}^{l} x_i} (B')^{\sum_{i=2}^{l} x_i H_2(M_i||w^*)}\right)^{-1}, g) \\ &= e((A')^{\sum_{i=1}^{l} x_i} (B')^{\sum_{i=1}^{l} x_i h_{i,*}} \cdot (A')^{-\sum_{i=2}^{l} x_i} (B')^{-\sum_{i=2}^{l} x_i h_{i,*}}, g) \\ &= e((A')^{x_1} (B')^{x_1 h_{1,*}}, g) = e(A', g^{x_1}) \cdot e(B', g^{x_1 h_{1,*}}) \\ &= e(A', X) \cdot e(B', X^{h_{1,*}}). \end{aligned}$$

We now analyze the success probability of the simulator $\mathcal{B}$. At first, $\mathcal{B}$ succeeds the simulation if he does not abort in the simulation of signature queries and he correctly guesses the time period w^* such that $w^* = w'$ in the forged aggregate signature from the adversary $\mathcal{A}$. $\mathcal{B}$ aborts the simulation of signature queries if the time period w' is given from $\mathcal{A}$ and he incorrectly guessed the index k since he cannot generate a signature. Thus $\mathcal{B}$ succeeds the simulation of signature queries at least $q_{H_2}^{-1}$ probability since the outputs of H_2 are independently random. Next, $\mathcal{B}$ can correctly guess the time period w^* of the forged aggregate signature with at least $|\mathcal{W}|^{-1}$ probability since he randomly chooses a random w'. Note that the probability $H_2(M_2||w^*) = h'$ is negligible. Therefore, the success probability of $\mathcal{B}$ is at least $|\mathcal{W}|^{-1} \cdot q_{H_2}^{-1} \cdot \mathbf{Adv}_{\mathcal{A}}^{SyncAS}$ where $\mathbf{Adv}_{\mathcal{A}}^{SyncAS}$ is the success probability of $\mathcal{A}$. This completes our proof. □

4.4 Discussions

Efficiency. The public key of our SyncAS scheme consists of just one group element since our SyncAS scheme is derived from the SeqAS scheme of the previous section, and the synchronized aggregate signature consists of one group element and one integer since anyone can compute A, B using the hash functions. The signing algorithm requires two group hash operations and two exponentiations, and the aggregate verification algorithm requires two group hash operations, three pairing operations, and l exponentiations where l is the number of signers in the aggregate signature. Our SyncAS scheme provides the shortest aggregate signature size compared to the previous SyncAS schemes [1, 14]

Combined (Multi-modal) Aggregate Signature. We can construct a combined aggregate signature scheme that supports sequential aggregation and synchronized aggregation at the same time by combining our SeqAS scheme and our SyncAS scheme since the private key and the public key of our two schemes are the same. In the combined aggregate signature scheme, the public parameters is $PP = (p, \mathbb{G}, \mathbb{G}_T, e, g, Y, H_1, H_2)$, the private key and the public key are $SK = x$ and $PK = X$ respectively. The security model of the combined aggregate signatures can be defined by combining the security models of SeqAS schemes and

SyncAS schemes. The details of this scheme are given in the full version of this paper [16].

Removing Random Oracles. If the number of messages is restricted to be polynomial, then we can use the universal one-way hash function [13, 21]. However, the SeqAS scheme using the universal one-way hash function of Canetti et al. is inefficient since it requires large number of exponentiations.

5 Conclusion

In this paper we concentrated on the notion of aggregate signatures which applications are in reducing space of signatures for large repositories (such as in the legal, financial, and infrastructure areas). We proposed a new sequential aggregate signature scheme and a new synchronized aggregate signature scheme using a newly devised "public key sharing" technique, and we proved their security under the LRSW assumption. Our two aggregate signature schemes in this paper sufficiently satisfy the efficiency properties of aggregate signatures such that the size of public keys should be short, the size of aggregate signatures should be short, and the aggregate verification should be efficient.

Acknowledgements. We thank the anonymous reviewers of FC 2013 for their valuable comments.

References

1. Ahn, J.H., Green, M., Hohenberger, S.: Synchronized aggregate signatures: new definitions, constructions and applications. In: ACM Conference on Computer and Communications Security, pp. 473–484 (2010)
2. Ateniese, G., Camenisch, J., de Medeiros, B.: Untraceable rfid tags via insubvertible encryption. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM Conference on Computer and Communications Security, pp. 92–101. ACM (2005)
3. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005), `http://eprint.iacr.org/2005/385`
4. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007)
5. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. J. Cryptology 22(1), 114–138 (2009)
6. Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. Cryptology ePrint Archive, Report 2007/438 (2010), `http://eprint.iacr.org/2007/438`
7. Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights. J. Cryptology 25(1), 57–115 (2012)
8. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

9. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
11. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007)
12. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
13. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
14. Gentry, C., Ramzan, Z.: Identity-based aggregate signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257–273. Springer, Heidelberg (2006)
15. Gerbush, M., Lewko, A., O'Neill, A., Waters, B.: Dual form signatures: An approach for proving security from static assumptions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 25–42. Springer, Heidelberg (2012)
16. Lee, K., Lee, D.H., Yung, M.: Aggregating cl-signatures revisited: Extended functionality and better efficiency. Cryptology ePrint Archive, Report 2012/562 (2012), http://eprint.iacr.org/2012/562
17. Lee, K., Lee, D.H., Yung, M.: Sequential aggregate signatures with short public keys: Design, analysis and implementation studies. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 423–442. Springer, Heidelberg (2013)
18. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
19. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
20. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems (Extended abstract). In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
21. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Johnson, D.S. (ed.) STOC, pp. 33–43. ACM (1989)
22. Schröder, D.: How to aggregate the CL signature scheme. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 298–314. Springer, Heidelberg (2011)
23. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
24. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Accumulators and U-Prove Revocation*

Tolga Acar[1], Sherman S.M. Chow[2], and Lan Nguyen[3]

[1] Intel Corporation
tolga.acar@intel.com
[2] Department of Information Engineering
Chinese University of Hong Kong
sherman@ie.cuhk.edu.hk
[3] Microsoft Research
lan.duy.nguyen@microsoft.com

Abstract. This work introduces the most efficient universal accumulator known today. For the first time, we have an accumulator which does not depend on hidden order groups, does not require any exponentiations in the target group associated with the pairing function, and only requires two pairings to verify a proof-of-knowledge of a witness.

We present implementations of our accumulator and another recent proposal utilizing Groth-Sahai proofs, with performance results. Our implementations are designed with cryptography agility in mind. We then built a library for revoking anonymous credentials using any accumulators, and integrated it with Microsoft U-Prove, which has a significant contribution to an European Union's privacy standardization effort. Our work enables U-Prove revocation without compromising untraceability.

Keywords: dynamic universal accumulator, U-Prove, revocation, blacklist, privacy enhancing technologies, efficiency, anonymity, accountability, authentication, pairing, implementation, cryptography agility.

1 Introduction

Accumulator. A cryptographic accumulator allows aggregation of a large set of elements into one constant-size *accumulator value*, and proving about whether an element has been accumulated. Via the proof system, a prover with a *witness* can convince a verifier about the truth of a *statement*, but any adversary cannot convince a verifier about a false statement. The basic proof is about *membership* statement, for proving that an element has been accumulated. An accumulator is said to be *universal* if it has another proof system, called *non-membership*, to prove that a given element is not accumulated. Moreover, if the costs of updating the accumulator or witnesses, when elements are added to or deleted from the accumulator, do not depend on the number of elements aggregated, we say it is *dynamic*. Two universal dynamic accumulators have been proposed so far. One is based on the Strong RSA assumption in hidden order groups [16], which

* This is an extended abstract. We thank the help of Benoît Libert and Toru Nakanishi.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 189–196, 2013.

is derived from a basic (non-universal) accumulator proposed [8]. The other is based on the Strong Diffie-Hellman assumption in prime-order groups with bilinear map (or pairing) [3] which is derived from a pairing-based non-universal accumulator proposed [19]. There is another pairing-based dynamic accumulator [7]; however, it is not universal.

Applications. Accumulators have been used in various applications for different purposes. Two of its major benefits include minimizing the bandwidth requirement and protecting privacy. With the proliferation of mobile devices, bandwidth requirement is becoming more crucial. On the other hand, privacy is a growing concern in various different sectors such as healthcare, military, intelligence, and mobile devices industry. The emergence of cloud computing also leads to a different balance between trust and anonymity in identity management. Some applications of accumulators include space-efficient time-stamping [4], electronic voting [10], and many privacy-preserving authentication mechanisms which include ad-hoc anonymous authentication [12], (ID-based) ring signatures [12,19,11], dynamic k-times anonymous authentication [2], and *anonymous credentials* which are *revokable* [8,7].

Anonymous Credentials. Using *anonymous credentials* a user can prove the possession of some credentials without revealing any other private information such as her identity. Applications include direct anonymous attestation [6], anonymous electronic identity token [9,17], and implementations such as U-Prove [17], Idemix [9] and in Java cards [5]. In practice, *revocation* is indispensable in credential systems, as dispute, compromise, mistake, identity change, hacking and insecurity could make any credential become invalid before its expiration, especially when they are carried around by mobile devices.

Revocation. Revoking credentials is a notorious issue in cryptography, not to say anonymous credentials. Consider public key infrastructure, there is a certificate revocation list which consists of invalid certificates, and is time-stamped and signed by a certificate authority. Checking if a certificate is revoked requires searching for the certificate's identity in the entire list. For anonymous credential, it is even trickier. In particular, the anonymity and the run-time requirements of honest users should not be affected by revocation of other credentials.

Our contribution. The major contribution of this work is our efficiency improvement of (non-)membership proofs associated with an accumulator, which brings benefits to many existing applications utilizing accumulator. Our efficiency gain comes from a new design which only requires proving equations in a base group $\mathbb{G}_1$ associated with the curves featuring the pairing function, instead of the standard approach in the literature which proves about some pairing equations. As a result, not only the number of pairing operations is greatly reduced, there are also no exponentiation or other operations in the target group $\mathbb{G}_T$ (nor in the other base group $\mathbb{G}_2$), which are often inefficient when compared with the operations in $\mathbb{G}_1$. (Exponentiation in $\mathbb{G}_T$ could be 4 or 5 times more expensive than that in $\mathbb{G}_1$, and exponentiation in $\mathbb{G}_2$ could still be 1.5 to 2 times more expensive.) In particular, our proof systems require less exponentiations and pairings compared to previous pairing-based schemes [19,3] which use

symmetric pairings. It is especially suitable for Barreto-Naehrig pairing curves (BN curves) [14], arguably one of the most efficient pairing-friendly curves, where $\mathbb{G}_1$'s operations are also very efficient.

2 U-Prove with Revocation Using Our New Accumulator

In this section, we show how to use the non-membership proof of our accumulator to blacklist U-Prove tokens. It will be embedded in the U-Prove presentation protocol as detailed in U-Prove Crypto Specification V1.1 ('Spec') [17]. Due to the lack of spaces, this part is a sketch and should be read in conjunction with the Spec [17]. The text "[**For Revocation**]" highlights the additions of revocation parts to the existing U-Prove protocols.

2.1 Entities

In the original U-Prove Spec, there are Issuer, Prover and Verifier. Issuer issues each Prover U-Prove tokens that contain Issuer's certification of the Prover's attributes. A Prover can prove some of her attributes to a Verifier in a U-Prove token presentation protocol. Besides these three entities as in Spec, we introduce *Blacklist Authority (BA)*, who could be the same as or independent from Issuer. There could be several BAs and several blacklists. Apart from proving Prover's attributes, our new token presentation protocol can also prove that the token is not in any of the blacklists, with full anonymity.

2.2 System Parameters

The generation of the system parameters follows closely as the original Spec, except that the system parameters for the accumulator will also be included.

Issuer Parameters: Define

$$IP := (\mathsf{UID}_\mathrm{P},\ \mathsf{desc}(\mathbb{G}_q),\ \mathsf{UID}_{\mathcal{H}}, (g_0, g_1, \ldots,\ g_n,\ g_t), (\mathrm{e}_1, \ldots,\ \mathrm{e}_n), S))$$

where $\mathsf{UID}_{\mathcal{H}}$ is an application-specific unique identifier for Issuer parameters and $\mathsf{UID}_{\mathcal{H}}$ is an identifier of a cryptographically secure hash algorithm.

[For Revocation] Accumulator Parameters: Define

$$\mathsf{param} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, P_{pub}, H, K, G_1)$$

where $P_{pub} := {P_2}^{\delta}$, $K := H^{\delta}$ for a random $\delta \in \mathbb{Z}_q$ and $G_1, H \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$.

1. One may include vector $\boldsymbol{t} := ({P_1}^{\delta}, {P_1}^{\delta^2}, \ldots, {P_1}^{\delta^k})$ for the users to compute their own witness (to be described). We can have a constant-size public-key (without the vector $\boldsymbol{t}$) when the BA computes the witness for each user using the auxiliary secret value δ.
2. The order of the bilinear groups where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ should all have the same prime order q, i.e., the accumulator should be instantiated with the same prime number q used in Issuer's parameter.

2.3 Issuing U-Prove Token

The protocol for Issuer to issue U-Prove tokens to Provers is the same as in the original Spec, except that there is a designated attribute x_{id} in the U-Prove token for revocation purpose.

Generating U-Prove Token: A U-Prove token has the same form as in Spec:

$$(\mathsf{UID}_{\mathrm{P}},\ h,\ TI,\ PI,\ \sigma'_z,\ \sigma'_c,\ \sigma'_r)$$

where

- The public key h, with a number of attributes $x_t, x_1, \ldots, x_n$ embedded, is $h = (g_0 {g_1}^{x_1} \ldots {g_{\mathsf{id}}}^{x_{\mathsf{id}}} \ldots {g_n}^{x_n} {g_t}^{x_t})^{\alpha} \pmod{q}$.
- The private key is $\alpha^{-1} \in \mathbb{Z}_q^*$.
- TI/PI are token/prover information field respectively.
- A valid signature from an issuer is given in the form of $(\sigma'_z,\ \sigma'_c,\ \sigma'_r)$.

2.4 Blacklist

A blacklist is published and managed by a BA. Note that BA has the auxiliary information to efficiently compute the accumulator values. For example, on day 1, three elements $x_{\mathsf{id}_1}, x_{\mathsf{id}_2}, x_{\mathsf{id}_3}$ can be accumulated in V_1; then, on day 2, x_{id_2} can be deleted, and x_{id_4} can be added, which results in an updated V_2.

[For Revocation] Maintaining Revocation List: BA decides an attribute x_{id} for revocation. We can have some options here: x_{id} could uniquely identify a token, a user or an organization. Suppose there are m revocations, BA publishes the list of $\{x_{\mathsf{id}_i}\}$, and accumulates them by

$$V := {P_1}^{\prod_{i=1}^{m} (\delta + x_{\mathsf{id}_i})}.$$

2.5 Presenting U-Prove Tokens

The difference from the basic presentation protocol of a valid U-Prove token is that, now a Prover also needs to prove the knowledge of x_{id}, which is the attribute used for revocation, is not accumulated in BA's blacklist.

Input:

1. Ordered indices of disclosed attributes: $D \subset \{1,\ \ldots,\ n\}$
2. Ordered indices of undisclosed attributes: $U = \{1,\ \ldots,\ n\} \setminus D$
3. U-Prove token: $\mathcal{T} = (\mathsf{UID}_{\mathrm{P}},\ h,\ TI,\ PI,\ \sigma'_z,\ \sigma'_c,\ \sigma'_r)$
4. Messages: $m \in \{0,1\}^*$
5. Private key: $\alpha^{-1} \in \mathbb{Z}_q$
6. Attribute values: $(A_1, \cdots, A_n) \in (\{0,1\}^*)^n$
7. **[For Revocation]** The current accumulator value: V
8. **[For Revocation]** The updated witness: (W, Q, d) where
   - $d = \prod_{i=1}^{m} (\delta + x_{\mathsf{id}_i}) \pmod{(\delta + x_{\mathsf{id}})} \in \mathbb{Z}_q$,
     which can be computed without knowing δ by simple polynomial division of $\prod(z + x_{\mathsf{id}_i})$ by $(z + x_{\mathsf{id}})$, and is non-zero since $x_{\mathsf{id}} \notin \{x_{\mathsf{id}_i}\}$.

- $W = P_1{}^{(\prod_{i=1}^{m}(\delta+x_{\mathsf{id}_i})-d)/(\delta+x_{\mathsf{id}})}$
  which is computable by using vector $\boldsymbol{t}$ in param.
- $Q = VW^{-x_{\mathsf{id}}}{P_1}^{-d}$.

Proof Generation

1. [**For Revocation**] Prepare for the commitment part of the proof that the credential is not revoked:

   Randomly pick $x, u, t_1, t_2, t_3, r_x, r_u, r_{t_1}, r_{t_2}, r_{t_3}, r_{\beta_1}, r_{\beta_2}, r_{\beta_3}, r_d, r_{d'} \in \mathbb{Z}_q$. Set:

   $$\begin{array}{llll} X := WH^{t_1}, & Y := QK^{t_1}, & C := {G_1}^{x}H^{u}, A := {G_1}^{r_x}H^{u} \\ R := {G_1}^{t_1}H^{t_2}, & S := {G_1}^{d'}H^{t_3}, & \Gamma := X^{-r_x}H^{r_{\beta_1}}K^{r_{t_1}}{P_1}^{-r_d}, \\ T_1 := {G_1}^{r_{t_1}}H^{r_{t_2}}, & T_2 := {G_1}^{r_{\beta_1}}H^{r_{\beta_2}}R^{-r_x}, & T_3 := {G_1}^{r_{d'}}H^{r_{t_3}}, T_4 := H^{r_{\beta_3}}S^{-r_d}. \end{array}$$

2. [**For Revocation**] (In the original Spec, $a := \mathcal{H}(h^{w_0}(\prod_{i\in U} g_i^{w_i}))$.
   Compute $a := \mathcal{H}(h^{w_0}(\prod_{i\in U} g_i^{w_i}), \mathcal{H}(X, Y, R, S, T_1, T_2, T_3, T_4, \Gamma, \mathsf{param}))$.
3. $c := \mathsf{GenerateChallenge}(IP, \mathcal{T}, a, m, \varnothing, D, \{x_i\}_{i\in D}) \in \mathbb{Z}_q$.
4. [**For Revocation**] Prepare for the response part for the proof that the credential is not revoked, by computing the following:

   $$\begin{array}{llll} \beta_1 := t_1 x_{\mathsf{id}}, & \beta_2 := t_2 x_{\mathsf{id}}, & \beta_3 := t_3 d, & d' := d^{-1} \\ s_u := -cu + r_u, & s_x := -cx + r_x, & s_d := -cd + r_d, & s_{d'} := -cd' + r_{d'} \\ s_{t_1} := -ct_1 + r_{t_1}, & s_{t_2} := -ct_2 + r_{t_2}, & s_{t_3} := -ct_3 + r_{t_2}, & \\ s_{\beta_1} := -c\beta_1 + r_{\beta_1}, & s_{\beta_2} := -c\beta_2 + r_{\beta_2}, & s_{\beta_3} := -c\beta_3 + r_{\beta_3}. & \end{array}$$

5. $x_t := \mathsf{ComputeXt}(IP, TI) \in \mathbb{Z}_q$.
6. For each $i \in \{1, \ldots, n\}$, $x_i := \mathsf{ComputeXi}(IP, A_i) \in \mathbb{Z}_q$.
7. Generate w_0 at random from $\mathbb{Z}_q$, set $r_0 := c\alpha^{-1} + w_0 \pmod q$.
8. For each $i \in U$, generate w_i at random from $\mathbb{Z}_q$, set $r_i := -cx_i + w_i \pmod q$.
9. Return the U-proven token proof $\left(\{A_i\}_{i\in D},\ a,\ r_0, \{r_i\}_{i\in U}\right)$.
10. [**For Revocation**] Also return the non-revoked proof:

$$(c, s_u, s_x, s_d, s_{d'}, s_{t_1}, s_{t_2}, s_{t_3}, s_{\beta_1}, s_{\beta_2}, s_{\beta_3}, C, X, Y, R, S)\,.$$

Proof Verification

1. Execute $\mathsf{VerifyTokenSignature}(IP, \mathcal{T})$ which verifies $(\sigma'_z,\ \sigma'_c,\ \sigma'_r)$.
2. $x_t := \mathsf{ComputeXt}(IP,\ TI)$.
3. For each $i \in D$, $x_i := \mathsf{ComputeXi}(IP, A_i)$.
4. Set $c := \mathsf{GenerateChallenge}(IP, \mathcal{T}, a, m, \varnothing, D, \{x_i\}_{i\in D})$.
5. [**For Revocation**] Execute the following steps for non-membership proof verification:

   $$\begin{array}{l} \tilde{T_1} := {G_1}^{s_{t_1}}H^{s_{t_2}}R^{c}, \tilde{T_2} := {G_1}^{s_{\beta_1}}H^{s_{\beta_2}}R^{-s_x}, \\ \tilde{T_3} := {G_1}^{s_{d'}}H^{s_{t_3}}S^{c}, \tilde{T_4} := {G_1}^{-c}H^{s_{\beta_3}}S^{-s_d}, \\ \tilde{A} := {G_1}^{s_x}H^{s_u}C^{c}, \quad \tilde{\Gamma} := X^{-s_x}H^{s_{\beta_1}}K^{s_{t_1}}{P_1}^{-s_d}(V^{-1}Y)^{c}. \end{array}$$

   Verify if $e\,(Y, P_2) \stackrel{?}{=} e(X, P_{pub})$, and if

$$a = \mathcal{H}((g_0 g_t^{x_t} \prod_{i \in D} g_i^{x_i})^{-c} h^{r_0} (\prod_{i \in U} g_i^{r_i}), \mathcal{H}(X, Y, R, S, T_1, T_2, T_3, T_4, \Gamma, \mathsf{param})).$$

3 Crypto-Agile Software Design

The design of our accumulator-and-applications system is illustrated in Figure 1. There is one common application program interface (API) for all accumulators, providing the interface for the operations. Two accumulators, the new one in this paper (based on Fiat-Shamir transformation [13], denoted as ACN) and the existing one (based on Groth-Sahai proof [15], denoted as AccuGS) [1], have been implemented according to the API. They are the first implementation of universal accumulators, and AccuGS provides the only solution for revoking delegatable anonymous credentials not relying on random oracles.

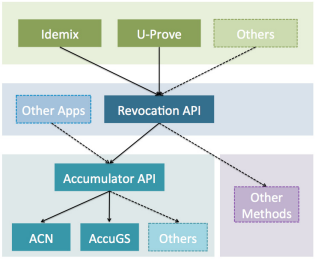


Fig. 1. Our Software Design

The API could be used to develop accumulator's applications. One such application is for revoking anonymous credentials. We further implement a revocation API. A blacklist authority could use it to create a blacklist and accumulate revoked anonymous credentials, and an user could prove that a credential is not accumulated in a blacklist. Several anonymous credential systems based on prime order could use this revocation. We have used it for U-Prove.

This design supports crypto-agility. With a common accumulator API, it is easy to add other accumulators' implementations based on different assumptions, to replace an existing implementation with a more efficient or safer one, and to switch among them with minimum code refactoring. With a single revocation API and a single accumulator API, we just need a single implementation of "Revocation using Accumulators". This reduces redundancy and allows us to have a painless changes from, say, AccuGS to ACN.

The system is developed in `C++` and built into 3 dynamic link libraries (dll) in Windows. The first, `accumulator.dll`, implements ACN and AccuGS. The second, `RAC.dll`, utilizes the first library for revoking anonymous credentials. The third, `UProveRAC.dll`, uses the second library to integrate revocation into U-Prove. `UProveRAC.dll` consists of all existing UProve API and additional API functions with revocation capability. The new UProveRAC functions allow generating revocation parameters and keys, computing and updating witnesses, and proving and verifying that U-Prove tokens are not revoked.

Table 1. Performance of ACN, ATSM, and AccuGS

(a) Operation counts: (E_1, E'_1, and E_T denote exponentiations in $\mathbb{G}_1$, $\mathbb{G}'_1$, and $\mathbb{G}_T$ resp.; $x(+y)$: x pairings, another y of them can be pre-computed)

Algorithms	ACN		ATSM		
	E_1	$e(\cdot,\cdot)$	E'_1	E_t	$e(\cdot,\cdot)$
NMPrf()	21	0	19	4	2 (+4)
NMVfy()	20	2	19	5	2 (+4)
MemPrf()	14	0	13	3	2 (+3)
MemVfy()	13	2	13	4	2 (+3)

(b) Running time ("Update witness" here is for adding 40 and removing 5 elements, i.e., a total of 45 unit updates.)

Operations	ACN	AccuGS
Update witness	77.102	78.980
NMPrf()	11.882	30.609
NMVfy()	18.714	102.978
MemPrf()	24.148	117.897
MemVfy()	80.635	587.164

4 Performance

For a fair comparison, we compare the performances of ACN and the only previous universal accumulator ATSM [3], which derived the membership proof from Nguyen's [19] and introduced a new NM proof. We only compare the proof systems, as other algorithms are very similar. Both the proofs of ACN and ATSM can be made interactive and do not rely on the random oracle heuristics, or can be converted into non-interactive version via Fiat-Shamir heuristics [13] which relies on the random oracle. ATSM only uses symmetric pairing $\mathbb{G}'_1 \times \mathbb{G}'_1 \to \mathbb{G}'_T$, whereas ACN works for both symmetric and asymmetric settings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. So ACN could use much more efficient asymmetric pairing groups, such as BN curves [14]. Indeed, as discussed, operations in $\mathbb{G}_1$ are much more efficient than corresponding operations in $\mathbb{G}'_1$ and $\mathbb{G}'_T$. ACN is especially suitable for BN curves, which features not only one of the most efficient pairing operations, but also very efficient exponentiation in $\mathbb{G}_1$. Even if we do not perform pre-computation for some pairings, ACN is still significantly more efficient than ATSM. Table 1a compares the numbers of exponentiations and pairings between these proof systems.

Table 1b shows the performance (in milliseconds) of the major functions in ACN and AccuGS [1], running on a modest machine — Intel Core2 2.4 GHz with 4 GB RAM, 64-bit Win7, using 254-bit BN curves. Again, the underlying accumulators are the same so we focus on witness operations. Accumulating 40 elements take 3.5 ms, and updating accumulator with 1 element takes 1.71 ms. Finally, it takes 36.055ms to generate a U-Prove proof and 73.817ms to verify it. Those numbers never depend on the blacklist's size.

References

1. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 423–440. Springer, Heidelberg (2011)
2. Au, M.H., Susilo, W., Mu, Y., Chow, S.S.M.: Constant-size dynamic k-times anonymous authentication. IEEE Systems Journal 17(3), 46–57 (2013)

3. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 295–308. Springer, Heidelberg (2009)
4. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
5. Bichsel, P., Camenisch, J., Groß, T., Shoup, V.: Anonymous credentials on a standard java card. In: ACM CCS 2009, pp. 600–610 (2009)
6. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: ACM CCS 2004, pp. 132–145 (2004)
7. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
8. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
9. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: ACM CCS 2002, pp. 21–30 (2002)
10. Chow, S.S.M., Liu, J.K., Wong, D.S.: Robust receipt-free election system with ballot secrecy and verifiability. In: NDSS 2008, pp. 81–94 (2008)
11. Chow, S.S.M., Susilo, W., Yuen, T.H.: Escrowed linkability of ring signatures and its applications. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 175–192. Springer, Heidelberg (2006)
12. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in *ad hoc* groups. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
13. 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)
14. Geovandro, C.C.F.P., Simplício Jr., M.A.S., Naehrig, M., Barreto, P.S.L.M.: A family of implementation-friendly BN elliptic curves. Journal of Systems and Software 84(8), 1319–1326 (2011)
15. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
16. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
17. Microsoft. U-Prove cryptographic specification V1.1 Revision 2 (2013), `http://research.microsoft.com/en-us/projects/u-prove` (last visited on May 6, 2013)
18. Nelson, D.: Crypto-agility requirements for remote authentication dial-in user service (RADIUS). RFC 6421 (Informational) (November 2011)
19. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)

Towards a Publicly-Verifiable Mix-Net Providing Everlasting Privacy

Johannes Buchmann[1], Denise Demirel[1], and Jeroen van de Graaf[2]

[1] Technische Universität Darmstadt, Germany
[2] Universidade Federal de Minas Gerais, Brazil

Abstract. All implementations of verifiable mix-nets provide computational privacy only, because the audit information published is encrypted using some public key algorithm. Consequently, at some time in the future, when the underlying cryptographic assumption is broken, privacy is violated, and each output message can be traced back to its input. We address this problem by presenting a mix-net that uses a homomorphic, unconditionally hiding commitment scheme to encrypt the audit information, implying unconditional or everlasting privacy towards the public. The correctness of our mix-net is guaranteed with overwhelming probability even if all authorities conspire, under the assumption that the commitment scheme is computationally binding until the mixing process has ended. An implication of our result is that many current applications that use mix-nets can be upgraded to unconditional privacy.

Keywords: Mix-Net, Everlasting Privacy, Universal Verifiability.

1 Introduction

1.1 Motivation

Mix-nets were introduced by David Chaum in 1981 [2] to allow anonymous communication within a network. Its basic functionality is to process a set of input messages, so that the content remains unchanged while any link between a single input and its associated output is removed. The reencryption mix-net introduced in 1993 by Park et al. [10] is based on this technique and allows that its correctness can be verified by any third party, i.e. showing that each message that comes in, goes out. Universally verifiable mix-nets are of interest for several applications where privacy plays an important role. Examples of real use are electronic auctions [6], electronic exam systems [7], and electronic voting [1,12].

In all solutions for universal verifiable mix-nets that we know of, the audit information is encrypted using some public key algorithm, which is assumed to be computationally hard. However, when the underlying cryptographic assumption is broken (perhaps decades later) all the audit information can be decrypted and privacy is violated: each output message published can be traced back to its input. With current trends in technology, like quantum computers, such a scenario is realistic. With the cost for storing information becoming less, the

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 197–204, 2013.

fact that (encrypted) information has been published on the internet makes it virtually impossible to remove this data later on. In addition, processing power increases continually following Moore's law. So all an attacker needs to do is to download the audit information published, wait until the cryptographic assumption is broken, and decrypt it. In other words, the privacy offered by current implementations of mix-nets has an (often unknown) expiration date.

1.2 High-Level Description of Our Result

In this paper we show how to use a homomorphic, unconditionally hiding commitment scheme to encode the audit information, i.e. the encoded messages and the proofs published for correct mixing. The system commits to a submitted message t with a (randomly chosen) decommitment value s. Like with homomorphic encryption, each mix can recode (or rerandomize) a commitment $u = \mathsf{Com}(t, s)$ by multiplying it with $\mathsf{Com}(0, s')$, that is, $u' = \mathsf{Com}(t, s + s')$. Further, due to the homomorphic property of the commitment scheme the correctness of the rerandomization can be shown by proving knowledge of the used permutation and rerandomization value. However, in order to open shuffled commitments, one needs to know the decommitment values. Our solution is to send this data together with message t as auxiliary information through a **private** mix-net to which the public has no access. Any rerandomization $u' = \mathsf{Com}(0, s')u$ has matching rerandomizations $\langle v', w' \rangle = \langle \mathsf{Enc}(0)v, \mathsf{Enc}(s')w \rangle$, where $v = \mathsf{Enc}(t)$ and $w = \mathsf{Enc}(s)$ and Enc is a suitable homomorphic encryption scheme. So essentially we use two tightly synchronized mix-nets run by the same mixes: one which is mixing commitments and which is fully public. And a second mix-net which uses homomorphic encryption to which the public has no access. Then, after the last mix M_n has published its data, v_n and w_n are jointly decrypted yielding $s^* = s_0 + \cdots + s_n$ and t, the opening values of $u_n = \mathsf{Com}(t, s_0 + \cdots + s_n)$.

The scheme sketched in the above paragraphs already provides everlasting privacy towards observers. But it has one drawback: the first mix, to whom the users submit their message, gets to see $\mathsf{Enc}(t)$. So when, in some future, the encryption scheme gets broken, this mix, if dishonest, could reconstruct the user's message. If this is considered a problem, an obvious solution is to split the input in two (or more) parts, submit each part to a separate mix-net, and have the parts recombined, decoded, and published by a special publication authority.

1.3 Related Work

Despite the vast literature on mix-nets, we are not aware of any publication that accomplishes everlasting privacy for mixing. The protocol that comes closest to what we propose here is the shuffling used in the Split-Ballot voting system [9]. However, this protocol is devised for a very particular situation: two voting authorities who want to shuffle ballot shares of a specific format, allowing them to jointly compute the tally. It is not at all obvious how to utilize this shuffle protocol in other voting protocols, nor how to obtain a general, unconditional mix-net for mixing messages of arbitrary format. Another way to illustrate the difference

is as follows: in Split-Ballot, the voter shares a secret with two authorities "in parallel", whereas in conventional mixing a message is processed sequentially.

In [4] similar ideas presented here are directly applied on the Helios voting system. In independent work parallel to ours, Pereira et al. [11] approach the ideas of [4] and this paper from a different angle. The commitment scheme with matching encryption is presented as a new, unified primitive, and then used to show that any voting scheme which uses homomorphic tallying can have an unconditionally (or perfectly) private audit trail. A similar statement is made for voting protocols that use mix-nets, but, unlike this paper, no details of the mixing process are presented, nor does it address the issue of a mix-net providing everlasting privacy towards both the public and the mixing authorities.

The structure of this paper is straightforward: In Section 2 we show how the standard reencryption process can be adapted to provide everlasting privacy towards observers and present formal statements of the properties. Section 3 describes a mixing process providing everlasting privacy also towards the authorities, followed by conclusion and future work.

2 Mixing with Everlasting Privacy Towards the Public

2.1 Cryptographic Primitives

The protocol uses two cryptographic primitives. First, a commitment scheme in order to provide everlasting privacy and universal verifiability towards the public. Second, a matching encryption scheme allowing the authorities to open the commitment at the end of the mixing process and at the same time providing privacy towards the authorities.

A (non-interactive) **commitment scheme** is a triple $(\mathsf{GenCom}, \mathsf{Com}, \mathsf{Unv})$ such that $\mathsf{GenCom}(1^\kappa)$ generates the public commitment key ck for security parameter κ. Note that the security parameter defines the message space $\mathcal{M}$ and the randomization space $\mathcal{R}$. We will suppose implicitly the presence of ck in the remainder, leaving it out of the notation. $c = \mathsf{Com}(m, s) \in \mathcal{C}$ takes as input a message $m \in \mathcal{M}$, a uniformly chosen decommitment value $s \in \mathcal{R}$, resulting in a commitment $c \in \mathcal{C}$. The algorithm $\mathsf{Unv}(c, m, s)$ returns m if $c = \mathsf{Com}(m, s)$ and $\perp$ if not. The commitment scheme has to provide the following properties. (1) Correctness: For any $m \in \mathcal{M}, r \in \mathcal{R} : \mathsf{Unv}(\mathsf{Com}(m, r), m, r) = m$. (2) Non-Interactive: All communication goes from the sender to the receiver. (3) Computationally Binding: Given a commitment $c = \mathsf{Com}(m, r)$, for any PPT $\mathcal{A}$ the probability to find a second opening pair (m', r') with $m \neq m'$ such that $\mathsf{Com}(m, r) = \mathsf{Com}(m', r')$ is negligible in κ. (4) Unconditionally Hiding: For any pair $m, m' \in \mathcal{M}$ the distribution of the randomized values $\mathsf{Com}(m, r)$ and $\mathsf{Com}(m', r')$ must be identical when $r, r' \in \mathcal{R}$ are chosen uniformly random. (Obviously, this property can be weakened to statistically hiding, but for ease of exposition we do not explore this.) (5) Homomorphic: For all $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$ $\mathsf{Com}(m, r) \cdot_{\mathcal{C}} \mathsf{Com}(m', r') = \mathsf{Com}(m +_{\mathcal{M}} m', r +_{\mathcal{R}} r')$. In the following, we will denote the neutral element $0_{\mathcal{M}}$ of $\mathcal{M}$ and $0_{\mathcal{R}}$ of $\mathcal{R}$ as 0. Furthermore, we will refer to the group operations $+_{\mathcal{M}}$ and $+_{\mathcal{R}}$ by $+$.

The **encryption scheme** $(\mathsf{GenEnc}, \mathsf{Enc}, \mathsf{Dec})$ used is identical to the one already used in standard reencryption mix-nets. The mixing process is based on a **homomorphic public key encryption algorithm** defined by the triple $(\mathsf{GenEnc}, \mathsf{Enc}, \mathsf{Dec})$ such that GenEnc generates two separate keys, a public key pk and a private key sk, where the private key is presumed to be shared among a set of key trustees $\mathcal{T}$ using threshold decryption. $\mathsf{Enc}(t, s) = c$ denotes the encryption of message $t \in G$ with randomness $s \in H$ and public key pk. $\mathsf{Dec}(c) = t$ denotes the decryption of ciphertext c to message $t \in G$ using private key sk. Note that the algorithm should provide semantical (CCA) security and be homomorphic in t and in s, meaning that for all $t, t' \in G$ and for all $s, s' \in H : \mathsf{Enc}(t, s) \cdot \mathsf{Enc}(t', s') = \mathsf{Enc}(t +_G t', s +_H s')$, where $+_G$ and $+_H$ are operations in group G and H. As a consequence, reencrypting a ciphertext $c = \mathsf{Enc}(t, s)$ without knowing and changing message t is possible by multiplying it with an encryption of the neutral element 0_G of group G, i.e. $\mathsf{ReEnc}(c, s') = \mathsf{Enc}(t, s) \cdot \mathsf{Enc}(0_G, s') = \mathsf{Enc}(t, s + s')$. However, because both the message $m \in \mathcal{M}$ and the auxiliary value $r \in \mathcal{R}$ have to be sent over the private channel, we need two instances of this encryption scheme. One, denoted as $E_{\mathcal{M}}$, which must be homomorphic over the message space $\mathcal{M}$. The second one, denoted as $E_{\mathcal{R}}$, whose message space is homomorphic over group $\mathcal{R}$.

An additional ingredient for reencryption mix-nets are **proofs of correct reencryption**: each mix has to provide a zero-knowledge proof that the data has been processed correctly, such that the set of output values is a valid shuffle of the set of input values. These proofs are made public, thus providing universal verifiability. In addition this mix-net uses **proofs of consistency**: each mix has to privately prove that the same permutation and random values have been used to rerandomize the published commitments and to reencrypt the corresponding private encrypted opening values. Note that both proofs have to be perfect zero-knowledge. More precisely, if a proof published by the prover is correct, even a computationally unbounded verifier cannot learn more than that.

To show the viability we now give some instantiations of a commitment scheme with matching encryption scheme satisfying the properties above. For their Split-Ballot voting system, Moran and Noar proposed the use of Paillier encryption in combination with a slightly modified Pedersen commitment Scheme [9, Appendix A]. We also believe that an alternative implementation with unconditional Jacobi symbols commitments and Rabin encryption might work, but the scheme would be very inefficient since each Jacobi symbol would implement only one bit. More interesting is the recent suggestion of Pereira et al. [11] for an efficient unconditionally hiding commitment scheme with matching homomorphic encryption scheme based on elliptic curves. If this could be used in combination with a non-interactive proof (or rather, argument) [5,8] this might lead to a very efficient mixing procedure.

2.2 Assumptions

For a standard reencryption mix-net to be secure the following assumptions are made: **(A)** *The authorities cannot break the underlying computational problem of*

the encryption scheme. **(B)** *At least one mix is honest and keeps the association between its input and output values secret.* **(C)** *Using* (k, n)*-threshold decryption at least* $(n - k + 1)$ *"key trustees" act honestly and keep their key portion secret.* **(D)** *All random challenge bits used in the mixing and verification steps comes from a trusted Random Beacon and are unpredictable.*

Apart from that, we make the following additional assumptions: **(E)** *The authorities cannot break the computational binding property of the commitment scheme for the parameters chosen before the messages have been published and certified.* **(F)** *There exists a private channel between the user and the first mix* M_1 *of the mix-net, between each mix and its successor in the mix-net, and between the last mix and the key trustees* $\mathcal{T}$*.* **(H)** *After the protocol has been certified all authorities destroy all information private to them.*

2.3 Adapted Mixing Process

For legibility, we will interpret a batch of messages each from a different user by a vector, denoted by a capital letter. Operations on the entries of vectors carry over to the vectors. For instance, the Perm operation permutes the entries of a vector: $T' = \mathsf{Perm}_\pi(T)$ means that $t'(i) = t(\pi(i))$.

(I) Submission. To submit a message $t \in \mathcal{M}$ encoded with randomness s, the user calculates the triple $(u, v, w) = (\mathsf{Com}(t, s), \mathsf{Enc}_{\mathcal{M}}(t), \mathsf{Enc}_{\mathcal{R}}(s))$ and provides a proof of knowledge (POK) that the t and s used in all three components are the same. The user uses a private channel to send the triple (u, v, w) to the system together with the POK for consistency. The input batch of the first mix consists of all triples received ordered in some canonical way and is denoted as (U_0, V_0, W_0). The public part of the input batch, U_0, is published.

(II) Mixing. We now describe the shuffling procedure for K inputs and a mix-net consisting of n mixes $M_1, M_2 \ldots, M_n$. The input batch of M_j is defined as the output batch of the preceding mix M_{j-1}, except that the first mix receives its inputs directly from the system. In addition, the output batch of the last mix will be sent to the key trustees $\mathcal{T}$. In other respects, the shuffling procedure for each mix is identical. (1) Let the input batch of the mix M_j be $(U_{j-1}, V_{j-1}, W_{j-1})$. Then M_j rerandomizes the commitment vector: $U'_j = U_{j-1} \cdot \mathsf{Com}(0, S_j)$, it reencrypts the vector of encryptions of the contents: $V'_j = V_{j-1} \cdot \mathsf{Enc}_{\mathcal{M}}(0)$ and, through homomorphic encryption, updates the decommit value: $W'_j = W_{j-1} \cdot \mathsf{Enc}_{\mathcal{R}}(S_j)$. (2) To obtain the output batch, M_j chooses a random permutation π_j and sets $(U_j, V_j, W_j) = \mathsf{Perm}_{\pi_j}(U'_j, V'_j, W'_j)$. (3) The commitments U_j are published, whereas the corresponding encryptions V_j and W_j are sent to M_{j+1} through a private channel. (4) M_j proves, in a publicly verifiable way, that U_j is a recoding and permutation of U_{j-1}. (5) M_j provides a POK to M_{j+1} that the output batch is a consistent rerandomization and permutation of the entire input batch, i.e. it shows it knows permutation π_j and the vector of random values S_j. Note that $\mathcal{T}$ verifies the output of the last mix.

(III) Decoding and publication. The key trustees $\mathcal{T}$ compute and publish $T^* = \mathsf{Dec}(V_n)$ and $S^* = \mathsf{Dec}(W_n)$. Observe that due to the homomorphic properties and to the fact that the same permutations π_j have been used in public

and the private network, we have that T^* and S^* are the values to open U_n, that is, $U_n = \mathsf{Com}(T^*, S^*)$.

(IV) Certification. Auditors verify whether $U_n = \mathsf{Com}(T^*, S^*)$ and certify the output if this condition and all public proofs of knowledge hold.

2.4 Properties

For proving **correctness**, it is necessary to show that the mixes did not change any of the messages. In other words, $T_0 \equiv T_n$, where $\equiv$ stands for the existence of a permutation that maps T_0 to T_n. In our case correctness does not follow straightaway from existing proofs of mixing schemes since these are all based on encryption implying that the message t is unambiguously defined. Since we want the correctness to be universally verifiable while preserving unconditional privacy, only public information can be used. However, as long as $\mathcal{T}$ can open the commitments that are published by the last mix, i.e. publish the values T^* and S^* such that $U_n = \mathsf{Com}(T^*, S^*)$, then $T_0 \equiv T_n$. In other words, we argue that for correctness it is completely irrelevant *how* $\mathcal{T}$ obtained T^* and S^*. This is a consequence of the fact that the commitment scheme used is unconditionally hiding and computationally binding. This can be proven by contradiction, showing that if after running the protocol $T_0 \not\equiv T_n$, then there exists an efficient algorithm violating the binding property by computing a commitment u that can be opened to two distinct values: $u = \mathsf{Com}(\tau_1, \sigma_1) = \mathsf{Com}(\tau_2, \sigma_2)$. Note that, because of the audit information published during all the mixing steps, any observer can perform the checks. This, together with the fact that the randomness of the challenge bit is guaranteed means that the protocol is **universally verifiable**. **Individual verifiability** follows, simply because of the fact that the users can check that their input u_0 is published.

If at least one mix is honest and keeps the used permutation secret, then **privacy** follows from the fact that the output batch T_{out} is an unknown permutation of the input batch, T_{in}. Even a computationally unbounded attacker cannot obtain any additional information, since the commitments used to encode the messages are unconditional and all used proofs provide perfect zero-knowledge.

Regarding **robustness**, it is clear from the construction that if everybody is honest, then the process must succeed, always. If cheating is detected the malicious mix or the whole mix-net can simply be replaced. To avoid that a mix can lie about its private input towards the verifier, we ask the user and each mix to sign its output. In addition the users have to send a POK for consistency to ensure that they cannot submit inconsistent inputs without being detected.

3 Everlasting Privacy Towards the Authorities

Using the protocol presented in Section 2, the users submit the triple (u, v, w), where v is an encryption of the message t. Though they use a private channel, this protects them from outsiders, but not from the first mix, who will be able to recover t once the computational assumption is broken. If this is unacceptable, a

possible solution is to have several mix-nets in parallel, have the users split their message, and submit each part to a different mix-net. As long as none of the first mixes nor any of their verifiers share their information with one mix of each mix-net, the privacy of the submitted message is guaranteed unconditionally.

A problem that arises here is how to match the various shares of one specific user that come out of the various mix-nets, needed to recombine the complete message. We solve this by supposing that a user submits each share labeled with the same, randomly chosen identification number r_i, which should be chosen large enough to avoid collision (say 128 bits). Since these r_is cannot be public (to prevent any of the first mixes or its verifier from tracing back a message), the message t and the random identity r_i are encoded in separate commitments which are published and the decommitment values are sent together with the opening values of the corresponding messages towards the private mix-nets. After all data have been made anonymous by the mix-nets, the r_is are decrypted by $\mathcal{T}$ and a new authority $\mathcal{Z}$ who keeps the obtained IDs private. The same r_i should appear in each of the output batches of the various mixes and $\mathcal{Z}$ should therefore be able to match shares coming from the same user, reconstruct the message, and publish it. In addition, $\mathcal{Z}$ can use the published commitments to convince the public that the correct tuples were matched.

Correctness, Individual and Universal Verifiability can be shown similar to Section 2.4. The only difference is that $\mathcal{Z}$ has to prove that messages coming from the same user have been matched correctly. This is accomplished by a POK showing that when $\mathcal{Z}$ matches, shares coming from different mixes have the same r_i. With respect to privacy, we claim that under the additional assumption that the publication authority $\mathcal{Z}$ does not share its information with any of the first mixes or its verifier, nor do one of the first mixes or its verifier collaborate with one mix of each mix-net, privacy is also unconditional towards the authorities. But if they do share information then, as long as a sufficient number of "key trustees" are honest, they still need to break the encryption algorithm. For robustness, if all parties are honest then the protocol terminates successfully, unless two users chose the same r_i, which is an extremely unlikely event. Observe that when using secret sharing, a user can submit inconsistent IDs. There seems no obvious way to verify their consistency while maintaining unconditional privacy towards the first i mixes, $M_{i,1}$. However, if users err, it just means their message cannot be decoded by $\mathcal{Z}$; it does not affect messages sent by other users. For applications where this can be caused by malicious software, a verification step during the submission phase could be implemented which prevents both, generating invalid messages and inconsistent IDs.

4 Conclusion and Future work

We presented a novel protocol for a publicly-verifiable mix-net which offers everlasting privacy towards observers, meaning that the information made public does not help an adversary with unlimited computational resources. Further, we showed how the input can be split, each share mixed by a separate mix-net and

recombined providing everlasting privacy towards both, the observers and the authorities. We believe this is an important step forward since, as Chaum himself already argued in 1984 [3], individuals cannot be expect to understand the difference between computational and unconditional security, and they should not have to worry about it. In particular, computational security is simply not enough in some applications like elections, invalidating many proposed schemes. We plan to introduce this mix-net to mixing based voting systems like Prêt à Voter [12] or Split-Ballot [9]. Further, all known instantiations for a homomorphic commitment and encryption scheme are based on computational problems that are not quantum resistant. We plan to work on this matter in the future.

Acknowledgement. This work is a direct result of the third author's visit to Dagstuhl and Darmstadt in July 2011, and he would like to express his gratitude to the Dagstuhl Institute, to SnT at the University of Luxemburg, to CASED at the University of Darmstadt, and to his hosts, Peter Ryan and Johannes Buchmann, for making this visit possible. His research was partially supported by a FAPEMIG grant, number APQ-02719-10. Further, this paper has been developed within the project 'VerKonWa', funded by the DFG. Finally, we would like to thank Tal Moran, Olivier Pereira and Jeremy Clark for interesting discussions.

References

1. Norwegian evote project, `http://www.regjeringen.no/en/dep/`
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981)
3. Chaum, D.: A new paradigm for individuals in the information age. In: IEEE Symposium on Security and Privacy, pp. 99–106 (1984)
4. Demirel, D., van de Graaf, J., Samarone dos Santos Araújo, R.: Improving helios with everlasting privacy towards the public. In: Proceedings of EVT/WOTE (2012)
5. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
6. Howlader, J., Ghosh, A., Pal, T.D.: Secure Receipt-Free Sealed-Bid Electronic Auction, p. 228 (2009)
7. Huszti, A., Pethö, A.: A secure electronic exam system. Publicationes Mathematicae Debrecen 77(3-4), 299–312 (2010)
8. Lipmaa, H., Zhang, B.: A more efficient computationally sound non-interactive zero-knowledge shuffle argument. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 477–502. Springer, Heidelberg (2012)
9. Moran, T., Naor, M.: Split-ballot voting: Everlasting privacy with distributed trust. ACM Trans. Inf. Syst. Secur. 13(2) (2010)
10. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
11. Pereira, O., Cuvelier, E., Peters, T.: Election verifiability or vote privacy: Do we need to choose? In: SecVote 2012 (2012), `http://secvote.uni.lu/`
12. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à voter: a voter-verifiable voting system. IEEE TransIFS 4(4), 662–673 (2009)

P4R: Privacy-Preserving Pre-Payments with Refunds for Transportation Systems

Andy Rupp[1], Gesine Hinterwälder[2], Foteini Baldimtsi[3], and Christof Paar[4]

[1] Karlsruhe Institute of Technology, Germany
andy.rupp@rub.de
[2] University of Massachusetts Amherst, USA
hinterwalder@ecs.umass.edu
[3] Brown University, USA
foteini@cs.brown.edu
[4] Ruhr-University Bochum, Germany
christof.paar@rub.de

Abstract. We propose a new lightweight payment scheme for transit systems called P4R: Privacy-Preserving Pre-Payments with Refunds. In P4R a user deposits money to obtain a bundle of credentials, where each credential allows to make an arbitrary ride. The actual fare of a trip is determined on-the-fly when exiting. Overpayments are refunded where all trip refunds of a user are aggregated in a single token thereby saving memory and increasing privacy. We build on Brands' e-cash scheme to realize the pre-payment system and a new variant of blind Boneh-Lynn-Shacham signatures to implement the refund capabilities. Our construction is secure against malicious users and guarantees user privacy. We also provide an efficient implementation that shows the suitability of our scheme as future transit payment system.

Keywords: E-cash, refunds, lightweight payments, transit systems.

1 Introduction

Deploying electronic payment systems in transportation as opposed to sticking with traditional systems (like cash or ticket systems) offers important benefits like significantly reduced revenue collection costs, enhanced customer satisfaction as well as improved services such as dynamic pricing. Hence, electronic payment systems for transportation (EPST) already are and will become an even more important component of the critical infrastructure "transportation".

Currently deployed systems like the MBTA "Charlie Card" [14] or the E-ZPass [9] show the potential of electronic payment systems as a reasonable, fair, and efficient method for revenue collection. However, at the same time they are examples demonstrating the serious shortcomings of today's EPST since they lack sufficient mechanisms protecting their security and especially the privacy of their users: One problem that EPST seem to share with many other commercial systems implementing security functions is the deployment of cryptographically weak proprietary primitives as, e.g., demonstrated for the Charlie Card or Oyster Card [13]. Besides security issues, frequently concerns about the location

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 205–212, 2013.

privacy of EPST users are raised, i.e., the un-/traceability of users within the transportation system. For instance, E-ZPass and Fast Lane toll plaza records have been used by lawyers to prove that their client's cheating spouses were not where they pretended to be at a certain date and time [11] which shows that these systems do not respect locational privacy at all. However, in order to enable a large-scale deployment and broad acceptance of EPST, adequate security and privacy mechanisms are essential.

While currently deployed EPST suffer from serious privacy and security flaws there is a wealth of cryptographic payment schemes (Section 1.1) offering strong security and privacy properties. However, the unique requirements of the transportation domain, especially engineering constraints and functional requirements, prevent the use of well-established crypto like e-cash schemes.

In this paper we restrict to the consideration of a *transit* payment scenario such as payment systems for subways. Here payment devices can be fairly low-cost platforms such as RFID transponders, contactless or hybrid smart cards which are provided by the Transportation Authority (TA). Given such a device, a user can charge it at a vending machine to pay for rides in the subway system. The entry and exit points are typically physically secured by turnstiles that are equipped with readers responsible for calculating fares and conducting payment transactions with user devices. To avoid congestion in front of turnstiles, transactions have to be fast: A payment transaction should be executable within a few hundred milliseconds. Transactions at the vending machine are less time-critical but should also not take longer than several seconds.

The resource constraints of the user devices together with the realtime requirements are one of the main obstacles preventing the application of e-cash schemes. User devices are typically equipped with only a few tens of kilobytes of memory and an 8 or 16-bit microcontroller running at not more than 16 MHz. This situation greatly limits the performance achievable for a crypto primitive. Currently, on widely-used microcontrollers, operated at 16 MHz, a modular exponentiation in 1024-bit RSA with full-length exponents requires about 5 s while a point multiplication on a 160-bit elliptic curve group takes around 400 ms [10].

So clearly, it is prohibitive to do much more than a single full public-key operation on such a CPU during a payment transaction. However, almost all e-cash schemes make excessive use of exponentiations or ECC point multiplications in the spending protocol. Fortunately, by employing an ECC coprocessor as accelerator, the runtime of a point multiplication can be improved by roughly one order of magnitude, (e.g., a factor 12 for the coprocessor in [16]). Yet, due to power constraints we may only assume the usage of such a coprocessor when the payment device is in contact mode, e.g., when interacting with the vending machine, which fits our transit scenario.

1.1 Related Work

E-Cash. An e-cash scheme typically consists of a bank, users, merchants and the following protocols: (1) a withdrawal protocol where a user obtains e-coins from the bank; (2) a spending protocol where the user sends coins to a merchant; (3)

a deposit protocol where a merchant deposits coins obtained from a user to his bank account; and (4) other protocols for identifying malicious behavior.

In his seminal paper [7] Chaum introduced anonymous electronic cash that allows anonymous and unlinkable payments, while at the same time it ensures unforgeability of e-coins. Since then, e-cash protocols have been extensively studied. To name only a few important results: Brands [5] proposed one of the first and most famous offline anonymous e-cash schemes with the most efficient protocol when it comes to spending an e-coin. However, Brands' coins occupy a fairly large amount of memory and the withdrawal protocol is relatively expensive. To solve the memory issue, Camenisch et al. [6] proposed so-called compact e-cash but at the cost of a far less efficient spending protocol.

Payment Schemes Tailored to Transportation. Recently, EPST has started to attract the attention of the crypto community. Heydt-Benjamin et al. [12] were the first to propose an informal cryptographic framework for transit payment systems. Sadeghi et al. [19] present an RFID-based e-ticket scheme which does not expect the user's payment device to carry out too much expensive computations, but the existence of external trusted devices is assumed for the costly operations and their system only protects a user's privacy with respect to outsiders and not the transportation authority. Blass et al. [4] proposed another offline "privacy-preserving" payment system for transit applications that solely relies on a 128-bit hash function and lots of precomputed data on the backend's side. Again, a user's privacy in their system is not protected from the TA.

Popa et al. [17], Balasch et al. [1], Meiklejohn et al. [15] and Day et al. [8] proposed privacy-preserving payment systems for electronic toll collection. In their schemes, user devices are battery-powered on-board units that collect GPS location data and report this data or fare information (computed thereof) to the toll collection provider. However, these were developed for a scenario where users subscribe for a service and pay by the end of a billing period. In the transit scenario, we cannot assume that each user has access to a trusted PC to settle accounts: an untrusted vending machine is more realistic.

1.2 Our Approach and Contribution

Due to their strong security and privacy guarantees, it would be highly desirable to build a transit payment system based on e-cash. A good candidate for this purpose is Brands' scheme because of its exceptionally efficient spending protocol.[1] On the downside, Brands' coins are large and their withdrawal is expensive. Hence, it would be beneficial to limit the amount of coins that a user has to spend to pay his fare, having to spend only a single coin per trip would be ideal. However, this conflicts with the necessity of allowing flexible and dynamic prices, i.e., fares should not be flat but arbitrary monetary amounts: Setting the denomination of a coin to be 1 cent certainly allows for flexible pricing but users would need plenty of them to pay for a trip. Setting the face value to $2

[1] Recent results show that Brands protocol cannot be proven secure using the currently known techniques [3], however it has not been shown that the scheme is insecure.

reduces the number of required coins per trip but severely restricts the system of fares. To do a tradeoff and have coins for different monetary values, one would need to deal with overpayments and change in a privacy-preserving way. This is especially difficult in EPST where bank and merchants are the same entity.

Our proposal of the transit payment system P4R addresses the issues discussed above. The idea is to "let a single coin be worth exactly the (variable) cost of an arbitrary trip in the system". More precisely, our payment system is not a typical e-cash scheme but based on the concept of pre-payments with refunds: a user has to make a deposit to get a coin worth an arbitrary ride and gets a refund if the actual fare is less than his deposit. We build our payment system using Brands' scheme where we minimize the number of coins a user needs to pay for his rides. To be precise, our approach is not limited to Brands' but also works for other schemes that can be modified in a way that coins can be shown twice without revealing the ID of a user (e.g., [2]). The refund system is realized using a new variant of blind BLS signatures and allows to aggregate refunds.

As for security, we can show that it is infeasible for malicious users to abuse the system. This includes users who try to dodge the fare or redeem higher refunds than issued. Such users will be identified by double-spending checks. Vice versa, we can show that the transportation authority cannot distinguish between trips of honest users with the same aggregated refund amounts and it cannot link rides of the same user thus providing user privacy.

Furthermore, we implemented P4R for 160-bit elliptic curves on the Moo computational RFID tag [20] which houses a 16-bit MSP430 microcontroller and may be seen as an approximation of a future payment token. The results show that the scheme is fairly efficient even on this, not for our purposes optimized, device. Assuming a clock rate of 16 MHz for a fielded version of the device, the computations required for spending can be executed in 2 ms, getting a refund takes 340 ms, and redeeming the refund token 350 ms. Withdrawal (5.29 s) is also not far from meeting real-world requirements and could strongly be improved by making use of dedicated hardware (e.g., an ECC coprocessor in contact mode).

Due to the page limit we refer to the full version [18] for a formal description of P4R, proofs of security and privacy, a detailed discussion of the implementation, as well as interesting tradeoffs between security, privacy, and efficiency.

2 A Privacy-Preserving Transit Payment System

High-Level Description. P4R is composed of three subsystems: Trip Authorization Token (TAT), Refund Calculation Token (RCT), and Refund Token (RT) system. The TAT system is offline. Here vending machines play the role of the "bank" issuing TATs and (offline) readers at the entry turnstiles play the role of a "merchant" where tokens can be spent. The RT system is online. Here roles are reversed compared to the TAT system, i.e., readers at the exit turnstiles issue refunds and the (online) vending machines redeem these tokens.

A TAT (aka ticket) is a credential that authorizes a user to make exactly one trip. A user initially makes a deposit to obtain a number of TATs where

the cost of a TAT equals the price of the most expensive trip in the system. Of course, to reduce this deposit it is also possible to have different types of TATs for different zones of the transportation system. The withdrawal of a TAT is done in a blind fashion such that a later use cannot be linked. The ID of a user is encoded in each TAT to prevent a repeated use for entering the system (double-spending detection). At the beginning of a ride a user presents an unused TAT to the reader at the entry turnstile. If it is valid and the user can show (using a zero-knowledge proof) that he knows the ID encoded in the TAT, access is granted. When leaving the system the actual fare is determined at the exit turnstile. This is done as follows: on entering, the user also received an RCT (aka stamped ticket), which contains a MAC on the TAT, the date and time, as well as on the ID of the reader. When he leaves he presents this RCT to the exit turnstile which calculates the trip cost based on this information. He also provides a blinded version of his RT (blank RT tokens are available from the vending machines) to which the reader adds the difference between the deposit for the TAT and the actual fare. To prevent re-using an RCT and thus claiming the same refund several times, the idea is to bind an RCT to the TAT which has been used on entering, and force a user to again prove the knowledge of the ID encoded into this TAT when he leaves. An RT (aka piggy bank) is used to add up several refunds instead of having a separate RT per refund. When a user decides to cash the collected refund he presents his RT to the vending machine which redeems the RT if it is not already marked as cashed in the central DB.

Some Technical Details. The TAT and RCT subsystems are loosely coupled and realized based on an extension of Brands' protocol that allows to show a TAT twice without revealing the user's identity. To setup the TAT system, the TA chooses a cyclic group $\mathbb{G}$ of prime order $q = \Theta(2^{k_1})$, group generators $g, g_1, g_2 \in \mathbb{G}$, a random number $x \in \mathbb{Z}_q^*$, and a collision-resistant hash function $H : \mathbb{G}^5 \to \mathbb{Z}_q$. The public key is $pk^{\mathsf{TAT}} = (\mathbb{G}, g, g_1, g_2, g^x, g_1^x, g_2^x, H)$ and the secret key is $sk^{\mathsf{TAT}} = x$. The user's secret ID is some number $id_{\mathcal{U}} \in \mathbb{Z}_q$. This corresponds to Brands' setting. A TAT is a tuple $\mathsf{TAT}_i = (A_i, B_i, C_i, \mathsf{sig}_x(A_i, B_i, C_i))$, where $A_i = g_1^{id_{\mathcal{U}} s_i} g_2^{s_i}$, $B_i = g_1^{x_i} g_2^{y_i}$, $C_i = g_1^{x_i'} g_2^{y_i'}$, and $\mathsf{sig}_x(A_i, B_i, C_i)$ is a Chaum-Pedersen signature. A_i encodes the user's ID and a random serial number. B_i and C_i (where C_i does not exist in Brands' original system) are commitments to random values which are later used as blinding factors for proving the ownership of a TAT (i.e., knowledge of $id_{\mathcal{U}}$ and s_i) when entering and leaving the system. If a user tries to show a TAT twice to enter or leave and is thus forced to re-use x_i, y_i or x_i', y_i', respectively, $id_{\mathcal{U}}$ can easily be revealed. Note that since a TAT is withdrawn in a blind fashion, only the user knows the values comprising TAT_i.

To setup the RCT system, a random k_2-bit key K for a MAC scheme is chosen. An RCT has the form $\mathsf{RCT}_i = (\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}}, \mathsf{MAC}_K(\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}}))$, where ts is the current timestamp and $id_{\mathcal{R}}$ is the ID of the reader at the entry.

The RT system is based on a new variant of blind BLS signatures. For setting up this system, the TA chooses cyclic groups G, G_T of prime order $p = \Theta(2^{k_3})$ with an efficient, non-degenerated pairing $e : G \times G \to G_T$, a generator $h \in G$, and an exponent $d \in \mathbb{Z}_p$. Let W be the set of all possible single-trip refunds.

$\mathcal{U}$		$\mathcal{V}$ (online)
(1) User buys TATs and receives fresh RT at vending machine		
	$\xleftarrow{\text{PoK of } id_{\mathcal{U}}}$	
	$\xleftrightarrow{\text{blind sigs on } (A_i, B_i, C_i)}$	
$\mathsf{TAT}_i = (A_i, B_i, C_i, \mathsf{sig}_x(A_i, B_i, C_i))$ where $A_i = g_1^{id_{\mathcal{U}} s_i} g_2^{s_i}$, $B_i = g_1^{x_i} g_2^{y_i}$, $C_i = g_1^{x'_i} g_2^{y'_i}$		
$\mathsf{RT} = \mathsf{SN}_{\mathsf{RT}}, R = 1, v = 0$	$\xleftarrow{\mathsf{SN}_{\mathsf{RT}}}$	$\mathsf{SN}_{\mathsf{RT}} \xleftarrow{\$} G$, add $\mathsf{SN}_{\mathsf{RT}}$ to DB
(4) User redeems RT at vending machine		
$r \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{RT}' = \mathsf{RT}^r$		
$R = Rr \bmod p$	$\xrightarrow{\mathsf{SN}_{\mathsf{RT}}, \mathsf{RT}', v, R}$	check validity of $\mathsf{SN}_{\mathsf{RT}}$
		$v \overset{?}{<} p - 1$
		$e(\mathsf{SN}_{\mathsf{RT}}^R, h^{d^v}) \overset{?}{=} e(\mathsf{RT}', h)$
		mark $\mathsf{SN}_{\mathsf{RT}}$ as redeemed in DB

$\mathcal{U}$		$\mathcal{R}$ (offline)
(2) User shows TAT and receives RCT when entering system		
	$\xrightarrow{\mathsf{TAT}_i}$	check validity of TAT_i
$z_1 = x_i + c(id_{\mathcal{U}} s_i) \bmod q$	$\xleftarrow{c}$	$c \xleftarrow{\$} \mathbb{Z}_q$
$z_2 = y_i + cs_i \bmod q$	$\xrightarrow{z_1, z_2}$	$g_1^{z_1} g_2^{z_2} \overset{?}{=} A_i^c B_i$
		add $(\mathsf{TAT}_i, z_1, z_2, c)$ to DB
	$\xleftarrow{\mathsf{RCT}_i}$	$\mathsf{RCT}_i = (\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}}, \mathsf{MAC}_K(\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}}))$
(3) User shows RCT and collects refund on his RT when leaving system		
	$\xrightarrow{\mathsf{RCT}_i}$	check validity of $\mathsf{RCT}_i, \mathsf{TAT}_i$
$z'_1 = x'_i + c'(id_{\mathcal{U}} s_i) \bmod q$	$\xleftarrow{c'}$	$c' \xleftarrow{\$} \mathbb{Z}_q$
$z'_2 = y'_i + c' s_i \bmod q$	$\xrightarrow{z'_1, z'_2}$	$g_1^{z'_1} g_2^{z'_2} \overset{?}{=} A_i^c C_i$
		add $(\mathsf{TAT}_i, z'_1, z'_2, c')$ to DB
$r \xleftarrow{\$} \mathbb{Z}_p^*$	$\xleftarrow{w}$	determine refund $w \in \mathbb{Z}_{p-1}$
$\mathsf{RT}' = \mathsf{RT}^r$	$\xrightarrow{\mathsf{RT}'}$	
$v = v + w$, $R = Rr \bmod p$, $\mathsf{RT} = \mathsf{RT}''$	$\xleftarrow{\mathsf{RT}''}$	$\mathsf{RT}'' = \mathsf{RT}'^{d^w}$

Fig. 1. Main P4R protocols executed by users $\mathcal{U}$, vending machines $\mathcal{V}$, and readers $\mathcal{R}$

Then, the public key is $pk^{\mathsf{RT}} = (G, G_T, e, h, (h^{d^w})_{w \in W})$ and the secret key is $sk^{\mathsf{RT}} = d$. Note that assuming users do not verify RTs (which is similar to not counting change for small amounts) pk^{RT} does not need to be stored on user devices. An RT holding the refund value $v \in \mathbb{Z}_{p-1}$ corresponds to the multi-signature $\mathsf{RT} = \mathsf{SN}_{\mathsf{RT}}^{d^v}$, where $\mathsf{SN}_{\mathsf{RT}}$ is a random serial number chosen by the TA. Clearly, a refund $\mathsf{RT} = \mathsf{SN}_{\mathsf{RT}}^{d^v}$ can be increased to $v + w$ by raising it to d^w. To blind an RT before collecting a refund RT is raised to some random $r \in \mathbb{Z}_p^*$. There is actually no need to remove the old blinding r before blinding RT again using r' to collect another refund, so blinding factors can be aggregated as $R = rr'$.

More details on P4R are given in Figure 1 showing the main interactions between users $\mathcal{U}$, vending machines $\mathcal{V}$, and readers $\mathcal{R}$.

3 Security and Privacy

TA Security. P4R guarantees that the TA does not lose any money: users are not able to receive reimbursements which exceed the overall deposit for TATs minus the overall fare of their trips without being identified by the TA. In order for a user to cheat the TAT subsystem he would need to either forge a TAT, reuse it or use a foreign (eavesdropped) one. Fortunately, all these possibilities are ruled out by the (assumed) security of Brands' scheme: unforgeability, double-spending security (restrictiveness), and soundness. A user could also try to cheat the RCT subsystem by creating RCTs himself, use the same RCT twice or use a "foreign" RCT not corresponding to his current trip. Security against those attacks can be reduced to MAC unforgeability and Brands' restrictiveness and soundness under the assumption that we exclude physical attacks (i.e., users jumping over turnstiles) and users adhere to the protocol schedule. Finally, a user cannot cheat the RT system: RTs cannot be redeemed twice due to online checks. Also, a user cannot forge RTs or claim higher values on his RT since this would require breaking the Σ-Incremental DH-Assumption [18].

User Security. Individual users are protected in the sense that a passive adversary is not able to steal tickets or refunds from a user. In particular, an adversary neither can use "foreign" TATs or RCTs since he would have to prove knowledge of the encoded $id_{\mathcal{U}}$ and s_i nor he can redeem an eavesdropped RT since he would need to know the aggregated blinding factor R. In the full version [18] we also consider security against more powerful adversaries.

User Privacy. A transportation system should provide *location privacy* for its users, i.e., it should not be possible to trace the movements of individual users within the system. Unfortunately, in P4R redeeming refunds leaks some information on the sequence of trips a user did. Luckily, we can show that this sequence is still hidden within the set of all possible trip sequences leading to the same total refund amount and argue that theoretically this set should be pretty large (equals the number of integer partitions of the refund value). However, one should also be aware of the limits of this approach: In practice, we cannot guarantee that during a certain period of time many such sequences actually appear in the records of the TA. For instance, it could happen that since the issue date of a RT nobody but the owner of this token did trips resulting in a particular refund amount (though in theory many sequences lead to this amount). Hence, the exact level of location privacy provided by P4R depends on the characteristics of the transportation system and user behavior. Nevertheless, we believe that for real-world transportation systems these limits are no real issues.

Acknowledgments. We would like to thank Marc Fischlin for his valuable input to the security proofs and the anonymous reviewers for helpful comments.

References

1. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: PrETP: Privacy-preserving electronic toll pricing. In: USENIX Security Symposium, pp. 63–78. USENIX Association (2010)
2. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. IACR Cryptology ePrint Archive 2012, 298 (2012)
3. Baldimtsi, F., Lysyanskaya, A.: On the security of one-witness blind signature schemes. IACR Cryptology ePrint Archive 2012, 197 (2012)
4. Blass, E.O., Kurmus, A., Molva, R., Strufe, T.: PSP: private and secure payment with rfid. In: Al-Shaer, E., Paraboschi, S. (eds.) WPES, pp. 51–60. ACM (2009)
5. Brands, S.: An efficient off-line electronic cash system based on the representation problem. Tech. Rep. CS-R9323, CWI (1993)
6. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
7. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO, pp. 199–203. Plenum Press, New York (1982)
8. Day, J., Huang, Y., Knapp, E., Goldberg, I.: SPEcTRe: spot-checked private ecash tolling at roadside. In: Chen, Y., Vaidya, J. (eds.) WPES, pp. 61–68. ACM (2011)
9. E-ZPass Interagency Group: E-ZPass, `http://www.ezpass.com/`
10. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 119–132. Springer, Heidelberg (2004)
11. Hager, C.: Divorce lawyers using fast lane to track cheaters, `http://msl1.mit.edu/furdlog/docs/2007-08-10_wbz_fastlane_tracking.pdf`
12. Heydt-Benjamin, T.S., Chae, H.-J., Defend, B., Fu, K.: Privacy for public transportation. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 1–19. Springer, Heidelberg (2006)
13. de Koning Gans, G., Hoepman, J.-H., Garcia, F.D.: A practical attack on the MIFARE Classic. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 267–282. Springer, Heidelberg (2008)
14. Massachusetts Bay Transportation Authority: CharlieCards & Tickets, `http://www.mbta.com/fares_and_passes/charlie/`
15. Meiklejohn, S., Mowery, K., Checkoway, S., Shacham, H.: The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion. In: USENIX Security Symposium. USENIX Association (2011)
16. Park, J., Hwang, J.T., Kim, Y.C.: FPGA and ASIC implementation of ECC processor for security on medical embedded system. In: ICITA (2), pp. 547–551. IEEE Computer Society (2005)
17. Popa, R.A., Balakrishnan, H., Blumberg, A.J.: VPriv: Protecting privacy in location-based vehicular services. In: USENIX Security Symposium, pp. 335–350. USENIX Association (2009)
18. Rupp, A., Baldimtsi, F., Hinterwalder, G., Paar, C.: Efficient and privacy-preserving payments in transportation systems: Cryptographic theory meets practice (2013), `http://homepage.rub.de/andy.rupp/papers/p4r_full.pdf`
19. Sadeghi, A.R., Visconti, I., Wachsmann, C.: User privacy in transport systems based on RFID e-tickets. In: Bettini, C., Jajodia, S., Samarati, P., Wang, X.S. (eds.) PiLBA. CEUR Workshop Proceedings, vol. 397. CEUR-WS.org (2008)
20. Zhang, H., Gummeson, J., Ransford, B., Fu, K.: Moo: A batteryless computational RFID and sensing platform. Tech. Rep. UM-CS-2011-020, Department of Computer Science, University of Massachusetts Amherst, Amherst, MA (June 2011), `https://web.cs.umass.edu/publication/docs/2011/UM-CS-2011-020.pdf`

Coupon Collector's Problem for Fault Analysis against AES — High Tolerance for Noisy Fault Injections

Yu Sasaki[1], Yang Li[2], Hikaru Sakamoto[2], and Kazuo Sakiyama[2]

[1] NTT Secure Platform Laboratories
sasaki.yu@lab.ntt.co.jp
[2] The University of Electro-Communications
{liyang,sakiyama}@uec.ac.jp

Abstract. In this paper, we propose a new technique for SQUARE Differential Fault Analysis (DFA) against AES that can recover a secret key even with a large number of noisy fault injections, while the previous approaches of the SQUARE DFA cannot work with noise. This makes the attack more realistic because assuming the 100% accuracy of obtaining intended fault injections is usually impossible. Our success lies in the discovery of a new mechanism of identifying the right key guess by exploiting the coupon collector's problem and its variant. Our attack parameterizes the number of noisy fault injections. If the number of noisy faults is set to 0, the analysis becomes exactly the same as the previous SQUARE DFAs. Then, our attack can work even with a large number of noisy faults. Thus our work can be viewed as a generalization of the previous SQUARE DFAs with respect to the number of tolerable noisy fault injections.

Keywords: AES, Fault analysis, DFA, Noisy fault model, SQUARE DFA, Coupon collector's problem.

1 Introduction

Fault analysis is one of the major approaches of the side-channel analysis. In particular, Differential Fault Analysis (DFA) has been applied to very wide range of ciphers. The suitable fault injection rounds for DFA (DFA round) are almost uniquely determined depending on the structure of the target cipher. These DFA rounds are the first place to be protected for the DFA countermeasures. A common approach is to take redundancy in hardware cost or time to trade the ability of detecting faulty calculations [1–3]. When the faults are injected before the DFA rounds, the complexity of the key recovery process usually becomes impractical in a straightforward DFA application.

For the AES-128 encryption [9], many DFA results [4–8] imply that the DFA rounds are 8 and 9. Even after the countermeasure is proposed so that rounds 8 and 9 are protected, researchers have proposed practical DFA variants based on

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 213–220, 2013.

Table 1. Comparison of SQUARE DFA

Approach	α	#Tolerable noise	Time	#Fault Injections
[10]	256	0	$2^{36.68}$	255
[12]	44	0	$2^{34.14}$	43
Ours	256	1610	$2^{44.54}$	59712
	128	49	$2^{41.15}$	5664
	64	13	$2^{39.94}$	2464

the fault injected at round 7 such as SQUARE DFA [10], impossible DFA [10, 11], and Meet-in-the-Middle (MitM) DFA [11].

The SQUARE DFA exploits a property of the AES algorithm, *i.e.*, if the attacker collects 256 ciphertexts where one byte at the beginning of round 7 takes all 256 distinct values and other bytes are fixed to a constant, in a few rounds after, all bytes take exactly 256 distinct values with probability 1. The illustration of the property is available in Fig. 1. Then, the attacker guesses the last round key and performs a partial inverse computations for 256 ciphertexts. The correctness of the key guess is verified by checking whether or not 256 distinct values are observed. The attack requires to obtain 255 distinct fault values on a target byte with exactly 255 fault injections. Hence, the attack cannot accept the noise, and [10] assumed the bit-fault model.

Kim improved the SQUARE DFA based on the observation that 256 distinct values are not necessary to recover the key [12]. The attacker collects only α distinct values at the beginning of round 7, and thus the number of fault injections is reduced and the attack model is relaxed to the random-byte fault model. The attack requires to obtain $\alpha - 1$ distinct fault values on a target byte with exactly $\alpha - 1$ fault injections. Hence, the attack cannot accept the noise.

In practice, fault injections cannot assure the 100% accuracy of the intended fault injections. As a result, it is necessary to introduce the *noisy fault model.*, where random faulty ciphertexts corresponding to unintended fault injections are mixed in the data to be analyzed for recovering a key.

Our Contributions. We propose a new technique for the SQUARE DFA, which can recover the key even with a large number of noisy fault injections.

Assume that an attacker can inject an intended fault every two trials. First of all, the attacker obtains one ciphertext without injecting the fault. In order to collect $\alpha - 1$ intended faulty ciphertexts, the same number of noisy ciphertexts are obtained, and the total number of data to be analyzed is $n = 2(\alpha - 1) + 1$. Our attack still can find the key with such a high probability of obtaining noise at a small additional cost. See Table 1 for the comparison with previous work.

Our attack collects α distinct values at the beginning of round 7. However, the attacker also obtains $n - \alpha$ unintended ciphertexts due to the noise, and she does not know which are intended ones. For the right key guess, at least α distinct values appear after the partial decryption. If the probability of this event is small enough, the key space is reduced. The probability of this event is estimated by solving the coupon collector's problem and its variant, Let β be

the size of each cell of the cipher ($\beta = 256$ for AES). The probability that a key guess is regarded as a right key candidate, *i.e.*, α distinct values appear in one byte after the partial decryption is equivalent to the probability where at least α out of β coupons are collected with n coupon drawing events.

2 Related Work

2.1 Specification of AES

AES-128 [9] is a 128-bit block cipher using a 128-bit key. It consists of 10 rounds. Subkeys sk_i for round i are generated from the original secret key K. At first, subkey sk_0 is XORed to the plaintext. Then a round function $AK \circ MC \circ SR \circ SB$ is iteratively applied, where SB is an S-box transformation, SR is a byte shift, MC is a multiplication by an MDS matrix, and AK is a subkey XOR. Note that MC is omitted for the last round. We denote the initial state in round x by $\#x^I$. Then, states immediately after each operation in round x are denoted by $\#x^{SB}$, $\#x^{SR}$, $\#x^{MC}$, and $\#x^{AK}$. Obviously, $\#x^{AK}$ is identical with $\#(x+1)^I$. We denote 4-byte positions in column j of state $\#S$ by $\#S[Col(j)]$.

2.2 Square DFA on AES

Phan and Yen showed that the key is recovered with the fault injected at the beginning of round 7 [10]. The attack uses the idea of the Square attack, which was firstly proposed by Daemen *et al.* for the Square cipher.

The attack collects 256 distinct values on a single byte at the beginning of round 7. The attacker firstly obtains a ciphertext. While the same plaintext is encrypted 255 times. she injects a fault on a target byte. The fault model by Phan and Yen [10] is summarized as follows.

Fault model 1 (bit-fault and deterministic-fault)

- *An attacker can inject the fault in any intended bit position, and moreover, several target bits can be flipped simultaneously.*
- *The fault injection to the target bits always succeeds.*

All in all, the attacker can obtain 255 distinct fault values with 255 injections without obtaining any noisy data. 256 distinct values for a single byte at $\#7^I$ result in 256 distinct values for all bytes at $\#9^{AK\prime}$ as shown in Fig. 1. Note that the order of the linear operations are exchanged in rounds 9 and 10.

After 256 ciphertexts following the data structure of Fig. 1 are obtained, the attacker recovers the converted last round key sk'_{10}, where $sk'_{10} = SR^{-1}(sk_{10})$. The key recovery phase is described in Fig. 2. At first, the inverse of SR is applied to all ciphertexts. Then, the attacker guesses sk'_{10} column by column. Let us focus our attention on column 0, which is colored in Fig. 2. 2^{32} value of $sk'_{10}[Col(0)]$ are exhaustively guessed. For each guess, the attacker decrypts $\alpha = 256$ converted ciphertexts $(C'_0, \ldots, C'_{255})$ up to $\#9^{AK'}[Col(0)]$. Then, the right key can be identified based on the following mechanism.

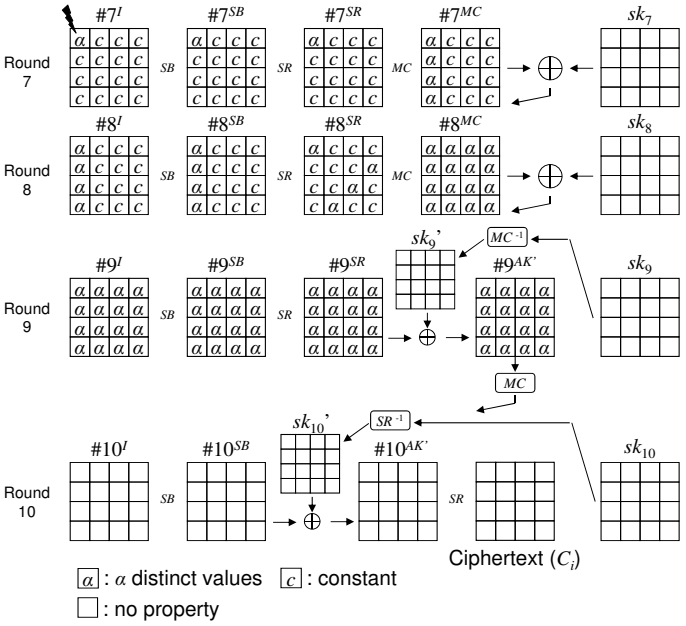


Fig. 1. Fault injection at $\#7^I$. The key is a fixed value but the notation c is omitted.

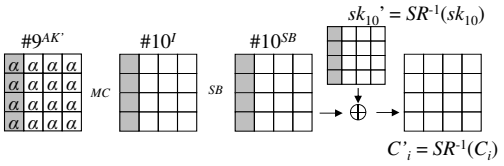


Fig. 2. The structure of the key recovery phase

Key Recovery Mechanism 1: *For the right guess,* 256 *distinct values appear for each byte of* $\#9^{AK'}[Col(0)]$*. This is unlikely to occur for wrong guesses. In fact, the probability of this event for a wrong guess is*

$$\left(\prod_{i=0}^{255}\frac{(256-i)}{256}\right)^4 \approx 0. \tag{1}$$

Finally, the right key value is recovered with 255 fault injections and about $2^{32}\cdot 256\cdot 1/10 \approx 2^{36.68}$ AES computations.

2.3 Improved SQUARE DFA with a Small Number of Fault Injections

Kim proposed an improved SQUARE DFA on AES [12]. The entire attack structure again follows Fig. 1. Suppose that the attacker obtains α distinct values

at the target single byte at $\#7^I$, where $1 \leq \alpha \leq 256$. The only difference from the original SQUARE DFA [10] is the value of α. Phan and Yen only considered $\alpha = 256$ while Kim regarded it as a variable ranging from 1 to 256. Due to the same reason as [10], each byte at $\#9^{AK'}$ takes α distinct values.

Fault model 2 (random byte-fault and deterministic fault)
- *An attacker can inject $\alpha - 1$ distinct fault values in an intended byte position. Fault value is uncontrollable and is uniformly distributed.*
- *The fault injection to the target byte always succeeds.*

For a fixed α, the right key can be identified based on the following mechanism.

Key Recovery Mechanism 2: *For the right guess, α distinct values appear for each byte of $\#9^{AK'}[Col(0)]$. If this event is unlikely to occur for wrong guesses, the right key can be identified. The probability of this event for a wrong guess is*

$$\Big(\prod_{i=0}^{\alpha-1} \frac{(256-i)}{256}\Big)^4. \tag{2}$$

Kim showed that when $\alpha = 44$, Eq. (2) is smaller than 2^{-32} and thus the right key is recovered. Finally, the right key value is recovered with 43 fault injections and about $2^{32} \cdot 44 \cdot 1/10 \approx 2^{34.14}$ AES computations.

3 SQUARE DFA Based on Coupon Collector's Problem

We generalize the previous SQUARE DFA to accept the noise. Our attack has two parameters. 1) how many distinct values can be obtained at $\#7^I$, α, and 2) how many ciphertexts are obtained before α intended ciphertexts are collected, n. Note that n is the sum of the numbers of intended and unintended ciphertexts. If we set the number of noisy fault to 0 ($n = \alpha$), the analysis becomes the same as [12]. If we further set $\alpha = 256$, the analysis becomes the same as [10]. Hence, we call our approach generalized SQUARE DFA. For the further generalization, we also parameterize the size of each cell by β, e.g. $\beta = 256$ for AES.

3.1 Overview

The parameters of the attacker's abilities α and n basically depend on the environment e.g., how much cost can be spent for the attack. We assume that the attacker's ability is independently measured before the analysis is started.

Fault model 3 (random byte-fault and probabilistic-fault)
- *An attacker can inject at most $\alpha - 1$ distinct fault values in an intended byte position, The fault value is uncontrollable and is uniformly distributed.*
- *The intended fault injection can be obtained only probabilistically, with a probability of $(\alpha - 1)/(n - 1)$.*

The attack firstly obtains 1 ciphertext without the fault injection. While the same plaintext is encrypted $n-1$ times, the attacker injects the fault $n-1$ times and obtains the corresponding $n-1$ ciphertexts. $\alpha-1$ intended faulted ciphertexts are included in the $n-1$ faulted ciphertexts, but which ciphertexts are intended ones is unknown. After collecting such $1+(n-1)=n$ ciphertexts, we recover the last round key sk'_{10} column by column.

> **Key Recovery Mechanism 3:** *For the right guess, at least α distinct values appear for each byte of $\#9^{AK'}[Col(0)]$. If this event is unlikely to occur for wrong guesses, the right key can be identified.*

Therefore, for each guess of $sk'_{10}[Col(0)]$, if less than α distinct values are observed in at least one byte of $\#9^{AK'}[Col(0)]$, we know the guess is wrong, and thus the key space is reduced. Let $P(\alpha,\beta,n)$ be the probability that, for each key guess, at least α distinct values appear for one byte of $\#9^{AK'}[Col(0)]$. The key space of $sk'_{10}[Col(0)]$ is reduced from β^4 to $\beta^4 \times P(\alpha,\beta,n)^4$ after the analysis.

For simplicity, let us discuss the case where $\alpha=\beta$, which implies that all values are collected at $\#7^I$. For AES, $\alpha=\beta=256$. In this case, $P(\alpha,\beta,n)$ is equivalent to the success probability of the coupon collector's problem.

> *There are β coupons. With 1 coupon-drawing event, 1 coupon which is uniformly distributed from β coupons is obtained. The success probability of the coupon collector's problem for parameters (β,n) is the one that all β coupons are completed with n coupon-drawing events.*

For $\alpha<\beta$, $P(\alpha,\beta,n)$ is equivalent to a variant of the coupon collector's problem where at least α kinds out of β kinds of coupons are collected with n coupon-drawing events. In the next section, we will evaluate the value of $P(\alpha,\beta,n)$.

3.2 Probability Evaluation of $P(\alpha,\beta,n)$

Proposition 1 *Let $Q(\alpha,n)$ be the number of permutations of n coupons including all of the α different coupons completely $(1\le\alpha\le n)$. Then*

$$Q(\alpha,n)=\sum_{k=1}^{\alpha}\left((-1)^{\alpha-k}\binom{\alpha}{k}k^n\right). \tag{3}$$

Proof. As $Q(j,n)\binom{\alpha}{j}$ covers all the possible ways to collect j different coupons chosen from α coupons $(j\le\alpha)$ using n random coupon-drawing trials, its summation for $j=1,2,\ldots,\alpha$, where $\alpha=0,1,\ldots,n$, always becomes α^n as

$$\sum_{k=1}^{\alpha}\left(Q(k,n)\binom{\alpha}{k}\right)=\alpha^n, \tag{4}$$

For $a_n=\sum_{k=1}^{n}\binom{n}{k}b_k$, we have $b_n=\sum_{k=1}^{n}(-1)^{n-k}\binom{n}{k}a_k$. Therefore for Eq. (4), consider α^n as a_n and $Q(k,n)$ as b_k, we can derive Eq. (3). □

Proposition 2 *Let $P(\alpha, \beta, n)$ be the probability that one collects at least α out of β coupons through n trials, where $2 \leq \alpha \leq \beta$. Then*

$$P(\alpha, \beta, n) = \binom{\beta}{\alpha}\binom{\alpha}{1}\sum_{i=\alpha}^{n}\frac{Q(\alpha-1, i-1)}{\beta^i}, \tag{5}$$

Proof. We have the probability that one collects α coupons out of β exactly at the i-th trial, *i.e.*, $\alpha - 1$ coupons are collected through the $i-1$ trials, as

$$\frac{Q(\alpha-1, i-1)\binom{\beta}{\alpha-1}\binom{\beta-\alpha+1}{1}}{\beta^i} = \binom{\beta}{\alpha}\binom{\alpha}{1}\frac{Q(\alpha-1, i-1)}{\beta^i}.$$

As the probability of $P(\alpha, \beta, n)$ equals the summation of the above probabilities for $i = \alpha, \alpha+1, \ldots, n$, Eq. (5) is derived. □

4 Details of the Generalized SQUARE DFA on AES-128

The parameters (α, n) should be measured depending of the attacker's ability. The parameter β is fixed to 256. In this section, we explain the attack with the parameter $(\alpha, \beta, n) = (256, 256, 1553)$, where $P(\alpha, \beta, n) = 1/2$ according to Eq. (5). Note that $(\alpha, n) = (256, 1553)$ is just an example for the case study.

The attack consists of the data collecting phase and the key recovery phase. The procedure of the data collecting phase is as follows.

1. Obtain a ciphertext C_0 without injecting the fault, and store it in a table.
2. While the same plaintext is encrypted, try to inject a fault in a target single byte at $\#7^I$. If the obtained ciphertext does not overlap with already stored ones in a table, add this ciphertext into a table. Otherwise, discard it.
3. Repeat Step 2 until n ciphertexts $C_0, \ldots, C_{n-1}$ are obtained.

For one plaintext, a set of n ciphertexts is constructing. For $(\alpha, n) = (256, 1553)$, we collect 8 sets of n ciphertexts by using 8 different plaintexts.

The input to the key recovery phase is 1553 ciphertexts $C_0, \ldots, C_{1552}$, where 256 values are collected in the target single byte at $\#7^I$. Then, do as follows.

1. Compute $C'_i \leftarrow SR^{-1}(C_i)$ for $0 \leq i \leq 1552$.
2. For each column ($j = 0, 1, 2, 3$), exhaustively guess the value for $sk'_{10}[Col(j)]$ and compute $\#9^{AK'}[Col(j)]$.
3. If only less than 256 distinct values are observed in at least one byte of $\#9^{AK'}[Col(j)]$, discard the guessed $sk'_{10}[Col(j)]$ from the key candidates.
4. Repeat steps 1 to 3 by 8 times with changing the ciphertext set.

The attack cost is evaluated as follows. The number of the faulty ciphertexts is $8 \cdot 1553 = 12424$. In the key recovery phase, Step 1 costs $8 \cdot 1553$ inverse SR computations, which essentially does not cost anything. Step 2 costs $8 \cdot 1553 \cdot 2^{32}$ one round computations, which is $(8 \cdot 1553 \cdot 2^{32})/10 \approx 2^{42.28}$ AES computations. The memory requirement is 2^{32} AES state to count the remaining key space.

The time complexity can be optimized. Once a key candidate is identified to be a wrong key, it does not have to be examined again. Therefore, the complexity is $1553 \cdot (2^{32} + 2^{28} + 2^{24} + \cdots + 2^4)/10 \approx 2^{39.37}$ AES computations.

If not only for the ciphertext, but also the plaintext is available for the attacker, she can combine the brute force attack. Hence, step 4 can be stopped when the key space is reduced to a sufficiently small size, rather than 1.

5 Concluding Remarks

In this paper, we presented the new fault analysis called generalized Square DFA, which was an extension of the previous Square DFA with respect to the noisy fault injections. The probability that a key candidate is judged as a correct key is estimated with a coupon collector's problem and its variant.

References

1. Fischer, W.: Aspects of the development of secure and fault-resistant hardware. In: FDTC, pp. 18–22 (2008)
2. Guilley, S., Sauvage, L., Danger, J.L., Selmane, N.: Fault injection resilience. In: FDTC, pp. 51–65 (2010)
3. Satoh, A., Sugawara, T., Homma, N., Aoki, T.: High-Performance Concurrent Error Detection Scheme for AES Hardware. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 100–112. Springer, Heidelberg (2008)
4. Blömer, J., Seifert, J.-P.: Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
5. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on A.E.S. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg (2003)
6. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2004. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005)
7. Moradi, A., Shalmani, M.T.M., Salmasizadeh, M.: A Generalized Method of Differential Fault Attack Against AES Cryptosystem. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 91–100. Springer, Heidelberg (2006)
8. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
9. Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1998)
10. Phan, R.C.-W., Yen, S.-M.: Amplifying Side-Channel Attacks with Techniques from Block Cipher Cryptanalysis. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 135–150. Springer, Heidelberg (2006)
11. Derbez, P., Fouque, P.-A., Leresteux, D.: Meet-in-the-Middle and Impossible Differential Fault Analysis on AES. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 274–291. Springer, Heidelberg (2011)
12. Kim, C.H.: Efficient methods for exploiting faults induced at AES middle rounds. Cryptology ePrint Archive, Report 2011/349 (2011)

Mitigating Smart Card Fault Injection with Link-Time Code Rewriting: A Feasibility Study

Jonas Maebe, Ronald De Keulenaer, Bjorn De Sutter, and Koen De Bosschere

Computer Systems Lab, Ghent University
{jmaebe,rdkeulen,brdsutte,kdb}@elis.ugent.be

Abstract. We present a feasibility study to protect smart card software against fault-injection attacks by means of binary code rewriting. We implemented a range of protection techniques in a link-time rewriter and evaluate and discuss the obtained coverage, the associated overhead and engineering effort, as well as its practical usability.

Keywords: smart card, fault injection, software protection, binary rewriting.

1 Introduction

Cryptographic keys and PIN hashes are often embedded in bank smart cards. To steal that data, attackers inject faults with power glitches, clock period alterations, temperature rises, active probing of buses, or light attacks [2]. The faults cause bit flips to alter data values or program code. When this remains undetected, security barriers risk being broken: private keys can leak [1], encryption rounds are skipped [4], buffers can overflow [2], and critical checks can be skipped [8]. To protect software against such attacks, redundancy can be inserted in the code to detect occurring faults. Ideally, some tool can apply generic forms of low-overhead redundancy fully automatically to implement security policies specified in a convenient form without impeding the programmer's productivity.

Many redundancy schemes have been proposed in the past to protect software against soft errors [14] and to prevent control flow from deviating from predetermined paths [13]. However, all automated techniques that try to limit the performance overhead of the introduced redundancy is implemented in tools that do not cooperate well with other compilers.

In practice, companies rely on multiple in-house and third-party development tool chains that may change over time. To maintain interoperability with different tool chains and avoid vendor lock-in, tools that automate the implementation of security policies should therefore be separate tools that do not break existing tool chains and do not depend on their internal operation. This leaves two basic options to insert redundancy: source code rewriting and binary code rewriting.

Source-to-source rewriters essentially insert redundancy in the source code by duplicating statements. They suffer from major drawbacks. First, optimizing compilers risk undoing the protection by eliminating the redundancy they detect in the source code. Secondly, as security is a problem that concerns many abstraction and implementation layers, many security policies involve lower-level

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 221–229, 2013.

aspects that are hard to control in source code when the used compilers are black boxes. Thirdly, source-to-source rewriters are by definition language-dependent and hence need to be redeveloped for every programming language. Finally, white and black box security testing typically takes place on the final binary code. Communication between testing teams and protection tool developers is much harder if the former are studying assembly code generated by some black-box compiler while the latter are working on source code.

Binary code rewriters do not suffer from these problems. They do suffer, however, from the fact that they have to operate on binary code that lacks symbolic information. This limits the precision and scope of many program analyses and transformations, which affects the provided level of protection and the overhead.

This paper presents a *feasibility study* of link-time binary code rewriting to protect against fault-injection attacks. We argue why such an approach is feasible. We evaluate the coverage and overhead of link-time code rewriting that implements certain security policies. The policies we examine are cookbook recipes [10] for local hardening of code against certain classes of single-instruction failures, i.e., single instructions that are skipped as the result of an injected fault.

We know of no automated fault-injection protection tools in use today for smart card software. We believe that the need to provide this protection manually is the main reason why manual assembly programming is still so common. By providing convincing arguments for the feasibility of automating this protection in a tool that does not disrupt proprietary compiler-based tool flows, we hope to contribute a significant step towards more productive smart card programming.

2 Link-Time Software Protection against Fault Injection

Around the turn of the century, link-time rewriters were presented for performance optimization [12] and code compaction [6]. Today, compilers like gcc and LLVM also include so-called link-time optimizations. Those operate on the compiler's intermediate representation, however, not on binary code. These compilers therefore do not meet the practical requirements set forward in the introduction.

By contrast, Diablo (http://diablo.elis.ugent.be) is a true link-time rewriting framework [15]. Its flexibility, versatility, extensibility, retargetability, portability, and reliability have been demonstrated extensively in the past, which make it a promising candidate tool for the automated protection of binary code. In short, Diablo has been used to develop tools that rewrite code for a range of goals including optimization, compaction, (de-)obfuscation, anti-tampering, formal verification, instrumentation, GUI executable editing, and OS customization. These tools have been applied to binaries from different source languages, including C, C++, assembler, and Fortran. Several of those tools have been applied to binaries generated by different compiler generations covering 10 years of proprietary as well as open-source compilers (incl. ARM ADS, ARM RVCT, and gcc). Finally, the Diablo framework has been used to rewrite the Linux kernel for both ARM and x86. This kernel features artifacts such as code for physical and virtual address spaces, privileged instructions, manually written

assembler not adhering to the conventions as specified in application binary interfaces (ABIs), and a complicated, non-standard build process. For an overview of all Diablo-related research results, we refer to the Diablo website.

Together, these demonstrations show that link-time rewriters can meet the requirements discussed before: they can cooperate reliably with black box, third-party tool chains and code libraries. It is, however, not clear *a priori* that a tool like Diablo can deliver acceptable protection at acceptable overhead, with an acceptable engineering effort. Link-time tools are designed to depend only on information available in object files, such as symbol and relocation information [9]. This enables them to handle code generated by open-source as well as closed, proprietary compilers. However, this also limits their capabilities.

First, they lack high-level semantic information about the code to be rewritten. For example, no type information is available, which makes alias analysis much less precise [11]. Consequently, link-time rewriters typically need to handle memory by and large as a black box. Rather than performing register allocation globally, like compilers do, they need to find free registers locally to store temporary values. In case they cannot find them, the values have to be spilled to the stack. Applying this spilling locally in a link-time rewriter introduces considerably more overhead. Secondly, at link time indirect control flow transfers of which binary code analysis cannot resolve the targets precisely have to be modeled conservatively, i.e., over-approximated, on the basis of relocation information [5]. Through additional edges that model the over-approximation, the control flow graph (CFG) then models a superset of all possible program executions. This is safe, but leads to a loss in analysis precision. It is mainly because of these limitations that a feasibility study like this one is needed.

For this feasibility study, we implemented three first-line-of-defense cookbook protection schemes that provide local hardening against certain single-instruction failure attacks [10]. We implemented these in a tool on top of the Diablo framework for the ARM Cortex-M0 instruction set architecture (ISA), which is used in the ARM smart card SecurCore SC000 processors [16] that target future smart card applications. In these protection schemes, the program executes some "invalid state exception" code when a fault is detected. In our prototype, we opted for an infinite loop to prevent the export of any secret data.

Conditional Branch Duplication. Sensitive code paths are often shielded by checks that, e.g., verify whether a correct password was entered. On smart card processors like the ARM SecurCore SC000, these checks correspond to conditional branches in the binary code. The branches are taken or not depending on flags in a status register, that can either be set using an explicit comparison instruction or implicitly according to the result of an ALU operation. Attacks can focus on the input values used to perform the operation that sets the flags, on that operation itself, or on the conditional branch that depends on the flag.

```
    ldr r0,[r1]
    cmp r0,#5
    beq .success
    <failure handling code>
.success:
    <sensitive code>
```

Fig. 1. Original code

To protect against attacks that make the checks ineffective by skipping one of these instructions, we duplicate the computation of the flags and the conditional branch. A typical scenario targeted by this transformation is depicted in Figure 1, which is then transformed into the code of Figure 2. The shown transformation defends against all avenues of attack mentioned above: The input value is protected by duplicating the defining instruction, the flag setting is covered by duplicating the comparison, and branching based on the flag values is covered by duplicating the conditional branch.

More complex variations of the code shown in Figure 1 can occur. First, there may be multiple, different definitions of the input value(s) of the comparison in different predecessors of the comparison's basic block. In that case the duplication of the definition can be skipped, which weakens the protection because of reduced redundancy, or a significant amount of code needs to be duplicated through so-called tail duplication [11] even before the actual redundancy can be inserted. So far, we implemented only the former in our prototype. Secondly, sometimes we cannot simply duplicate the instruction defining the value to be compared because its source operands are no longer available. This is the case, e.g., when the defining instruction is a load like `ldr r0, [r0]`. In that case, we need to find a free register to store a copy of the original source operands, and use that register in the duplicate. When no free registers are found, they are created by inserting spill code. Apart from spilling data register to the stack to free them for use in the duplicated code, it can also be necessary to spill the status register when flags are used beyond the conditional branch. This adds overhead. Finally, when the flags are set by an ALU operation that overwrites a source operand with a new value, we also needs to find a free register.

```
   ldr r0,[r1]
   cmp r0,#5
   beq .dup1
   <failure handling code>
.dup1:
   ldr r0,[r1]
   cmp r0,#5
   beq .success
   <invalid state exception>
.Success:  <sensitive code>
```

Fig. 2. Protected code

Call Graph Integrity. Security analyses performed on a program's call graph are only as trustworthy as the guarantee that only modeled calls or returns can occur. By injecting bogus call or return addresses into the execution of a program, it is possible to invalidate any call graph constructed statically. Our integrity transformation works at a local level: at each individual call and return site a value is set to be checked at the intended destination. At every function entry and at every return, we then verify that control indeed came from one of the allowed points. This is less strong than a protection that verifies entire call chains, but it can be easily applied to call graphs with hard to analyze constructs such as recursion.

```
Caller:
   ...
   mov r4,#id
   blx Callee
   ...
Callee:
   cmp r4,#id
   beq .success
   <invalid state exception>
   ...
.success:
  <sensitive code>
```

Fig. 3. ID Passing & checking

Since our transformation is applied at link time, supporting indirect function calls through function pointers or polymorphic method invocations requires extra care. Lacking type information, link-time rewriters typically cannot determine the exact targets of an indirect call. This is solved by clustering all functions that

may be called indirectly (according to the symbol and relocation information found in the object files) and treating them as a single function as far as this transformation is concerned. While this makes the protection less tight, it allows dealing statically with uncertainties introduced by dynamic program behavior.

Figure 3 shows how each check consists of two parts. Before every call a register or global variable is set to the unique identifier *id* of the callee (or cluster of callees). Next, instructions inserted in each function's prologue verify whether the set value matches its identifier (or that of its cluster). Similarly, before every return instruction a register or global variable is set to another id. This value is checked at the return points in the callers.

The process starts by partitioning the program's functions into clusters whose members can call each other indirectly or that can be called from the same indirect call site. Next, the registers free on entry and exit in all functions in a cluster are collected. If some register is always free on entry, it will be used to pass the value from the caller to the callee, otherwise the value is passed via a global variable. The same happens at the function exits.

```
  str r0,[r1]
  ldr r2,[r1]
  cmp r0, r2
  beq .ok
  <invalid state exception>
.ok:
  ...
```

Fig. 4. Store verification

Memory Store Verification. The failure of a store operation at run time generally means that program state is lost. This can be addressed by checking that the store actually took place and that the correct value was written to memory. Such a transformation also introduces some resiliency to memory errors.

The proper execution of a store can be verified by loading the stored value back from memory and verifying that it matches the value that should have been stored. This happens with a comparison, as depicted in Figure 4. This transformation requires an extra free register to reload the stored value. Additionally, the status flags must be available since we insert a comparison. We do not have to duplicate the comparison in order to be safe from a single-instruction failure attack, since such an attack corrupts either the store or the comparison, but not both. Multiple attacks can be dealt with by duplicating the inserted code as many times as the number of attacks that should be handled.

Besides discussing the transformations we implemented support for, it is useful to discuss the engineering effort we invested in Diablo's core infrastructure. This demonstrates that we can build on existing infrastructure in link-time rewriters to solve technical issues of automated protection, rather than having to adapt their fundamental concepts or having to implement ad-hoc solutions.

First, inserting redundancy involves static as well as dynamic code duplication: In Figure 2, the load occurs twice in the code statically and it will be executed twice dynamically. For memory-mapped IO, this is problematic: Memory-mapped IO operations should obviously not be duplicated dynamically. So on architectures like ARM, our tool faces the problem to differentiate between normal memory operations and memory-mapped IO operations. To solve this, the software developer has to provide our tool with a list of IO register memory addresses. Diablo constant propagation analysis [11] detects instructions that access constant memory addresses. When such instructions are detected

that access IO registers, we have those instructions marked as memory-mapped IO to prevent the protecting transformations from duplicating them. This required no changes at all to Diablo's basic infrastructure. In practice, it works because memory-mapped IO is typically programmed with hard-coded constant addresses, for example through macros in the source code, for which the detection based on constant propagation works well. In theory, it is possible to write an application for which this solution will not work (and for which no fully automated solution will work, due to the undecidability of the aliasing problem).

Secondly, like all compilers, Diablo iteratively applies analyses and transformations. To simplify their implementation, most of them make some assumptions about the state of the IR. In Diablo, many analyses and transformations assume that the CFG contains no unconnected nodes. To guarantee this, an unreachable code elimination pass [11] is run before almost all analyses.

```
int some_routine(void){
    ...
    return some_value;
    check_unreachable();
}
```

Fig. 5. Unreachable code

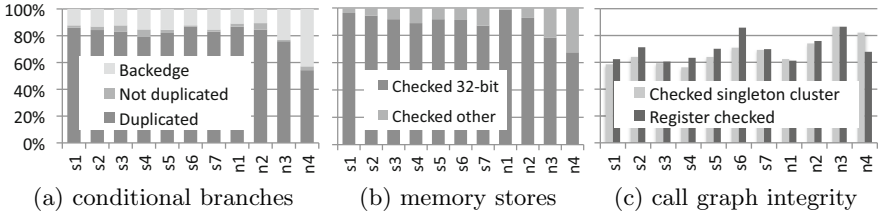
This is problematic when a programmer wrote code that is unreachable under normal, fault-free conditions as in Figure 5, but that was added for security reasons. By default, Diablo eliminates the call to `check_unreachable`. To avoid this we adapted Diablo's basic infrastructure to keep track of the program points where it deleted unreachable code. The user of our tool can provide exception handling code, which our tool will then insert at those points before writing out the code. This provides a simple mechanism to compensate for eliminated code.

3 Experimental Evaluation

To evaluated our protections, we compiled and protected seven C MiBench [7] benchmarks (see Table 1) for a semi-hosted simulation environment (QEMU 1.0 [3]) with which we verified correctness, but on which no accurate performance can be measured. We refer to these benchmark versions as s1–s7. We used the ARM RVCT 4.1 compiler for ARM Cortex-M0 platforms with -O2. This compiler is a centerpiece of Keil (http://www.keil.com/smartcards), a tool box often used in industrial smart card software development. Next, we ported four benchmarks to a USB device with an ARM Cortex-M3 with 32 kB of flash ROM and 8 kB of SRAM. This memory was too small for all benchmarks, so this limits our evaluation on real hardware to the four stripped-down benchmarks in Table 1. We will refer to these four natively executed, stripped-down programs as n1–n4. All binaries are linked statically, such that they include all needed RVCT 4.1 C library code. Whereas developers of real smart card applications would apply protections to

Table 1. Benchmarks (code size in bytes)

benchmark	domain	nr	size
full benchmarks for semi-hosted simulation			
basicmath_small	floating-point	s1	19480
bitcnts	integer bitcounting	s2	8204
qsort_large	3D point sorting	s3	14824
qsort_small	string sorting	s4	8012
susan	image processing	s5	32172
aes	crypto	s6	37092
sha	crypto	s7	7296
stripped-down benchmarks for native execution			
basicmath_small	floating-point	n1	12928
bitcnts	integer bitcounting	n2	3124
aes (encoder only)	crypto	n3	3760
sha	crypto	n4	1592



(a) conditional branches (b) memory stores (c) call graph integrity

Fig. 6. Coverage results

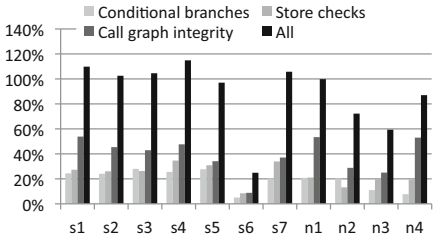


Fig. 7. Code size overhead

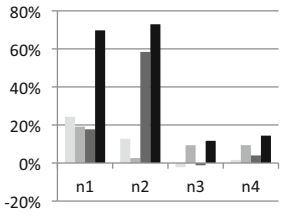


Fig. 8. Performance overhead

only the sensitive parts of the applications, there is no notion of sensitivity in our benchmarks. They serve the purpose of estimating the potential overhead of our protections, so we made our tool apply them to the whole programs (with the exception of back edges for the conditional branch duplication).

Figure 6(a) shows the fraction of all conditional branches that get protected, together with the instructions setting the condition flags and, if available in the branch's basic block, the instruction defining the operand of that instruction. From all conditional branches, 97% on average and at least 92% per benchmark can be protected with our current implementation. As for the small fraction of branches not being duplicated, this was mainly the result of not finding the flag-setting instruction in the branch's basic block. Figure 6(b) shows that our prototype was able to insert checks for 100% of the stores. We differentiate between 32-bit and other stores because the latter require an additional instruction to mask the 32 bits of which 16 or 8 are stored. The bars in Figure 6(c) show the fractions of all call sites and return points of at which the call graph integrity is checked. The first bar depicts the number of points checked with a strong check, i.e., for which the callee is in a singleton cluster. The second bar depicts the number of points at which the identifier is passed in a register for minimal overhead, not in a global variable. A relatively large number of all calls involves clustered functions, for which we can typically not find a free register. The clustering mainly happens for C-library's use of function pointer tables.

Figure 7 depicts the code size overhead of our transformations. The combined overhead varies between 25–115%. The main reason why the combined protection overhead is bigger than the sum of the isolated overheads is that Diablo's liveness analysis becomes less precise after transformations have been applied. Another reason is that as the programs grow bigger, the corresponding ever larger branch offsets can no longer be encoded in single 16-bit Thumb2 instructions. Compared

to the size overhead, the performance overhead depicted in Fig. 8 for the benchmarks for which we could conduct precise measurements is relatively small. We should remind the reader that we blindly applied the protections to the whole benchmarks, which inflates the overhead. In reality, smart card developers will likely limit them to the sensitive parts of their applications.

4 Conclusions

From previously demonstrated capabilities to reliably cooperate with black-box, third-party, industrial-strength proprietary compilers, combined with experimental results obtained with our prototype tool, we conclude that automated link-time fault-injection protection is a realistic, promising direction.

References

1. Aumüller, C., Bier, P., Fischer, W., Hofreiter, P., Seifert, J.P.: Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 260–275. Springer, Heidelberg (2003)
2. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. Cryptology ePrint Archive, Report 2004/100 (2004)
3. Bellard, F.: QEMU, a fast and portable dynamic translator. In: Proc. USENIX, pp. 41–46 (2005)
4. Choukri, H., Tunstall, M.: Round reduction using faults. In: Proc. FDTC, pp. 13–24 (2005)
5. De Sutter, B., De Bus, B., De Bosschere, K.: Link-time binary rewriting techniques for program compaction. ACM Trans. Prog. Lang. and Syst. 27(5), 882–945 (2005)
6. Debray, S.K., Evans, W., Muth, R., De Sutter, B.: Compiler techniques for code compaction. ACM Trans. Prog. Lang. and Syst. 22(2), 378–415 (2000)
7. Guthaus, M., Ringenberg, J., Ernst, D., Austin, T., Mudge, T., Brown, R.: Mibench: A free, commercially representative embedded benchmark suite. In: Proc. IEEE WWC-4, pp. 3–14 (2001)
8. Kim, C.H., Quisquater, J.-J.: Fault attacks for CRT based RSA: New attacks, new results, and new countermeasures. In: Sauveron, D., Markantonakis, K., Bilas, A., Quisquater, J.-J. (eds.) WISTP 2007. LNCS, vol. 4462, pp. 215–228. Springer, Heidelberg (2007)
9. Levine, J.R.: Linkers and Loaders. Morgan Kaufmann Publishers Inc. (1999)
10. Markantonakis, C., Mayes, K., Tunstall, M., Sauveron, D., Piper, F.: Smart card security. In: Nedjah, N., Abraham, A., de Macedo Mourelle, L. (eds.) Computational Intelligence in Information Assurance and Security. SCI, vol. 57, pp. 201–233. Springer, Heidelberg (2007)
11. Muchnick, S.S.: Advanced Compiler Design and Implementation. Morgan Kaufmann (1997)
12. Muth, R., Debray, S.K., Watterson, S., De Bosschere, K.: Alto: a link-time optimizer for the Compaq alpha. Softw. Pract. Exper. 31(1), 67–101 (2001)

13. Oh, N., Shirvani, P.P., McCluskey, E.J.: Control-flow checking by software signatures. IEEE Trans. Reliability 51(1), 111–122 (2002)
14. Reis, G.A., Chang, J., Vachharajani, N., Rangan, R., August, D.I.: SWIFT: Software implemented fault tolerance. In: Proc. ACM CGO, pp. 243–254 (2005)
15. Van Put, L., Chanet, D., De Bus, B., De Sutter, B., De Bosschere, K.: DIABLO: a reliable, retargetable and extensible link-time rewriting framework. In: Proc. ISSPIT, pp. 7–12 (2005)
16. Yiu, J.: The Definitive Guide to the ARM Cortex-M0. Newnes (2011)

On the Need of Physical Security for Small Embedded Devices: A Case Study with COMP128-1 Implementations in SIM Cards

Yuanyuan Zhou[1], Yu Yu[2],
François-Xavier Standaert[3], and Jean-Jacques Quisquater[3]

[1] Brightsight, Delft, The Netherlands
[2] East China Normal University and Tsinghua University, China
[3] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium

Abstract. Ensuring the physical security of small embedded devices is challenging. Such devices have to be produced under strong cost constraints, and generally operate with limited power and energy budget. However, they may also be deployed in applications where physical access is indeed possible for adversaries. In this paper, we consider the case of SIM cards to discuss these issues, and report on successful side-channel attacks against several (old but still deployed) implementations of the COMP128-1 algorithm. Such attacks are able to recover cryptographic keys with limited time and data, by measuring the power consumption of the devices manipulating them, hence allowing cards cloning and communications eavesdropping. This study allows us to put forward the long term issues raised by the deployment of cryptographic implementations. It provides a motivation for improving the physical security of small embedded devices early in their development. We also use it to argue that public standards for cryptographic algorithms and transparent physical security evaluation methodologies are important tools for this purpose.

Keywords: side-channel analysis, hardware security, embedded devices.

1 Introduction

Protecting present information systems requires considering both hardware and software security issues, with their specific risks and constraints. In general, software attacks are cheaper and tools for performing them can be rapidly disseminated. Yet, they are also easier to patch with code updates. By contrast, hardware attacks are more difficult to perform, as they require laboratory equipment that ranges from low-cost to highly expensive. But they can be more difficult to fix a posteriori, as hardware updates imply more expensive development processes, and usually take place in the longer term. Hence, finding the best balance between hardware and software security is a difficult task for system designers. This concern is particularly critical with cryptographic implementations that may be the target of fault insertion attacks [2] and side-channel attacks [11,12,19]. In the latter case (that will be our focus in this paper), the adversary exploits physical information leakage such as the power consumption

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 230–238, 2013.

of the device running a cryptographic algorithm, in order to extract secret information such as secret keys. As the power consumption of a device is expected to be correlated with the data it manipulates, these attacks essentially proceed by comparing key-dependent leakage predictions with actual measurements. When no particular care is taken, cryptographic implementations frequently turn out to be highly susceptible to side-channel attacks, as recently exhibited with results against the KeeLoq remote keyless entry systems (at CRYPTO 2009 [8]), the Mifare DESFire contactless smart cards (at CHES 2011 [18]), or Xilinx's FPGA bitstream encryption mechanisms (at ACM CCS 2011 [16]).

Since side-channel attacks do not target algorithms but instances of their implementation in various technologies, it is hard to design general solutions that allow making *any* implementation of an algorithm secure. Hence, state-of-the-art techniques to improve security against such attacks rely on heuristic assumptions (e.g. the masking and hiding in [14]), and need to be confirmed by empirical evaluation. Note that although this situation raises challenging research problems (e.g. discussed at the CHES workshops [6]), producing practically secure integrated circuits is not out of reach. Nowadays, most smart card companies have products evaluated by independent laboratories and granted with high security levels by certification authorities, e.g. [1,5]. But this improved security usually comes at the cost of implementation overheads that may limit their practical deployment. In addition, and although having certificates may be a good selling point, obtaining them also takes time and money (see, e.g. the Common Criteria [7] and EMVco [9]). Hence, while such certificates are a frequent requirement for security products of government agencies and banking applications, they are less usual in lower-cost applications using SIM or transport cards.

A typical example of this lack of general approaches for preventing side-channel attacks was actually given by a team from IBM in 2002, for implementations of the COMP128-1 algorithm used in GSM communications. In a paper from IEEE S&P [20], Rao et al. first showed that a straightforward application of Differential Power Analysis (DPA) was not successful against the instances of SIM cards they were analyzing (presumably because of some ad hoc countermeasures). Then, they observed that at the first round of COMP128-1's compression function, the substitution-box (S-box) consists of 512 values (i.e. are accessed by a 9-bit index). It implies that on low-speed SIMs (with 8-bit CPU) this S-box has to be implemented using two (typically equal-size) lookup tables. Knowing which table is being accessed (which could be identified from the power traces) could result in a key recovery with a maximum of 1000 random challenges, or 255 chosen ones, or just 8 adaptively chosen ones (i.e. as efficient as a binary search). This data corresponds to the monitoring of a few minutes of SIM card operations. In other words, while the standard DPA approach did not directly lead to successful key recoveries, a slightly modified path taking advantage of the implementation specificities did a perfect job. Fortunately, the attack (exploiting the 8-bit addressing) was only applicable to 8-bit-CPU SIM cards. Since 2003, the major operators have been gradually phasing out the use of legacy SIM by issuing products equipped with 16-bit CPU data bus, ruling out this possibility.

In this paper, we take advantage of this SIM card example to discuss the practical challenges raised by hardware security issues. For this purpose, we investigate the resistance of SIM cards from two different GSM operators and four different manufacturers against DPA. Our experiments target implementations of the COMP128-1 algorithm in 16-bit CPUs, that are secure against the IBM 2002 attack. They are also secure against the algorithmic collision attacks described in [3]. While COMP128-1 is progressively being replaced by improved versions, it is still deployed in commercial devices, and sometimes being distributed. We show how DPA can be used to recover its 128-bit secret key, allowing cards cloning and communications eavesdropping. Depending on the targets and measurement setup available to the adversary, the attacks require physical access to the device ranging from minutes to a couple of hours. Interestingly, our results can be seen as the methodological counterpart of the 2002 ones. While the previous analysis in [20] targets instances of SIM cards (presumably) secure against standard DPA attacks but weak against dedicated ones, our instances are robust against the IBM attack but weak against standard DPA.

The important conclusions of this work are methodological. First, our results exhibit the long term nature of physical security concerns. While cryptographic implementations are not deployed as long as algorithms, they may remain in service for a couple of years, and are not straightforward to upgrade. This observation makes a case for considering physical security as an important feature of small embedded devices in general. Technical solutions exist to make side-channel attacks significantly more difficult to perform, e.g. the previously mentioned masking and hiding. But they work best if considered early in a design process. Second, we observe that public standards for cryptographic algorithms are useful to improve the efficiency of countermeasures against physical attacks. By contrast, the closed-source nature of COMP128-1 has significantly limited the amount of research about its secure implementations. Finally, transparent and reproducible (possibly standardized) methodologies for physical security evaluations are required, in order to quantify physical security on a sound basis.

Contact with the Operators. Our experiments have been performed in 2010. The different operators exploiting the SIM cards that we discuss in this paper have been contacted before publication of our results. Updates towards implementations of COMP128-2 and COMP128-3, including protections against side-channel attacks, are under development (or maybe already deployed).

2 Background

For place constraints, details about the GSM infrastructure and previous works on SIM cloning fraud and countermeasures have been deferred to the long version of the paper [23]. In this section, we briefly recall the processing of the compression function in COMP128-1 and necessary basics on side-channel attacks.

COMP128-1 is a cryptographic hash function that takes a 32-byte input (i.e. a 16-byte challenge RAND and a 16-byte secret key KI), and produces a 12-byte output by iterating 8 rounds. In our attacks, the most important part

of this algorithm is its compression function that consists of 5 (sub-)rounds that combine the key material and randomness. In particular, the sensitive operations that we will target are the following data update occurring in the first round:

$$y = (\mathsf{KI}[m] + 2{\cdot}\mathsf{RAND}[m]) \bmod 2^{9-j},$$
$$z = (2{\cdot}\mathsf{KI}[m] + \mathsf{RAND}[m]) \bmod 2^{9-j},$$

that occur for $0 \leq m \leq 15$ secret key bytes (with j the RAND byte index).

Side-channel attacks generally exploit the existence of data-dependent and physically observable phenomenons caused by the execution of computing tasks in present microelectronic devices. Typical examples of such information leakages include the power consumption and the electromagnetic radiation of integrated circuits. We will focus on the first one in the rest of this paper. The literature usually divides such attacks in two classes. First, Simple Power Analysis (SPA) attempts to interpret the power consumption of a device and deduce information about its performed operations. This can be done by visual inspection of the power consumption measurements in function of the time. SPA in itself does not always lead to key recovery. For example with block ciphers, distinguishing the encryption rounds does not reveal any sensitive information. Yet, it is usually an important preliminary step in order to reduce the computational requirements of more advanced attacks. Second, Differential Power Analysis (DPA) intends to take advantage of data-dependencies in the power consumption patterns. In its standard form [15], DPA is based on a divide-and-conquer strategy, in which the different parts of a secret key (usually denoted as "subkeys") are recovered separately. The attack is best illustrated with an example. Say one targets the first round of a block cipher, where the plaintext is XORed with a subkey and sent through a substitution box S. DPA is made of three main steps:

1. For different plaintexts x_i and subkey candidates k^*, the adversary predicts intermediate values in the target implementation. For example, one could predict S-box outputs and get values $v_i^{k^*} = \mathsf{S}(x_i \oplus k^*)$.
2. For each of these predicted values, the adversary models the leakages. For example, if the target block cipher is implemented in a CMOS-based microcontroller, the model can be the Hamming weight (HW) of the predicted values. One then obtains modeled leakages $m_i^{k^*} = \mathsf{HW}(v_i^{k^*})$.
3. For each subkey candidate k^*, the adversary compares the modeled leakages with actual measurements, produced with the same plaintexts x_i and a secret subkey k. In the univariate DPA attacks (that we will apply), each $m_i^{k^*}$ is compared independently with many single points in the traces, and the subkey candidate that performs best is selected by the adversary.

Different statistical tools have been proposed to perform this comparison. In our experiments, we will consider a usual DPA distinguisher, namely Pearson's correlation coefficient [4]. In this case, and denoting a leakage sample produced with plaintext x_i and subkey k as l_i^k, the adversary selects the subkey candidate as:

$$\tilde{k} = \underset{k^*}{\operatorname{argmax}} \frac{\sum_i (m_i^{k^*} - \overline{m}^{k^*}) \cdot (l_i^k - \overline{l}^k)}{\sqrt{\sum_i (m_i^{k^*} - \overline{m}^{k^*})^2 \cdot \sum_i (l_i^k - \overline{l}^k)^2}},$$

where $\overline{m}^{k^*}$ and $\overline{l}^k$ are the sample means of the models and leakage samples. The complete master key is recovered by repeating this procedure for every subkey.

3 DPA Attacks against Implementations of the COMP128-1 Algorithm in SIM Cards

3.1 Target SIM Cards and Measurement Setup

In this section, we perform DPA attacks on four representative SIM cards denoted as #1,#2, #3 and #4. Besides corresponding to various operators and manufacturers, the main difference between these implementations is that they sometimes include protections against the algorithmic collision attacks described in [3], next denoted as the "Indexed Challenges" (I-C) and "Collision Free" (C-F) countermeasures. The details of these countermeasures are not necessary for the understanding of the paper, but are given in the long version [23]. As summarized in Table 1, SIM#1 and SIM#2 are susceptible to collision attacks in 20 000 and more queries, SIM#3 and SIM#4 are immune against them.

Table 1. Target SIM cards

	Manufact.	Operator	Countermeasure(s)
SIM#1	I	A	Not Available
SIM#2	II	B	I-C
SIM#3	III	B	I-C + C-F
SIM#4	IV	B	I-C + C-F

We used a LeCroy WavePro 950 oscilloscope to acquire the power traces, via a small resistor of 25 Ohm between the GND of power supply and the GND of a self-made Card-to-Terminal adapter. The Card-to-Terminal adapter was tweaked to provide an external DC power to the test card via a Kenwood P18A power supply (+5V), and to provide an external clock to the card via an Agilent 33120A function generator(5MHz Frequency, 2.2V Amplitude and 1.1V Offset). We used a commercially available card reader and software to control the test card during the acquisitions. In addition, we used a Keithley 488 GPIB card (i.e. a PCI card installed inside a PC) to communicate with the oscilloscope.

3.2 Preprocessing of the Traces

As usual when implementing side-channel attacks, we started by applying SPA in order to identify the relevant parts of the power traces. This task is easy for

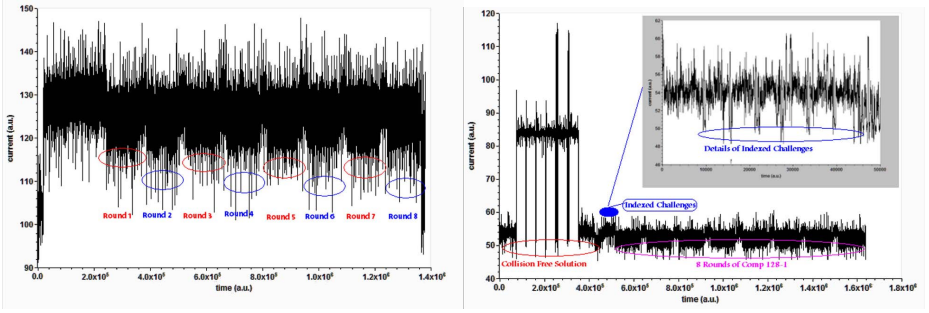


Fig. 1. Left: a power trace from SIM#1. Right: a power trace from SIM#3.

SIM#1 and SIM#2. As shown in the left part of Figure 1, we can identify the 8 iterative rounds of COMP128-1 by visual inspection. By further zooming on the different iterations, we could even observe the 5 sub-rounds of the COMP128-1 compression function (see Figure 2 in [23]). Therefore, it is directly possible to extract the parts of the power traces where to apply DPA for these two targets. The situation slightly differs for SIM cards #3 and #4, where the Collision Free countermeasure was implemented. As illustrated in the right part of Figure 1 (and Figure 6 in [23]), it is again possible to identify the COMP128-1 operations (as well as the Indexed Challenges) in the power traces. Yet, the Collision Free countermeasure includes a randomized memory writing operation (i.e. it uses randomness to decide whether to store a current request or not). Therefore, the length of the power traces varies for different inputs, which requires special care for aligning the traces after acquisition. In order to deal with this situation, a simple solution is to apply pattern matching techniques. That is, we selected a characteristic pattern including the samples of interest for our DPA attacks, and then systematically identified them in following traces using cross-correlation. As the noise level in our measurements was relatively low, such a simple heuristic was sufficient for performing successful key recoveries, as will be described next.

3.3 DPA Attack Results

Since no countermeasures in our target SIM cards prohibit random queries, we generated our traces by repeatedly executing the COMP128-1 algorithm with such inputs. Next, we applied exactly the divide-and conquer strategy focusing on the intermediate values y and z at the first sub-round of the first round in the implementation of COMP128-1, as described in Section 2. For each $0 \leq m \leq 15$, we built predictions for the 256 possible values of $\mathsf{KI}[m]$ and performed the comparison. The result of such a comparison for one of the 16 COMP128-1 subkeys is given in Figure 2 for SIM#2 and SIM#3 (similar results are given for the other targets in [23]). The figures plot the value of Pearson's correlation coefficient over time, using y as a target value. We observe that a significant peak is distinguishable at the time samples where the computation of y actually takes place, and this peak only appears for the correct subkey candidate. As expected, the situation was slightly more challenging for SIM#3 and SIM#4. This is due

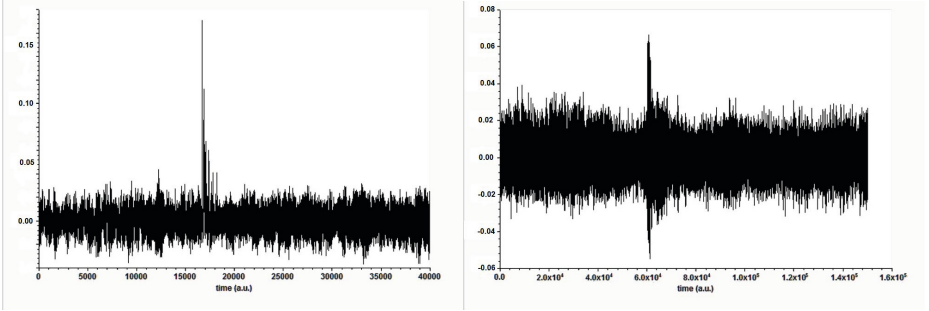


Fig. 2. Left: DPA result against SIM#2. Right: DPA result against SIM#3.

to more noisy traces and the previously mentioned synchronization issue. Yet, in both cases, a DPA peak remained clearly distinguishable, and we could always identify the COMP128-1 subkeys. Finally, we consistently recovered the full key of SIM#1 and SIM#2 with an amount of traces in the hundreds range, and this number extends to the thousands range for SIM#3 and SIM#4. These estimated data complexities are in accordance with the work of Mangard at CT-RSA 2004 [13], where it is shown that the number of measurement traces needed to recover a subkey is inversely proportional to the square of the correlation coefficient estimated for the correct key candidate. In practice, these data complexities corresponds to a few minutes to a couple of hours of acquisition.

4 Conclusions and Future Work

Technically, it is not a surprise that weakly protected chips can be defeated by side-channel attacks. Yet, our results exhibit (or recall) that such attacks are relatively easy to implement, and are certainly accessible to determined adversaries. Taking the example of SIM cards, this can have severe consequences for the security of GSM communications. Overall, the security of a system is always as strong as its weakest point. Hence, distributing cryptographically-enhanced chips without a sufficient care for physical security leads to unbalanced situations, as side-channel attacks may constitute a trapdoor to circumvent mathematical security. This is especially important for small embedded devices, for which physical access may sometimes be granted to adversaries. In this respect, it is more surprising that (somewhat) security sensitive applications do not always build on certified chips (following what is done, e.g. for bank cards). Admittedly, the target SIM cards investigated in this paper implement old versions of the GSM algorithms, in old technologies. Nevertheless, some of these cards are still in circulation and cards cloning is an important concern that could prevent the adoption of new services. Hence, this situation illustrates the long term nature of hardware security issues. It provides a general motivation for considering them as an important element to take into account early in cryptographic developments. In this respect, we note that the use of proprietary algorithms in commercial products significantly slows down progresses in securing their implementation. In view of the implementation-specific nature of physical attacks,

it frequently turns out that protection mechanisms that are tailored to certain cryptographic algorithms provide the best efficiency vs. security tradeoffs. For example, secure implementations of the AES have been the subject of a large literature over the last 10 years. By contrast, no similar analysis is available for COMP128-1. Worse, the use of large (e.g. 512-bit) tables makes it hardly suitable for implementation of countermeasures such as software masking [10]. Following this observation and in the long term, considering protections against physical attacks as a design criteria for cryptographic algorithms could be useful.

While resorting to certification would be an important step in improving the security of SIM cards, we finally note that the procedures used by evaluation laboratories could also benefit from an improved transparency. That is, currently certified chips certainly rule out the possibility of simple attacks as we describe in this paper. But it remains that the exact security level they guarantee is opaque for the end-users, and this opaqueness generally increases as countermeasures are added to the chips. Proposals of worst-case security evaluations aiming at limiting the risks of a "false sense" of security could improve this situation [21,22]. Considering the strongest available adversaries and taking advantage of the latest cryptanalytic progresses during evaluations of cryptographic hardware appears important in view of the difficulty to fix physical security breaches a posteriori. Eventually, better reflecting side-channel evaluation methodologies in public standards would be highly beneficial too. In this respect, it is noticeable that the ISO 19790 draft standard on "security requirements for cryptographic modules" (aka. FIPS-140-3 [17]) leaves the section on non-invasive attack methods essentially optional to vendors, with little details about the evaluation procedures.

Acknowledgements. Yu Yu was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61172085, 61061130540, 61073174, 61103221, 11061130539, 61021004 and 61133014. F.-X. Standaert is an associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by the ERC project 280141 on CRyptographic Algorithms and Secure Hardware.

References

1. ANSSI. Agence nationale de la securite des systemes d'information, `http://www.ssi.gouv.fr/en/products/certified-products/` (retrieved on February 1, 2012)
2. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
3. Briceno, M., Goldberg, I., Wagner, D.: GSM Cloning (1998), `http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html` (retrieved on October 14, 2011)
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
5. BSI. Federal office for information security, `https://www.bsi.bund.de/en/topics/certification/certification_node.html` (retrieved on February 1, 2012)

6. CHES, http://www.chesworkshop.org/
7. Common Criteria, http://www.commoncriteriaportal.org/
8. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Manzuri Shalmani, M.T.: On the power of power analysis in the real world: A complete break of the keeloqcode hopping scheme. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 203–220. Springer, Heidelberg (2008)
9. EMVco, http://www.emvco.com/ (retrieved on April 11, 2012)
10. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
11. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
12. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
13. Mangard, S.: Hardware countermeasures against DPA – A statistical analysis of their effectiveness. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 222–235. Springer, Heidelberg (2004)
14. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
15. Mangard, S., Oswald, E., Standaert, F.-X.: One for all – all for one: unifying standard differential power analysis attacks. IET Information Security 5(2), 100–110 (2011)
16. Moradi, A., Barenghi, A., Kasper, T., Paar, C.: On the vulnerability of fpga bitstream encryption against power analysis attacks: extracting keys from xilinx virtex-ii fpgas. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM CCS, pp. 111–124. ACM (2011)
17. National Institute of Standards and Technologies, http://csrc.nist.gov/publications/PubsDrafts.html (retrieved on March 25, 2012)
18. Oswald, D., Paar, C.: Breaking mifare desfire MF3ICD40: Power analysis and templates in the real world. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 207–222. Springer, Heidelberg (2011)
19. Quisquater, J.-J., Samyde, D.: ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
20. Rao, J.R., Rohatgi, P., Scherzer, H., Tinguely, S.: Partitioning attacks: Or how to rapidly clone some gsm cards. In: IEEE Symposium on Security and Privacy, pp. 31–44 (2002)
21. Standaert, F.-X.: Some hints on the evaluation metrics & tools for side-channel attacks. In: proceedings of the NIST non-Invasive Attacks Testing workshop, Nara, Japan (September 2011)
22. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
23. Zhou, Y., Yu, Y., Standaert, F.-X., Quisquater, J.-J.: On the Need of Physical Security for Small Embedded Devices: A Case Study with COMP128-1 Implementations in SIM Cards (long version)

Securely Solving Simple Combinatorial Graph Problems

Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet,
Olivier Pereira, and Mathieu Van Vyve

Université catholique de Louvain
ICTEAM and IMMAQ institutes
1348 Louvain-la-Neuve – Belgium

Abstract. We investigate the problem of solving traditional combinatorial graph problems using secure multi-party computation techniques, focusing on the shortest path and the maximum flow problems. To the best of our knowledge, this is the first time these problems have been addressed in a general multi-party computation setting. Our study highlights several complexity gaps and suggests the exploration of various trade-offs, while also offering protocols that are efficient enough to solve real-world problems.

Keywords: Secure multi-party computation, Graph theory, Algorithms.

1 Introduction

Secure multi-party computation – the problem of jointly evaluating a function on a set of secret inputs without leaking anything but the output of the function – has been at the center of cryptography research for almost 30 years. A first series of foundational works [1,2,3,4] demonstrated the possibility to evaluate any function in various models, the function being described as a circuit. The attention then largely focused on building solutions for the evaluation of functions of specific interest, leading to secure and efficient protocols for auctions [5], voting [6], benchmarking [7], face recognition [8] or AES evaluation [9] to only mention a few.

One common point of all these applications is that the function evaluation process is naturally oblivious of the inputs on which the function has to be evaluated. Computing the highest of n bids or summing n votes is carried out by performing n comparisons or sums independently of the values that are considered.

There are large classes of problems however for which the natural evaluation process depends on the input data. In that case, even if all the manipulated data are appropriately shared or encrypted, the execution flow might just be sufficient to leak undesirable information.

This is typically the case in combinatorial problems, of which graph problems are one of the most common examples. Consider, for instance, a consortium of delivery companies covering different territories through regular distribution circuits. These companies might be interested in computing the fastest way to bring

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 239–257, 2013.

a package from one place to another, but be reluctant to share with each other the precise connections they use and the performance of their trucks. Their problem could be solved by securely evaluating traditional shortest path algorithms such as those of Bellman-Ford or Dijkstra. The immediate way of securely computing the shortest path would be to blind (encrypt or share) the weight of all the edges of the corresponding graph. However this approach could completely miss its purpose depending on the graph encoding and shortest path algorithm that are used: if the algorithm conditionally visits the graph by branching as a function of the secret weights, then the branching patterns could leak a substantial amount of secret information. In a similar way, the resolution of combinatorial problems, even on obfuscated inputs, can leak substantial information through the structure of the combinatorial object that is manipulated, as well as through its running time. We stress that this is not just a theoretical concern: numerous techniques have been developed, notably in the line of work on side-channel attacks [10], that can successfully exploit branching patterns and running times in order to recover the secrets on which computation is performed.

1.1 Our Contributions

This paper investigates the problem of securely solving combinatorial problems in a multi-party setting through a series of examples taken from graph theory. To the best of our knowledge, this is the first time that these most classical algorithmic problems have been addressed in a general secure multi-party setting. Our solutions have applications in the numerous contexts where a graph is shared between competing entities. Natural examples include: privacy-preserving GPS guidance in which one party knows the map while the other knows his origin and destination, privacy-preserving determination of topological features in social network (the number of different ways to connect two people can be seen as a special case of the maximum flow problem, for instance, in which case each party would know his own friends but no more), or privacy-preserving determination of the performance of the cooperation between competing network operators (gas, electricity, logistics, ...), in which each party would know the capacity of his own infrastructure but no more. Furthermore, our study raises several intriguing complexity gaps and suggests the exploration of various trade-offs.

Algorithm Design. We focus our research on computing the shortest path and the maximum flow based on the secure arithmetic black-box functionality of Damgård and Nielsen [11] augmented with comparison [12]. That is, our protocols assume access to a functionality that offers secure addition, multiplication and comparison. This allows us to abstract from the specific security model in which we want our protocol to be secure: depending on the implementation of the secure arithmetic black-box that is used, our protocols will be secure only in the presence of an honest majority or with up to all but one corrupted player, in the information theoretic or computational model, in front of passive or active adversaries, ... Various such implementations, in various models, are available in tools designed for multi-party computation such as FairplayMP [13], Sepia [14], Sharemind [15] or VIFF [16].

We focus on two of the most standard graph problems, chosen for their wide diversity of applications: computing shortest paths and maximum flows. For each of these problems, we discuss secure evaluation techniques inspired from classical algorithms of various complexities: Bellman-Ford and Dijkstra for the shortest path, and Edmonds-Karp and Push-Relabel for the maximum flow.

Our resulting algorithms offer quite different overheads, depending on the algorithm and the graph structure, as illustrated in Table 1. For those algorithms, the table shows first the traditional (non secure) complexity, then the complexity of our secure versions expressed in number of calls to the arithmetic functionality. There, we consider the case of a graph with public structure and then with private structure, meaning that not only the weight of each edge is kept secret, but that the adjacency relation between vertices is kept private as well.

Several observations can already be made.

- The best implementations, using advanced data structures as dynamic trees [17] or Fibonacci heaps [18], are definitely non-trivial to replicate in the secure setting (see also discussion in Section 1.2 below). Their relevance is also unclear for the relatively small size of the problems that we are addressing, as they usually come with large constants.
- The overheads resulting from moving from the original algorithms to their secure versions largely differ between algorithms: in the case of a public structure for instance, we see either no difference, or an $|E|$ factor or a $|V|$ factor depending on the algorithm.
- The overhead resulting from hiding the graph structure largely differs depending on the algorithm and type of graph. For Bellman-Ford and Push-Relabel, the difference essentially corresponds to always handling a complete graph when the structure needs to be hidden. For Dijkstra however, the secrecy of the graph structure has no impact.
- While Bellman-Ford is traditionally less efficient than Dijkstra, this is not true anymore (asymptotically at least) for our secure variants: Bellman-Ford becomes substantially more efficient for sparse graphs (e.g., if $|E| = \mathcal{O}(|V|)$) and the asymptotic complexities are similar for dense graphs.

The overheads in terms of number of protocol participants, round complexity, ... largely depend on the implementation of the secure arithmetic functionality, and are in line with traditional works.

Table 1. Asymptotic complexities: original algorithms and secure versions with public and private graph adjacency matrix

	Optimized	Original	Public Structure	Secret Structure
Bellman-Ford	$\|V\|\|E\|$	$\|V\|\|E\|$	$\|V\|\|E\|$	$\|V\|^3$
Dijkstra	$\|E\| + \|V\| \log \|V\|$	$\|V\|^2$	$\|V\|^3$	$\|V\|^3$
Edmonds-Karp	$\|V\|^2\|E\|$	$\|V\|\|E\|^2$	$\|V\|\|E\|^2$	$\|V\|^5$
Push-Relabel	$\|V\|\|E\| \log(\frac{\|V\|^2}{\|E\|})$	$\|V\|^3$	$\|V\|^2\|E\|$	$\|V\|^4$

Complexity: The Constants Matter. In order to challenge our algorithms in practice, we implemented them all using the Virtual Ideal Functionality Framework (VIFF) of Geisler et al. [16], in the honest-but-curious model.

This allowed us to further investigate the constants hidden by the asymptotic notations discussed above. This made particularly visible the difference of cost between the different black-box primitives that we used: addition based on linear secret sharing [19] comes for free (no communication involved), multiplication is noticeable (it involves one secret sharing), and comparison (based on Toft's protocol [20]) is ≈ 165 times more expensive than a multiplication, something that strongly contrasts with the execution time of traditional algorithms.

These differences have strong practical impact and motivated some trade-offs as well.

- Our version of Dijkstra's algorithm involves only $|V|^2$ comparisons compared to $|V|^3$ (or $|V||E|$) in Bellman-Ford. As a result of this, for dense graphs or when the graph structure is secret, Dijkstra's algorithm remains considerably more efficient than Bellman-Ford's, even when the structure of the graph is public, provided that the graphs have a reasonably small size (a hundred vertices).
- For sparse public graphs that contain a small number of paths from the source to the sink, a variant of the Edmonds-Karp's algorithm that relies on an exhaustive public enumeration of the source to sink paths can be considerably simpler and more efficient than a secure version of the breadth-first search for augmenting paths that is performed in the original algorithm: this allows trading expensive book-keeping and addressing operations for more but much simpler rounds.

So, besides the fact that our work offers the first solutions for the secure evaluation of various graph properties, we think that it raises several intriguing complexity issues. Notably, we wonder whether the complexity gaps that we have are inherent to the added security or if they can be improved.

1.2 Related Works

As mentioned above, the large majority of works on secure multi-party computation focused on functions whose evaluation execution flow is independent of the secret inputs. There are important exceptions to this, however.

Branching Programs. Branching programs are decision procedures that, based on some inputs and decision parameters such as thresholds, perform a specific classification of the input. Secure versions of these programs, where a user does not learn the branching program of the server while the server does not learn the user's inputs, have been considered in various works [21], [22], [23], [7], [24]. While these works share our goals of hiding the data path through which the program is going, they do not aim at hiding the length of that path which, in our case at least, could leak a substantial amount of information.

Shortest Path In The Two-Party Setting. Brickell and Shmatikov [25] addressed the problem of solving some graph problems securely and their work is, as such, the closest to ours. Substantial differences appear, though.

First, their security model is quite different from ours. Their protocols, which are based on a privacy-preserving set union protocol, proceed by making their outputs known to the participants progressively as part of the execution (e.g., edge by edge as the protocol runs). Even though this is not revealing more than the eventual outcome, this makes their protocols unusable as sub-components of other higher-level protocols that would rely on using these outputs as part of their secret state. Revealing outputs part-by-part as the protocol runs might also be problematic in applications in which some participants could abort the protocol in the middle of its execution, based on what they have already learned. Our protocols, on the other hand, can be freely used as subroutines, and one of our secure max-flow algorithms will make use of a secure shortest-path algorithm.

Second, the graph problems they consider are different from ours as well. They do not consider the maximum flow problem at all: their work focuses on computing shortest distances, from a known source to all the vertices or for all the vertex pairs, in a setting where all the participants assign a weight to all edges. We further investigate the problem of computing the shortest path from a single source to a single destination, which cannot be done using their set union technique as it would reveal much more information than the specific distance we are interested in.

Eventually, their protocols are not based on generic building-blocks, like the arithmetic black-box functionality on which we rely. Specifically, their protocols are designed for the two-party computation setting in the honest-but-curious model. While these specifics allow them to develop techniques that are quite efficient in this two-party setting, it is unclear how efficient a transposition of their approach to the multi-party setting would be.

Efficient Secure Datastructures. The problem of computing securely on datastructures has recently been investigated by Toft [12], in the case of a secure priority queue, which he implements using a variation of bucket heap. The problem studied there shares similar flavors with those we address here: to compute securely on structured data by keeping the actions independent of the inputs. The computational overhead compared to the efficiency of the original bucket heap is logarithmic, making it occupy an interestingly different spot in the list of overhead examples discussed above.

Similar effects could also be achieved through the use of oblivious memories [26], [27].

Overview. Section 2 describes the building blocks we will use and our main implementation choices. Section 3 describes our approach to the classical single source and single-pair shortest path problems, and Section 4 describes our approaches of the maximum flow problem.

2 Preliminaries

2.1 Black-Box Operations

Arithmetic Black-Box. We build our protocols on top of an ideal functionality : the arithmetic black-box functionality $\mathcal{F}_{ABB}$ of Damgård and Nielsen [11], whose definition captures the properties we need.

This functionality allows n parties to securely store elements of a ring $\mathbb{Z}_m$, to repeatedly perform the ring operations of addition and multiplication on these elements, and to open the result of the computation when needed. Following Toft [12], we consider a slightly extended and abstracted version of this functionality that offers the possibility to perform secure comparison and consider any possible ring. So, storing, opening, addition, multiplication and comparison will be the only secure operations on which our protocols will rely. Following the tradition, we will write $[x]$ to address the version of x securely stored by $\mathcal{F}_{ABB}$, and denote the secure arithmetic operations on secret values in the natural way, e.g., $[z] \leftarrow [x] + [y]$ for the addition of two secrets. The actual protocol implementing these operations depends on the details of the realization of this functionality. Numerous MPC schemes can be used for that purpose [4,3,11] or, for more recent approaches [28,29,30], depending on the security model that is appropriate.

Graph Representation. Depending on the algorithm we are trying to compute and on the part of the graph description that is part of the secret input, different graph representation approaches will show to be useful.

In all cases, we will assume that the number of vertices in the graph is public (or at least an upper bound on it). Depending on the setting, the adjacency relationship between the vertices might be public or not. For instance, it is natural to have it public if the graph represents the connections between places on a map, but it might be desirable to keep it secret if the presence of edges reveals the existence of transactions between competing companies.

A traditional structure for storing a graph consists in storing, with every vertex, a list of its neighbors (and the weight of the corresponding edges). This structure is quite efficient in terms of memory. However, it might be quite problematic from a security point of view, as it discloses the degree of each vertex. A solution would be to tolerate the leakage of an upper bound on these degrees, but that upper bound would be close to imply the storage of a complete graph as soon as one single vertex is of high degree. Furthermore, even if the leakage of the degree of the vertices is tolerated, algorithms that perform breadth-first search on vertices and branch depending on the weight of edges could reveal a lot of information. As a result, this graph representation can show to be very effective in some cases, but completely inappropriate in others, even when the graph structure is public.

A second traditional way of representing graphs is to store their adjacency matrix, the elements of the matrix representing the weight of the edges between vertices. This approach has the benefit of offering a storage that is independent of the graph structure. While running our algorithms, we will often need to

perform some operations on a specific vertex designated by a secret index. This will typically be performed by running that operation on all vertices, including a canceling factor everywhere but on the vertex that needs to be treated. An obvious way of testing whether we are working on the right vertex would be to perform a test at each step. We actually use a more effective approach by representing the index of the vertex i by a vector $[\mathbf{i}] \in \{[0], [1]\}^{1\times n}$ where each entry is $[0]$ except for the i'th which is $[1]$. We can then access the weight of the edge from vertex i to vertex j by computing the matrix product $[\mathbf{i}].[\mathbf{M}].[\mathbf{j}]^t$. Protocol 1 provides a way to update an element in a shared list and it can be easily extended to update an element in a shared matrix. This protocol also exemplifies a common way of emulating a branching depending on a secret value: the arithmetic operation in the loop is actually equivalent to computing **if** $[\mathbf{i}]_j = [1]$ **then** $[\mathbf{l}]_j = [x]$.

Protocol 1: Update an element in a shared list and at a shared position

Input: A list $[\mathbf{l}]$ of length n, a shared index $[\mathbf{i}]$, a shared value $[x]$.
Output: The list $[\mathbf{l}]$ with the update $[\mathbf{l}]_i = [x]$.

```
1 for j ← 1 to n do
2 |  [l]_j ← [l]_j + [i]_j · ([x] − [l]_j);
3 end
```

For a graph with n vertices, this protocol allows retrieving a secret position in the adjacency matrix in $\mathcal{O}(n^2)$ multiplications instead of $\mathcal{O}(n^2)$ comparisons, which is considerably more efficient, even if it implies a considerable overhead in storage (moving from 1 secret index to n secret bits). We note that, in all cases, this approach implies treating the graph as if it were complete, which can be a considerable waste of resources if the graph is actually sparse.

3 Privacy-Preserving Shortest Path Problem

The single-source shortest path problem is a major problem in graph theory. It has several immediate applications. The typical one is finding the shortest way to connect two cities on a road map where each city is represented by a vertex and each road between two cities by an edge. The edge weights are the road distances between cities. In this context, a user may then want to obtain driving directions without revealing neither his starting point nor his destination. Another application is the one of two entities owning each a secret location in a shared network and willing to compute the distance between them without disclosing their location. We note that such a problem is worth solving even for relatively small graphs. Consider for instance a routing network with a dozen hubs in different European countries and three competing logistic companies having each their own transportation costs for a defined set of roads. As costs typically represent sensitive information that should not be disclosed to competitors, being able to solve the shortest path problem securely for 3 parties and a graph with a dozen of vertices is quite helpful. Similar problems happen for network traffic on routers where a small number of big hubs is involved. Competing companies have to solve the shortest

path to define routing scheme without revealing sensitive information about internal network configuration.

Shortest path algorithms are also used as subalgorithms for more advanced problems like the Chinese postman problem or the max-flow problem that we address below: this highlights the importance of keeping our protocols composable.

We investigate two standard algorithms for finding the single-source shortest path in a graph with weighted edges: Dijkstra's algorithm and Bellman-Ford's algorithm. The first one requires all edge weights to be positive, while the second one only assumes there is no negative-weight cycle in the input graph. As the non-secure version of all the algorithms that we treat is widely available [31], we will only briefly outline them.

All our protocols assume that inputs are already stored in the $\mathcal{F}_{ABB}$ functionality and give access to the stored outputs (that can be opened through opening requests to $\mathcal{F}_{ABB}$). This feature guarantees the composability of the protocols. The way inputs and outputs are shared depends on the application: they might come from a specific problem, or from the needs of a higher-level protocol using this protocol as a sub-routine, for instance.

3.1 Bellman-Ford's Algorithm

The algorithm of Bellman-Ford is particularly simple, making it a natural target for building a secure version. This algorithm proceeds by repeatedly scanning all edges, in search of adding edges that decrease the ongoing distance from the source to the various vertices. If a pass over the edges did not improve the current solution, or if the edges were scanned $|V|$ times, the algorithm halts. An interesting feature of this algorithm is that its flow of operations only depends on the structure of the graph but not on the weight of the edges. Its drawback is its time-complexity: its classical implementation runs in $\mathcal{O}(|V||E|)$ time.

Protocol 2 (the SSP1 protocol) presents our secure shortest path protocol based on Bellman-Ford. Note that $h(e)$ and $t(e)$ represent the head and tail vertex of an edge e respectively. Also, note that $\top$ is a number agreed in advance by the players as a higher bound for some calculations of the protocol. We refer to Appendix A for discussion of the values of $\top$ and m in all our protocols. Finally, note that `updatevector` refers to Protocol 1. The SSP1 protocol differs from the original algorithm only in a limited number of aspects: *a*) the branching corresponding to the discovery of a shorter path is handled on Lines 8–10 through arithmetic operations as in Protocol 1, *b*) the early termination condition of the Bellman-Ford algorithm, which is triggered if the inner loop happens to have no effect during one pass, is removed as it could leak information. This does not invalidate the correctness of the algorithm but only increases the running time.

The structure of this algorithm makes it easy to implement with either of the two graph representations discussed above (list or matrix), making it possible to fully exploit the sparsity of graphs when it is public (we use the matrix representation if it has to be kept secret).

Protocol 2: SSP1 protocol based on Bellman-Ford's algorithm

Input: A graph $G = (V, E)$ where V is the list of vertices and E the list of edges, a set of shared weights $[\mathbf{w}]_e$ for each $e \in E$, and a share of the source vertex $[s] \in V$.
Output: The list of immediate predecessors $\mathbf{p}$ and/or total distances $\mathbf{d}$.

for $i \leftarrow 1$ **to** $|V|$ **do**
  $\mathbf{p}_i \leftarrow [0]$; $\mathbf{d}_i \leftarrow [\top]$;
end
`updatevector`$([\mathbf{d}], [\mathbf{s}], [0])$;
for $i \leftarrow 1$ **to** $|V|$ **do**
  for $e \leftarrow 1$ **to** $|E|$ **do**
    $[y] \leftarrow [\mathbf{d}]_{t(e)} - [\mathbf{d}]_{h(e)} + [\mathbf{w}]_e$;
    $[x] \leftarrow [y] < 0$;
    $[\mathbf{d}]_{h(e)} \leftarrow [\mathbf{d}]_{h(e)} + [x] \cdot [y]$;
    $[\mathbf{p}]_{h(e)} \leftarrow ([1] - [x]) \cdot [\mathbf{p}]_{h(e)} + [x] \cdot t(e)$;
  end
end
If there was an update during the very last pass, solution is unbounded ($\exists$ negative cycle). Open required output.

It can be seen that our implementation requires $|V||E|$ secure comparisons, dominating the time required to perform $2|V||E|$ secure multiplications and $5|V||E|$ additions. These complexities grow to $\mathcal{O}(|V|^3)$ when the graph structure is secret, as the graph is then treated as complete (i.e., augmented with edges of infinite weight). Very interestingly, this algorithm is the only one among those we treated in which our solution does not raise any asymptotic overhead (when the structure is public).

Security. The simulation of an execution of this protocol is immediate from the simulators available for the different calls that can be made by the $\mathcal{F}_{ABB}$ functionality: the simulators corresponding to each of the '$+$', '$\cdot$' and '$<$' operations can be invoked in turn, in an order defined by the protocol execution, and a number of times that only depends on public values ($|V|$ and $|E|$). The same argument will apply to the other protocols we present in this paper, and we will therefore not come back to it.

3.2 Dijkstra's Algorithm

Dijkstra's algorithm computes the shortest path from the source to all vertices in the graph, that is, the shortest path tree rooted at the source. The algorithm is greedy. At each iteration one vertex (the one with the smallest distance label) is permanently updated to the status *scanned*.
Adapting Dijkstra. The fact that Dijkstra's algorithm goes through the graph in an order that depends on the weight of the edges makes it very difficult to efficiently exploit the sparsity of a graph: our best solutions have all a complexity that amounts to the one of a complete graph, and we therefore use the matrix representation in all cases for our protocol.

Protocol 3: SSP2 protocol based on Dijkstra's algorithm

Input: A graph $G = (V, E)$ where V is the list of vertices and E the list of edges, a matrix of shared weights $[\mathbf{M}]_{i,j}$ for $i, j \in \{1, ..., |V|\}$ and a source vertex $[\mathbf{s}] \in V$.

Output: The vector of distances $\mathbf{d}_i$ and the matrix of predecessor $[\mathbf{P}]_{i,j}$ for $i, j \in \{1, ..., |V|\}$.

```
1  for i, j ← 1 to |V| do
2  |  [P]_{i,j} ← [0]; [d]_i ← [⊤]; [q]_i ← [0];
3  end
4  updatevector([d], [s], [0]);
5  for i ← 1 to |V| do
6  |  [d'] ← [d] + [q];
7  |  [min], [k] ← binarymin([d']);
8  |  updatevector([q], [k], [⊤]);
9  |  for j ← 1 to |V| do
10 |  |  [a] ← ([d] + [M]_{*,j}) · [k];
11 |  |  [c] ← [a] < [d]_j;
12 |  |  [P] ← updaterow([P], j, [P]_j + [c] · ([k] − [P]_j));
13 |  |  [d]_j ← [d]_j + [c] · ([a] − [d]_j);
14 |  end
15 end
16 return [d], [P];
```

Protocol 3 (the SSP2 protocol) presents our secure shortest path protocol based on Dijkstra. Note that `updatevector` refers to Protocol 1 and that `updaterow` is the natural extension of `updatevector` for replacing a complete row in a shared matrix. Protocol `binarymin` has been introduced by Toft in [32] to obtain the minimal value out of a vector of shared values. It securely computes a share of the minimal value, [min], along with a share of its index, $[\mathbf{k}]$. The protocol uses $\mathcal{O}(n)$ comparisons and multiplications. Its overall round complexity is $\mathcal{O}(\log(n))$ rounds. Vector $\mathbf{q}$ records the status of each vertex. An entry is equal to zero if the corresponding vertex has not been scanned yet. It is updated to $\top$ as soon as the vertex has been scanned.

The main differences between the traditional and our secure version of Dijkstra's algorithm happen in the inner loop: *a*) On Line 9, the loop goes through all vertices instead of only considering the neighbors of the current vertex. In particular, this includes an always transparent step where we consider the current vertex and gives a substantial overhead if a public sparse graph is considered. *b*) On Lines 4 – 8 – 12, we need to go through all elements of a row or a vector, even if we know that only one of them is going to be updated. Those two modifications contribute to the same effect: they increase the original complexity of Dijkstra from $\mathcal{O}(|V|^2)$ to $\mathcal{O}(|V|^3)$. More precisely, the exact number of comparisons is $2|V|^2 - 3|V| + 1$ and the exact number of dot products (used for the multiplication of vectors, costing $|V|$ multiplications) is $2|V|^2 - |V|$ for $|V| \geq 4$.

As the comparison protocol we use requires 165 multiplications to compute a comparison, the number of multiplications to compute the shortest path in a complete tree is around $2|V|^3 + 329|V|^2 - 495|V| + 165$.

The switch from quadratic to cubic dominance is at around 165 vertices which is precisely the number of multiplications used by a single comparison.

Our secure version of Dijkstra comes with an overhead of a factor $|V|$ compared to the original one, even when the graph structure can be considered as public. We note that this was not the case in the work of Brickell [25] who considered running Dijkstra securely as well, but accepted to output the shortest paths step by step. Besides the limitation that this brings when the protocol has to be composed, we also observe that our algorithm can be used to solve problems that could not be solved by Brickell's approach, namely, computing the shortest path between two specific vertices without leaking any other information: their approach indeed leaks the shortest path to *all* vertices.

3.3 Implementation Prototype

We implemented our protocols over the Virtual Ideal Functionality Framework to challenge their performance. We considered a 3-party execution in the information theoretic model with passive security: secret values are shared using Shamir's secret sharing, the BGW protocol is used for multiplication [2], and Toft's protocol is used for comparison [20]. These choices were made for simplicity and ease of prototyping, though much more efficient protocols exist and would have led to considerably shorter running times [28,30]. The computation was performed on a single workstation equipped with an Intel Xeon CPUs X5550 (2.67GHz) and 24GB of memory, running a standard Debian Squeeze.

We ran the two shortest path protocols described above on complete graphs of various sizes. This first showed that Protocol 2 can only be conveniently used for graphs where $|V||E| \approx 10^3$ (a few minutes on a standard laptop): see Table 2.

Table 2. Execution times of Protocols 2 and 3 for a complete shortest path tree

Number of vertices		4	8	16	32	64	128
Execution times (in seconds)	SSP1	9	63	501	4003	31951	-
	SSP2	9	13	50	217	1018	5622

Our secure versions of Bellman-Ford and Dijkstra have approximately the same complexity for complete graphs. However the quadratic number of comparisons makes it possible to run our secure version of Dijkstra on a 64-vertex complete graph in roughly twice the time as taken by Bellman-Ford on a 16-vertex graph, and we have been able to run it up to a 128-vertex complete graph (i.e., counting 16256 directed edges) in a bit more than an hour.

While these timings might look fairly high, they still make it possible to solve natural problems in a reasonable time. For instance, if the 3-party, 12-vertex problem outlined above could be solved in around 30 seconds.

4 Maximum Flow

In an oriented graph where the edges have a constraint of *capacity*, the maximum flow problem consists in finding the maximum number of units that can be carried from a vertex called *source* to another vertex called *sink*. The *flow* through an edge designates the number of units passing by it. This number cannot exceed the capacity.

This problem has numerous classical applications. In the spirit of our previous examples, one of them could be competing transport companies willing to determine the capacity they could reach if they decided to realize a joint-venture. It is natural in such a context to expect that these companies will not be willing to disclose their full network structure to each other. As in the case of the shortest path, algorithms solving the maximum flow problem are also very useful as subroutines for solving other problems. The minimum cut problem is one such traditional example, which can be solved using $\mathcal{O}(|V|)$ invocations of the maximum flow algorithm. Solving this problem is then useful to determine where the weak points of the joint network would be.

Although we investigated many different algorithm for a transportation in MPC, this paper only presents two secure protocols based on the Edmonds-Karp's and the Push-Relabel algorithms.

4.1 Edmonds-Karp's Algorithm

The basic idea of the Edmonds-Karp's (EK) algorithm is to find an augmenting path in the residual graph that is the graph in which the edges are weighted by their residual capacity, i.e., the capacity minus the current flow. Each augmenting path increases the total flow so that the algorithm eventually terminates when there are no augmenting paths left. The increase is monotonic and paths are considered once only. Typically, the EK algorithm uses a breadth-first search to find the next augmenting path.

The asymptotic complexity of the traditional EK algorithm is $\mathcal{O}(|V||E|^2)$, which we can match securely (see Appendix B.) As we have seen in the case of the shortest path problem, this complexity will be prohibitive even for very small graphs if they are complete. It therefore makes sense to focus our attention on (oriented) strongly sparse graphs, of which we consider the structure to be public. More precisely, we consider graphs in which the number of paths from the source to the sink is fairly small, e.g., bounded by a small polynomial in the number of vertices.

The algorithm is given on input a list containing all the paths sorted in a growing order of length, $\mathbf{p} = (p_1, ..., p_k)$ where k is the number of paths in the graph. This list is not secret as the structure is not, and can therefore be easily constructed in public. Our protocol based on Edmonds-Karp (the SMF1 protocol) is presented in Protocol 4.

The main differences between this protocol and Edmonds-Karp's approach are: *a*) the public enumeration of all the paths instead of building of a breadth-first search for capacity augmenting paths, and *b*) the treatment of all the paths as if they were augmenting.

Protocol 4: SMF1 maximum flow protocol based on Edmonds-Karp's algorithm

Input: A graph $G = (V, E)$ where V is the list of vertices and E the list of edges, a source vertex $so \in V$, a sink vertex $si \in V$, and a list $\mathbf{p}$ of length k containing the paths between so and si sorted in a growing order of length. A set of capacities $[\mathbf{c}]_e$ and a set of flows $[\mathbf{f}]_e$ initially set to $[0]$ for $e \in E$. $\bar{e}$ is the edge opposite to e.

Output: The maximum flow value from so to si.

while $|\mathbf{p}| > 0$ **do**
  $p \leftarrow \mathbf{pop}(\mathbf{p})$;
  $[r], [\mathbf{i}] \leftarrow \mathtt{binarymin}_{e \in p}([\mathbf{c}]_e - [\mathbf{f}]_e)$;
  $[b] \leftarrow [r] > 0$;
  $[a] \leftarrow [b] \cdot [r]$;
  for $e \in p$ **do**
    $[\mathbf{f}]_e \leftarrow [\mathbf{f}]_e + [a]$;
    $[\mathbf{f}]_{\bar{e}} \leftarrow [\mathbf{f}]_{\bar{e}} - [a]$;
  end
end
return $\sum_{e \in S} [\mathbf{f}]_e$ where $S = \{e \in E | h(e) = so\}$;

The SMF1 protocol is correct as the set of all the augmenting paths is contained in the set of all the paths $\mathbf{p}$. Moreover, it ensures the confidentiality of the edge capacities as no information is leaked about which path of $\mathbf{p}$ is augmenting and which is not.

It is easy to see that the SMF1 Protocol requires $\mathcal{O}(k|V|)$ comparisons, as the length of the longest path in the graph is bounded by $|V| - 1$, and $\mathcal{O}(k)$ multiplications. This protocol makes a crucial use of the existence of a small number of paths in the graph, something that we were not able to use in the SSP2 protocol for instance. It is however highly inefficient for dense graph and would have a factorial complexity for complete graphs.

This protocol applies well to our previous example of the three competing logistic companies trying to determine the max flow in their joint networks. If we consider a case with 10 vertices and 37 different paths, the execution takes less than a minute as shown in Table 3.

Table 3. Execution times of Protocol 4 for 10-vertex graphs

Number of paths	2	4	8	14	37	86	135
Number of edges	22	21	25	25	32	30	30
Execution times (in seconds)	3	6	9	18	40	94	148

4.2 Push-Relabel Privacy-Preserving Implementation

The Push-Relabel algorithm, also called relabel-to-front when implemented with a FIFO list, introduces two additional attributes for the vertices, the height and

the excess. An edge is called admissible if it goes from a higher to a lower vertex. The algorithm alternatively pushes the excess along admissible edges and increases the height of the vertices until all excess has been pushed to the sink or back to the source.

The basic operation of the algorithm is Push/Relabel applied to a given vertex. This operation pushes all the excess through incident admissible edges (updating the excesses of incident vertices accordingly). Finally, in case not all the excess has been pushed, the elevation of the vertex is minimally increased so as to create at least one more admissible edge, and Push/Relabel terminates.

Throughout the algorithm a list L with vertices with positive excess (except the source and the sink) is maintained. At each iteration, one vertex of L is selected and Push/Relabel is applied. The algorithm terminates when the list is empty. In the FIFO implementation, the next vertex of L to be treated is selected in the FIFO order. This FIFO Push/Relabel algorithm terminates in $\mathcal{O}(|V|^3)$ operations.

Our protocol based on Push-Relabel is presented in Protocol 5. The main differences between this protocol and the traditional Push/Relabel algorithm are as follows : *a*) when Push/Relabel is applied to a vertex with zero excess, no update of the elevation is performed at the end, *b*) in each phase, treat *all* vertices except the source and the sink, in a fixed order agreed between the players, and *c*) during each Push/Relabel operation applied to a vertex i, the order in which the edges (i, j) are considered is fixed and agreed in advance between the players. It is clear that these changes do not modify the correctness of the original algorithm.

Protocol 5: A phase of the SMF2 protocol based on Push/Relabel

Input: A complete graph $G = (V, E)$ where V is the list of vertices and E the list of edges. A vertex i to be treated, a vector of elevations $[\mathbf{h}]$, a matrix of residual capacities $[\mathbf{R}]$ and a vector of excesses $[\mathbf{e}]$.

Output: Update of the elevations $[\mathbf{h}]$, the residual capacities $[\mathbf{r}]$ and the excesses $[\mathbf{e}]$.

$[\delta] \leftarrow 2|V|$; **for** $j \leftarrow 1$ **to** $|E|$ **do**
  $[\alpha] \leftarrow [\mathbf{h}]_i > [\mathbf{h}]_j$;
  $[x] \leftarrow \min([\mathbf{e}]_i, [\mathbf{R}]_{i,j})$;
  $[y] \leftarrow [\alpha].[x]$;
  $[\mathbf{R}]_{i,j} \leftarrow [\mathbf{R}]_{i,j} - [y]$; $[\mathbf{R}]_{j,i} \leftarrow [\mathbf{R}]_{j,i} + [y]$;
  $[\mathbf{e}]_i \leftarrow [\mathbf{e}]_i - [y]$; $[\mathbf{e}]_j \leftarrow [\mathbf{e}]_j + [y]$;
  $[\delta] \leftarrow \min([\delta], [\mathbf{h}]_j + 2|V| \cdot [\alpha])$;
end
$[\alpha] \leftarrow [\mathbf{e}]_i > 0$;
$[\mathbf{h}]_i \leftarrow [\mathbf{h}]_i \cdot (1 - [\alpha]) + ([\delta] + 1) \cdot [\alpha]$;

Moreover, it can be verified that the relabel-to-front algorithm terminates in maximum $4|V|^2 - 10|V| + 12$ complete phases. Therefore we obtain an "all-cases" complexity of $\mathcal{O}(|V|^2|E|)$, both in comparisons and multiplications. Note that this does not match the FIFO complexity, because we scan all edges at each pass, even when the excess of the tail vertex is zero.

The complexity of this algorithm remains lower than the one of the original Edmonds-Karp and it is asymptotically better than the optimized version of Edmonds-Karp presented in Section 4.1 for graphs with vertices of high degree. However, the running time of Protocol SMF2 remains very high. Experiments showed that the use of a traditional halting criterion at the end of each SMF2 phase (i.e. nothing has been pushed) results in dramatic running time improvements. However it also demonstrated a huge variability (the algorithm may halt after a single phase), which suggests that a substantial amount of information could be derived from it. Quantifying this information is left for future work, and its impact is likely to depend on the application.

5 Conclusion

We proposed two protocols for securely computing shortest paths as well as two protocols for securely computing maximum flows in graphs. Besides the interest that these protocols have in the numerous contexts in which their insecure counterparts have found applications in the past (possibly relying on a trusted third party), our investigation raised interesting complexity gaps between centralized algorithms and secure protocols, ranging from a constant to something growing like the number of vertices in the graphs. It is then natural to wonder whether these gaps, when they arise, can be decreased. Various avenues appear for that purpose:

- Design efficient datastructures adapted to the investigated problems. For instance, the recent work of Toft [12] on priority queues could lead to considerably more efficient versions of our secure shortest-path protocols. In particular, whether data structures similar to dynamic trees or Fibonacci heaps are implementable in a secured setting without revealing the execution flow remains an open question.
- Investigate whether secure comparisons, which often are a bottleneck, can be traded for other, cheaper, arithmetic operations. This raises unusual questions from a traditional algorithmic point of view, as comparisons are usually considered as basic operations.

Considering other standard combinatorial problems could also provide new insights. The protocols and results presented in the paper are prototypes that validate the theoretical complexity evaluations. While the running times given for the protocols look unpractical for large graphs, this issue must be put in perspective. Indeed, an implementation for concrete applications should definitively be improved by relying on lower level programming languages and optimized underlying libraries. Various optimization techniques (see, e.g., [28,30]) would lead to performance increases of several orders of magnitude, as was observed in the case of the AES during the last 3 years for instance (see [29] and the references within).

Acknowledgements. This research was supported by the WIST Walloon Region project CAMUS. Edouard Cuvelier and Sophie Mawet are funded by a FRIA grant of the F.R.S.-FNRS. Mathieu Van Vyve is supported by the Belgian IAP

Program initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors. The authors are grateful to Claudio Orlandi and the anonymous reviewers for their constructive feedback. They also sincerely thank Sylvie Baudine for her help in improving the paper.

References

1. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. IEEE (1982)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10. ACM (1988)
3. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC, pp. 11–19. ACM (1988)
4. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229. ACM (1987)
5. Bogetoft, P., Damgård, I., Jakobsen, T.P., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006)
6. Cramer, R., Franklin, M.K., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
7. Barni, M., Failla, P., Kolesnikov, V., Lazzeretti, R., Sadeghi, A.-R., Schneider, T.: Secure evaluation of private linear branching programs with medical applications. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 424–439. Springer, Heidelberg (2009)
8. Sadeghi, A.-R., Schneider, T., Wehrenberg, I.: Efficient privacy-preserving face recognition. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 229–244. Springer, Heidelberg (2010)
9. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
10. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
11. Damgård, I., Nielsen, J.B.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
12. Toft, T.: Secure data structures based on multi-party computation. In: PODC, pp. 291–292. ACM (2011)
13. Ben-David, A., Nisan, N., Pinkas, B.: Fairplaymp: a system for secure multi-party computation. In: CCS, pp. 257–266. ACM (2008)
14. Burkhart, M., Strasser, M., Many, D., Dimitropoulos, X.: Sepia: privacy-preserving aggregation of multi-domain network events and statistics. In: Proceedings of the 19th USENIX Conference on Security, USENIX Security 2010. USENIX (2010)
15. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A Framework for Fast Privacy-Preserving Computations. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 192–206. Springer, Heidelberg (2008)

16. Geisler, M.: Cryptographic protocols: theory and implementation. PhD thesis, Aarhus University Denmark, Department of Computer Science (2010)
17. Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. J. Comput. Syst. Sci. 26(3), 362–391 (1983)
18. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34(3), 596–615 (1987)
19. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
20. Toft, T.: Primitives and Applications for Multi-party Computation. PhD thesis, Department of Computer Science, Aarhus University (2007)
21. Kruger, L., Jha, S., Goh, E.J., Boneh, D.: Secure function evaluation with ordered binary decision diagrams. In: ACM CCS, pp. 410–420. ACM (2006)
22. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
23. Brickell, J., Porter, D.E., Shmatikov, V., Witchel, E.: Privacy-preserving remote diagnostics. In: ACM CCS, CCS 2007, pp. 498–507. ACM (2007)
24. Barni, M., Failla, P., Lazzeretti, R., Sadeghi, A.R., Schneider, T.: Privacy-preserving ECG classification with branching programs and neural networks. IEEE TIFS 6(2), 452–468 (2011)
25. Brickell, J., Shmatikov, V.: Privacy-preserving graph algorithms in the semi-honest model. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 236–252. Springer, Heidelberg (2005)
26. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. J. ACM 43(3), 431–473 (1996)
27. Damgård, I., Meldgaard, S., Nielsen, J.B.: Perfectly secure oblivious RAM without random oracles. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 144–163. Springer, Heidelberg (2011)
28. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 169–188. Springer, Heidelberg (2011)
29. Damgård, I., Keller, M., Larraia, E., Miles, C., Smart, N.P.: Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 241–263. Springer, Heidelberg (2012)
30. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012)
31. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press (2009)
32. Toft, T.: Solving linear programs using multiparty computation. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 90–107. Springer, Heidelberg (2009)

A Bounds

The size of the ring $\mathbb{Z}_m$ has to be chosen carefully to prevent overflows. For each protocol presented in this paper, we provide the bounds of m and the value of $\top$ in Figure 1. These bounds depend on numbers such as the maximum weight $\mathbf{w}$ or the maximum capacity $\mathbf{c}$ allowed for the edges. These maxima are agreed in advance by the players. Remark that $\top$ is smaller than m. Most comparison protocols require a much larger m than the values to compare. This dependence is taken into account via a function f.

Protocol	$\top >$	$m >$
SSP1	$\lvert V\rvert \cdot \mathbf{w}$	$f(\top)$
SSP2	$\lvert V\rvert \cdot \mathbf{w}$	$\lvert V\rvert \cdot f(\top)$

Protocol	$\top >$	$m >$
SMF1	-	$\lvert V\rvert \cdot f(\mathbf{c})$
SMF2	-	$\max(2\lvert V\rvert, \lvert V\rvert \cdot f(\mathbf{c}))$
SMF3	$\mathbf{c}$	$\max(f(\top + \mathbf{c}), \lvert V\rvert \cdot f(\mathbf{c}))$

Fig. 1. Minimal bounds on $\top$ and m to avoid overflows

B Protocols

Protocol 6 presents the complete Edmonds-Karp's implementation of the maximum flow problem in MPC. The `binarymin` function refers to the function introduced by Toft in [32] to securely compute the minimum between different values. The `Bellman-Ford` function is a natural adaptation of Protocol 2 that outputs the shortest path from the source to the sink in the form of a vector of $\{[0], [1]\}$ where a $[1]$ at the i-th position indicates that edge i belongs to the augmenting path. Note that the first augmenting path $\mathbf{p}$ is public and given in input. We refer to Figure 1 for the value of $\top$.

Protocol 6: SMF3 maximum flow protocol based on complete Edmonds-Karp's algorithm

Input: A graph $G = (V, E)$ where V is the list of vertices and E the list of edges, a source vertex $so \in V$, a sink vertex $si \in V$. A list of positive capacities $[\mathbf{c}]_i$ for each edge. The first augmenting path $\mathbf{p}$.
Output: The maximum flow value from so to si.

1 **for** $i \leftarrow 1$ **to** $|E|$ **do**
2 $\quad [\mathbf{f}]_i \leftarrow [0]$; $[\mathbf{w}]_i \leftarrow [1]$;
3 **end**
4 **for** $i \leftarrow 1$ **to** $|E|$ **do**
5 $\quad$ **for** $j \leftarrow 1$ **to** $|E|$ **do**
6 $\quad\quad [c]_j \leftarrow (1 - [\mathbf{p}]_j) \cdot \top + [\mathbf{c}]_j$; $[f]_j \leftarrow [\mathbf{p}]_j \cdot [\mathbf{f}]_j$;
7 $\quad$ **end**
8 $\quad [r], [\mathbf{k}] \leftarrow \texttt{binarymin}_{j \in \{1,\ldots,|E|\}}([c]_j - [f]_j)$; $[b] \leftarrow [r] > 0$; $[a] \leftarrow [b] \cdot [r]$;
9 $\quad$ **for** $j \leftarrow 1$ **to** $|E|$ **do**
10 $\quad\quad [\mathbf{f}]_j \leftarrow [\mathbf{f}]_j + [\mathbf{p}]_j \cdot [a]$; $[\mathbf{f}]_{\bar{j}} \leftarrow [\mathbf{f}]_{\bar{j}} - [\mathbf{p}]_j \cdot [a]$; $[cond] \leftarrow [\mathbf{c}]_j - [\mathbf{f}]_j = 0$;
$\quad\quad\quad [\mathbf{w}]_j \leftarrow [cond] \cdot |V| + (1 - [cond]) \cdot [\mathbf{w}]_j$;
11 $\quad$ **end**
12 $\quad [\mathbf{p}], [\mathbf{d}] \leftarrow \texttt{SSP1}(G, [\mathbf{w}], so, si)$;
13 **end**
14 **return** $\sum_{e \in S} [\mathbf{f}]_e$ where $S = \{e \in E | h(e) = so\}$;

The main differences between protocol and the traditional Edmonds-Karp algorithm are as follows :

- Each iteration goes through all the edges but only those which form the current path are updated.
- The SSP1 protocol is used instead of the Breath-First-Search protocol to find the smallest augmenting path because there is a serious overhead in a straightforward secure implementation of the BFS. To run the SSP1 protocol,

SMF3 maintains a list of shared weights [**w**] for the edges where the weight of an edge is [1] when it remains in the residual graph and it is $[|V|]$ otherwise.

It is straightforward to see that the asymptotic complexity of the algorithm is $\mathcal{O}(|V||E|^2)$ as the original algorithm. The number of comparisons is $|V||E|^2 + |E|^2 + |E|$ and the number of multiplications is $|V||E|^2 + 5|E|^2 + |E|$.

Parallel and Dynamic Searchable Symmetric Encryption

Seny Kamara[1] and Charalampos Papamanthou[2]

[1] Microsoft Research
senyk@microsoft.com
[2] UC Berkeley
cpap@cs.berkeley.edu

Abstract. Searchable symmetric encryption (SSE) enables a client to outsource a collection of encrypted documents in the cloud and retain the ability to perform keyword searches without revealing information about the contents of the documents and queries. Although efficient SSE constructions are known, previous solutions are highly sequential. This is mainly due to the fact that, currently, the only method for achieving sub-linear time search is the inverted index approach (Curtmola, Garay, Kamara and Ostrovsky, *CCS '06*) which requires the search algorithm to access a sequence of memory locations, each of which is unpredictable and stored at the previous location in the sequence. Motivated by advances in multi-core architectures, we present a new method for constructing sub-linear SSE schemes. Our approach is highly parallelizable and dynamic. With roughly a logarithmic number of cores in place, searches for a keyword w in our scheme execute in $o(r)$ parallel time, where r is the number of documents containing keyword w (with more cores, this bound can go down to $O(\log n)$, i.e., independent of the result size r). Such time complexity outperforms the optimal $\Theta(r)$ sequential search time—a similar bound holds for the updates. Our scheme also achieves the following important properties: (a) it enjoys a strong notion of security, namely security against adaptive chosen-keyword attacks; (b) compared to existing sub-linear dynamic SSE schemes (e.g., Kamara, Papamanthou, Roeder, *CCS '12*), updates in our scheme do not leak any information, apart from information that can be inferred from previous search tokens; (c) it can be implemented efficiently in external memory (with logarithmic I/O overhead). Our technique is simple and uses a red-black tree data structure; its security is proven in the random oracle model.

Keywords: Searchable encryption, parallel search, cloud storage.

1 Introduction

Cloud storage promises high data availability, easy access to data, and reduced infrastructure costs by storing data with remote third-party providers. But availability is often not enough, as clients need privacy guarantees for many kinds of sensitive data that is outsourced to untrusted providers.

The standard approach to achieving privacy in storage systems is to encrypt data using symmetric encryption. Storage systems based on this approach provide end-to-end privacy in the sense that data is protected as soon as it leaves the client's possession.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 258–274, 2013.

While such a solution provides strong security guarantees, it induces a high cost in terms of functionality and is therefore inadequate for storage systems that handle data at large scales. This is because after the data leaves the client's machine in encrypted form, the server cannot perform any meaningful computation on it.

To address this, one can either use general-purpose solutions (e.g., fully-homomorphic encryption [7] or oblivious RAMs [9]) or special-purpose solutions (e.g., searchable encryption). Although general-purpose solutions have advantages, including generality and stronger security properties, they are mostly of theoretical interest (e.g., recent work [19] has shown that ORAM can be relatively practical). On the other hand, special-purpose solutions like searchable encryption are practical and aim to provide a reasonable trade-off between efficiency, functionality and security.

Using a symmetric searchable encryption (SSE) scheme, a client can store a collection of encrypted documents remotely while retaining the ability to perform keyword searches without revealing any information about the contents of either the documents or the queries. There are two high-level approaches to designing reasonably efficient and secure SSE schemes. The first approach, proposed by Goh [8] and used in [3], associates to each document an encrypted data structure that can be tested for the occurrence of a given keyword. This approach naturally results in schemes with search time that is linear in n, where n the number of documents in the collection. The second approach, introduced by Curtmola et al. [6], associates an encrypted inverted index to the entire document collection. This approach yields very efficient schemes since search time is $O(r)$, where r is the number of files that contain the keyword. Note that, $O(r)$ is not only sub-linear, it is optimal. Due to its efficiency, the inverted index approach has been used in many subsequent works, including [4,11,13,20].

While the inverted index approach yields the most efficient SSE schemes to date, it has at least two important limitations. The first is that it is not well-suited to handle dynamic collections (i.e., document collections that must be updated). Although Kamara et al. [11] recently showed how to construct an encrypted inverted index that handles dynamic data, their construction is very complex and difficult to implement. In addition, the update operations reveal a non-trivial amount of information. The second limitation of the inverted index approach is that it is inherently sequential, requiring $\Omega(r)$ time even in a parallel model of computation. This is mainly because the encrypted indexes used by these constructions store data at random disk locations (for security and space efficiency), and because the associated search algorithms are adaptive in the sense that they find the next memory location to access at the currently accessed memory location. In addition, we note that even in the sequential setting, where the $O(r)$ time bound for search is optimal, these constructions can still introduce significant latency when searching for a very frequent keyword whose output contains thousands of documents.

Our Contributions. We introduce a new approach for designing SSE schemes that yields constructions with sub-linear search time but that has none of the limitations of the inverted index approach. In particular, our approach is simple, highly parallel and can easily handle updates. More precisely, for n documents indexed over m keywords and with p cores (processors) available, our construction has the following properties:

Table 1. Comparison of several SSE schemes, in terms of worst case parallel search time per keyword w. With n we denote the size of the documents collection, with r the number of documents containing keyword w, with m the size of the keywords space and with p the number of cores.

scheme	dynamism	security	search time	index size
Song et al. [18]	static	CPA	$O(\frac{n}{p})$	N/A
Goh [8]	dynamic	CKA1	$O(\frac{n}{p})$	$O(n)$
Chang and Mitzenmacher [3]	static	CKA1	$O(\frac{n}{p})$	$O(mn)$
Curtmola et al. [6] (SSE-1)	static	CKA1	$O(r)$	$O(m+n)$
Curtmola et al. [6] (SSE-2)	static	CKA2	$O(r)$	$O(mn)$
van Liesdonk et al. [20]	dynamic	CKA2	$O(n)$	$O(mn)$
Chase and Kamara [4]	static	CKA2	$O(r)$	$O(mn)$
Kurosawa and Ohtaki [13]	static	UC	$O(n)$	$O(mn)$
Kamara et al. [11]	dynamic	CKA2	$O(r)$	$O(m+n)$
THIS WORK	dynamic	CKA2	$O(\frac{r}{p}\log n)$	$O(mn)$

1. Searches for a keyword w run in $O((r/p)\log n)$ parallel time, where r is the number of documents containing keyword w. Note that for $p = \omega(\log n)$, our parallel search time is $o(r)$, i.e., *less than the optimal sequential search time*.[1]
2. Updates for a document f containing q unique keywords run in $O((m/p)\log n)$ parallel time. Again, in that case, for $p = \omega((m/q)\log n)$, our parallel update time is $o(q)$, i.e., *less than the optimal sequential update time*.
3. Finally, unlike the updates supported in the works of van Liesdonk et al. [20] and Kamara et al. [11], the updates of our scheme *do not leak* information about the keywords contained in a newly added or deleted document f, apart from information that is leaked through search tokens that have been issued in the past (updates in our scheme however require one round of interaction, as in [20]). For example, if we start adding documents into our encrypted index *before* we perform *any search* (which is common in practical applications like streaming), *no* information is leaked due to these update operations.

We finally note that our scheme enjoys security against adaptive chosen-keyword attacks (CKA2), as defined by Curtmola et al. [6] and can also be implemented efficiently in external memory (with logarithmic I/O overhead).

Our Approach. Our approach is based on a new tree-based multi-map data structure we refer to as a keyword red-black (KRB) tree. As we show in Section 3, KRB trees can index a document collection in such a way that keyword search can be performed in $O(r\log n)$ sequential time and $O(\frac{r}{p}\log n)$ parallel time. In addition, a KRB tree supports efficient updates because all the information it contains about a given file f can be found and updated in $O(\log n)$ time. To construct our SSE scheme, we show how to encrypt KRB trees based on simple and efficient primitives like pseudorandom functions and permutations and a random oracle. The resulting scheme is CKA2-secure and preserves the same (asymptotic) efficiency as an unencrypted KRB tree.

[1] Taking $p = \omega(\log n)$ is very reasonable, given that even 64-core CPUs are now available, e.g., see the TILE64 processor at `http://en.wikipedia.org/wiki/TILE64`.

Related Work. The problem of searching on symmetrically encrypted data can be solved in its full generality using the work of Goldreich and Ostrovsky [9] on oblivious RAM (ORAM). In addition to handling any type of search query, this approach also provides the strongest levels of security, namely the server does not learn any information about the data or the queries—not even information inferred by the client's access pattern. This approach requires interaction and has a high overhead for the server and the client, especially for more involved functionalities like search. Lorch et al. [14] explored the notion of parallel ORAM and proposed a parallel ORAM scheme based on a binary tree approach. Finally, while a recently introduced ORAM scheme [19] has been shown to be relatively practical, it still requires $O(\sqrt{n})$ storage at the client.

Song et al. [18] were the first to explicitly consider the problem of searchable encryption and presented a non-interactive solution that with search time that is linear in the length of the data collection. Goh [8] introduced formal security definitions for SSE and proposed a construction based on Bloom filters [1] that requires $O(n)$ search time and results in false positives. Chang and Mitzenmacher [3] proposed an alternative security definition and construction also with $O(n)$ search time but without false positives. While both the schemes of Goh and of Chang and Mitzenmacher can be naively parallelized, this would require a linear (in n) number of cores.

Curtmola et al [6]. gave the first constructions (SSE-1 and SSE-2) to achieve sublinear (and in fact optimal) search time. Like previous work [8,3], SSE-1 was shown secure against chosen-keyword attacks (CKA1). In that work, it was noted, however, that CKA1-security does not suffice for practical use. To address this, the stronger notion of security against *adaptive* chosen-keyword attacks (CKA2) was proposed and a CKA2-secure SSE scheme (SSE-2) was proposed. A similar CKA2-secure scheme was also described by Chase and Kamara [4] but its space complexity is high. Finally, recent work by Kurosawa et al. [13] shows how to construct a (verifiable) SSE scheme that is universally composable (UC). While UC-security is a stronger notion of security than CKA2-security, their construction requires linear search time.

None of the above schemes are "explicitly dynamic", i.e., to handle dynamic data one must use general dynamization techniques that are relatively inefficient (except for the scheme of Goh [8] which unfortunately has linear search time). The first explicitly dynamic scheme was presented by Liesdonk et al. [20], but unfortunately that construction supports a limited number of updates and can has linear search time *in the worst case*. Recently, Kamara et al. [11] constructed an SSE scheme that is CKA2-secure, achieves optimal search time (with small constants) and is explicitly dynamic, Unfortunately, this construction leaks the tokens of the keywords contained in an updated document. Our construction avoids such leakage and, in addition, is *considerably* simpler. A complete comparison of all the schemes can be found in Table 1.

All the above solutions are either inherently sequential or admit naive parallelization. For example, in the schemes based on the inverted index approach of Curtmola et al [4,6,11,13,20], in order to retrieve the documents $d_1, d_2, \ldots, d_r$ containing keyword w, the algorithm uses the pointer to d_1 along with a special token $t(w)$ to decrypt the pointer to d_2, then it uses the pointer to d_2 and the token $t(w)$ to decrypt the pointer to d_3 until the final pointer to d_r can be decrypted. Similarly, to delete a document that contains keywords $w_1, w_2, \ldots, w_q$, the algorithm deletes the entry corresponding

to w_i only after it deletes the entry corresponding to w_{i-1}. Both these procedures are sequential in nature.

Although this work focuses on the case of single-keyword equality queries, we note that more complex queries have also been considered. This includes conjunctive queries in the symmetric key setting [10]; it also includes conjunctive queries [16,2], comparison and subset queries [2], and range queries [17] in the public-key setting.

2 Preliminaries

Our construction makes use of several basic cryptographic primitives. A *private-key encryption* scheme consists of three algorithms $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that $\mathsf{Gen}(1^k; r)$ is a probabilistic polynomial-time (PPT) algorithm that takes a security parameter k and randomness r and returns a secret key K; $\mathsf{Enc}(K, m)$ is PPT algorithm that takes a key K and a message m and returns a ciphertext c; $\mathsf{Dec}(K, c)$ is a deterministic algorithm that takes a key K and a ciphertext c and returns m if K was the key under which c was produced. Informally, a private-key encryption scheme is CPA-secure if the ciphertexts it outputs do not leak any partial information about the plaintext even to an adversary that can adaptively query an encryption oracle.

In addition to encryption schemes, we also make use of *pseudorandom functions* (PRF), which are polynomial-time computable functions that cannot be distinguished from random functions by any PPT adversary and *random oracles*, to which we assume all parties have black-box access. We refer the reader to [12] for formal definitions of CPA-security, PRFs and random oracles.

Keyword Hash Tables. In our construction we use static hash tables [5] to store some specific information for each one of the m keywords. The entries of the hash table λ are tuples $(\mathsf{key}, \mathsf{value})$, where the key key is from a domain of exponential size, i.e., from $\{0,1\}^k$ and value is an encryption of a boolean value. However, the maximum number of entries in our hash table will be polynomial in k and equal to m, the number of keywords. If, for a table λ, the key field is from $\{0,1\}^k$, and there are at most m entries in λ, then we say λ is a (k, m) hash table. For each $x \in \{0,1\}^k$, we denote with $\lambda[x]$ the value associated with key x, if key x exists.

Searchable Symmetric Encryption. SSE allows a client to encrypt data so that it can later generate search tokens which the server can use to search over the encrypted data and return the appropriate encrypted files.

The encryption algorithm in an SSE scheme takes as input an index δ, a sequence of n files $\mathbf{f} = (f_{i_1}, \ldots, f_{i_n})$ that have unique identifiers $\mathbf{i} = (i_1, \ldots, i_n)$,[2] and a universe of keywords $\mathbf{w} = (w_1, \ldots, w_m)$. The index δ efficiently maps a keyword $w \in \mathbf{w}$ to a set of identifiers $\mathbf{i}_w \subseteq \mathbf{i}$ that correspond to a set of files $\mathbf{f}_w \subseteq \mathbf{f}$. The encryption algorithm outputs an encrypted index γ and a sequence of n ciphertexts $\mathbf{c} = (c_{i_1}, \ldots, c_{i_n})$, corresponding to the identifiers $\mathbf{i} = (i_1, \ldots, i_n)$. We assume all the ciphertexts include the identifiers of their plaintext files. All known constructions (except for [18]) can encrypt the files $\mathbf{f}$ using any CPA-secure encryption scheme. The encrypted index γ and

[2] The identifiers are chosen uniformly at random and do not reveal any information about the plaintexts they are representing, e.g., they can be the i-nodes of the underlying file system. In order not to overload the notation, file f_i has identifier i.

the ciphertexts $\mathbf{c}$ do not reveal any information about $\mathbf{f}$ other than the number of files n and their length,[3] so they can be stored safely at an untrusted cloud provider.

To search for a keyword w, the client generates a search token τ_s and given τ_s, γ and $\mathbf{c}$, the provider can find the subset of ciphertexts $\mathbf{c}_w \subseteq \mathbf{c}$ that contain w. Notice that the provider learns some limited information about the client's query. In particular, it knows that whatever keyword the client is searching for is contained in whichever files resulted in the ciphertexts $\mathbf{c}_w$. A more serious limitation of known SSE constructions (including ours) is that the tokens they generate are deterministic, in the sense that the same token will always be generated for the same keyword. This means that searches leak statistical information about the user's search pattern. Currently, it is not known how to design efficient SSE schemes with probabilistic trapdoors.

Our scheme supports parallel keyword search as well as parallel addition and deletion of files. For updates, we use 1.5 rounds of interaction (i.e., three messages between client and server). For example, to add a file f, the client generates—with the help of the server—an addition token τ_a. Given such token and γ, the server can update the index γ. The same pattern occurs for deletion. To account for this interaction, we slightly change the definition of CKA2-security for dynamic SSE which was recently presented by Kamara et al. [11].

Definition 1 (Dynamic SSE). *A dynamic SSE scheme is a tuple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{SrchToken}, \mathsf{Search}, \mathsf{UpdHelper}, \mathsf{UpdToken}, \mathsf{Update}, \mathsf{Dec})$ *of eight polynomial-time algorithms such that:*

1. $K \leftarrow \mathsf{Gen}(1^k)$*: is a probabilistic algorithm that takes as input a security parameter* k *and outputs a secret key* K*.*
2. $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}(K, \delta, \mathbf{f})$*: is a probabilistic algorithm that takes as input a secret key* K*, an index* δ *and a sequence of files* $\mathbf{f}$*. It outputs an encrypted index* γ *and a sequence of ciphertexts* $\mathbf{c}$*.*
3. $\tau_s \leftarrow \mathsf{SrchToken}(K, w)$*: is a (possibly probabilistic) algorithm that takes as input a secret key* K *and a keyword* w *and outputs a search token* τ_s*.*
4. $\mathbf{i}_w \leftarrow \mathsf{Search}(\gamma, \mathbf{c}, \tau_s)$*: is a deterministic algorithm that takes as input an encrypted index* γ*, a sequence of ciphertexts* $\mathbf{c}$ *and a search token* τ_s*. It outputs a sequence of identifiers* $\mathbf{i}_w \subseteq \mathbf{i}$*.*
5. $\mathsf{info}_{i,u} \leftarrow \mathsf{UpdHelper}(i, u, \gamma, \mathbf{c})$*: is a deterministic algorithm that takes as input a file identifier* i*, the update type* $u \in \{add, delete\}$*, an encrypted index* γ *and sequence of ciphertexts* $\mathbf{c}$*. It outputs helper information* $\mathsf{info}_{i,u}$ *for the specific update.*
6. $\tau_u \leftarrow \mathsf{UpdToken}(K, f_i, \mathsf{info}_{i,u})$*: is a (possibly probabilistic) algorithm that takes as input a secret key* K*, a file* f_i *and the respective helper information* $\mathsf{info}_{i,u}$ *as output by algorithm* $\mathsf{UpdHelper}$*. It outputs an update token* τ_u *for the update type* $u \in \{add, delete\}$*.*
7. $(\gamma', \mathbf{c}') \leftarrow \mathsf{Update}(\gamma, \mathbf{c}, \tau_u)$*: is a deterministic algorithm that takes as input an encrypted index* γ*, a sequence of ciphertexts* $\mathbf{c}$ *and an update token* τ_u*. It outputs a new encrypted index* γ' *and new sequence of ciphertexts* $\mathbf{c}'$*.*
8. $f \leftarrow \mathsf{Dec}(K, c)$*: is a deterministic algorithm that takes as input a secret key* K *and a ciphertext* c *and outputs a file* f*.*

[3] Note that this information leakage can be mitigated by padding if desired.

We can now easily define correctness of the above dynamic SSE scheme definition.

Definition 2 (Correctness). *Let $\mathcal{D}$ be a dynamic SSE scheme consisting of the tuple of eight algorithms as given in Definition 1. We say that $\mathcal{D}$ is correct if for all $k \in \mathbb{N}$, for all K output by $\mathsf{Gen}(1^k)$, for all tuples $(\delta, \mathbf{f})$, for all tuples $(\gamma, \mathbf{c})$ output by one execution of $\mathsf{Enc}(K, \delta, \mathbf{f})$ and successive executions of $\mathsf{Update}(\gamma, \mathbf{c}, \tau_u)$, where τ_u is the update token output by $\mathsf{UpdToken}(K, f_i, \mathsf{UpdHelper}(i, u, \gamma, \mathbf{c}))$ for all files f_i and all $u \in \{add, delete\}$, for all keywords w, for all tokens τ_s output by $\mathsf{SrchToken}(K, w)$, for all $\mathbf{i}_w$ output by $\mathsf{Search}(\gamma, \mathbf{c}, \tau_s)$, the plaintexts $\mathbf{f}_w = \{\mathsf{Dec}(K, c_i) : i \in \mathbf{i}_w\}$ are all the plaintexts in $\mathbf{f}$ containing keyword w.*

Security. Intuitively, the security guarantee we require from a dynamic SSE scheme is that (1) given an encrypted index γ and a sequence of ciphertexts $\mathbf{c}$, no adversary can learn any partial information about the files $\mathbf{f}$; and that (2) given, in addition, a sequence of search tokens $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_t)$ for an adaptively generated sequence of keywords $\mathbf{q} = (q_1, \ldots, q_t)$ (which can be for the search, add or delete operations), no adversary can learn any partial information about either $\mathbf{f}$ or $\mathbf{q}$. This exact intuition can be difficult to achieve and most known efficient and non-interactive SSE schemes [3,6,8,11,13,20] reveal the access and search patterns.[4] We therefore need to weaken the definition appropriately by allowing some limited information about the messages and the queries to be revealed to the adversary. To capture this, we follow the approach of [4] and [6] and parameterize our definition with a set of leakage functions that capture precisely what is being leaked by the ciphertext and the tokens. Specifically, the leakage functions we consider in this work are defined as follows:

1. $\mathcal{L}_1(\delta, \mathbf{f})$: given the index δ and the set of files $\mathbf{f}$ (along with the respective identifiers), this function outputs the number of keywords m, the number of files n, the identifiers $\mathbf{i}$ of the file and the size of each file;
2. $\mathcal{L}_2(\delta, \mathbf{f}, w, t)$: this function takes as input the index δ, the set of files $\mathbf{f}$ and a keyword w for a search operation that took place at time t. It outputs two different types of information, namely the search pattern $\mathcal{P}(\delta, q, t)$ and the access pattern $\mathcal{A}(\delta, \mathbf{f}, w, t)$, both of which are described in the following definitions.

Definition 3 (Search pattern). *Given a search query for keyword w at time t, the search pattern $\mathcal{P}(\delta, q, t)$ is defined as the binary vector of length t with a 1 at location i if the search at time $i \leq t$ was for w; and 0 otherwise. Namely the search pattern reveals whether the same search was performed in the past or not.*

Definition 4 (Access pattern). *Given a search query for keyword w at time t, the access pattern $\mathcal{A}(\delta, \mathbf{f}, w, t)$ is defined as the identifiers in the set $\mathbf{f}_w$ at time t.*

Discussion on Leakage. In our scheme (described in Section 3), if one searches for w at time t, the output of the leakage function $\mathcal{L}_2(\delta, \mathbf{f}, w, t)$ consists of $\mathcal{A}(\delta, \mathbf{f}, w, t)$, which includes the identifiers $\mathbf{i}_w$ of the files $\mathbf{f}_w$ that contain keyword w at time t. As we add files to the collection, $\mathcal{A}(\delta, \mathbf{f}, w, t)$ is expanded with the identifiers of the new

[4] One exception is the construction described in [4] which leaks only the access and the intersection patterns.

files that contain w. This is a limitation of our construction since search tokens are valid even for future documents added to the collection and effectively allow the server to use old tokens to search over newly-added documents. It is an open problem to construct *efficient* dynamic SSE schemes that do not have this limitation.[5]

We note that with our construction, if one adds documents to the index *before* performing any search operations (which is common in practical applications like streaming), *no information* is leaked due to the updates. From a security point of view, this is the main difference between our scheme and the recently proposed construction of Kamara et al. [11], which always leaks information on an update. In this sense, our construction satisfies a slightly stronger notion of security since since the leakage of our updates is conditional on previous operations and does not occur unconditionally. This is also why, unlike [11], we do not use an explicit leakage algorithm for updates.

Finally, as observed in [6], another issue with respect to SSE security is whether the scheme is secure against *adaptive* chosen-keyword attacks (CKA2) or only against *non-adaptive* chosen keyword attacks (CKA1). The former guarantees security even when the client's queries are based on the encrypted index and the results of previous queries. The latter only guarantees security if the client's queries are independent of the index and of previous results. Our scheme achieves the stronger notion of security, namely CKA2-security. In our definition of security below, we adapt the notion of CKA2-security from [11] to the setting of dynamic SSE with interactive updates. We model the interaction required by our scheme with an algorithm $\mathsf{UpdHelper}$.

Definition 5 (CKA2-security). *Let $\mathcal{D}$ be a dynamic SSE scheme consisting of the tuple of eight algorithms as given in Definition 1. Consider the following probabilistic experiments, where $\mathcal{A}$ is a stateful adversary, $\mathcal{S}$ is a stateful simulator and $\mathcal{L}_1$ and $\mathcal{L}_2$ are stateful leakage algorithms:*

$\mathbf{Real}_{\mathcal{A}}(k)$*: the challenger runs* $\mathsf{Gen}(1^k)$ *to generate a key K. $\mathcal{A}$ outputs a tuple $(\delta, \mathbf{f})$ and receives $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}(K, \delta, \mathbf{f})$ from the challenger. The adversary makes a polynomial number of adaptive queries by picking $q \in \{w, f_i\}$. If $q = w$ is a search query then the adversary receives from the challenger a search token $\tau_s \leftarrow \mathsf{SrchToken}(K, w)$. If $q = f_i$ is an update of type u, then the adversary also sends the helper information $\mathsf{info}_{i,u} \leftarrow \mathsf{UpdHelper}(i, u, \gamma, \mathbf{c})$ to the challenger and then receives from the challenger the update token $\tau_u \leftarrow \mathsf{UpdToken}(K, f_i, \mathsf{info}_{i,u})$. Finally, $\mathcal{A}$ returns a bit b that is output by the experiment.*

$\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k)$*: $\mathcal{A}$ outputs a tuple $(\delta, \mathbf{f})$. Given $\mathcal{L}_1(\delta, \mathbf{f})$, $\mathcal{S}$ generates and sends a pair $(\gamma, \mathbf{c})$ to $\mathcal{A}$. The adversary makes a polynomial number of adaptive queries by picking $q \in \{w, f_i\}$. If $q = w$ is a search query then the simulator is given $\mathcal{L}_2(\delta, \mathbf{f}, w, t)$. If $q = f_i$ is an update of type u, the simulator is given the updated output of $\mathcal{L}_2(\delta, \mathbf{f}, w, t)$ for all keywords w that have appeared before in the adaptive queries.*[6] *The adversary also sends $\mathsf{info}_{i,u} \leftarrow \mathsf{UpdHelper}(i, u, \gamma, \mathbf{c})$ to the simulator. The simulator returns an appropriate token τ. Finally, $\mathcal{A}$ returns a bit b that is output by the experiment.*

[5] The scheme of Chang and Mitzenmacher [3] does not have this limitation but requires linear time search.

[6] This will be used to simulate searches for previous tokens, the output of which (of those searches) has changed due to the update. This is the only leakage that our updates cause.

We say that $\mathcal{D}$ is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive dynamic chosen-keyword attacks if for all PPT *adversaries $\mathcal{A}$, there exists a* PPT *simulator $\mathcal{S}$ such that*

$$|\Pr[\mathbf{Real}_{\mathcal{A}}(k) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k) = 1]| \leq \mathsf{neg}(k).$$

3 Our Dynamic SSE Construction

In this section we describe our parallel and dynamic construction. Let $\mathbf{f} = (f_{i_1}, \ldots, f_{i_n})$ be a sequence of documents with corresponding identifiers $\mathbf{i} = (i_1, \ldots, i_n)$ over a set of keywords $\mathbf{w} = (w_1, \ldots, w_m)$. We view each individual f_i document as a bit-string of polynomial length, i.e., $f_i = \{0,1\}^{\mathsf{poly}(k)}$. Recall that an index δ maps a keyword $w \in \mathbf{w}$ to a set of documents identifiers $\mathbf{i}_w$.

We assume that the universe of keywords is fixed but that the number of documents can grow. In particular, we assume that the total number of keywords m is much smaller than the number of files n. We now introduce a standard (i.e., unencrypted) data structure keyword search which we refer to as a *keyword red-black* tree. KRB trees will be the basis of our dynamic SSE scheme.

The KRB Tree. The KRB tree is a dynamic data structure that—similarly to an inverted index—can be used to efficiently answer multi-map queries. A KRB tree δ is constructed from a set of documents $\mathbf{f} = (f_{i_1}, \ldots f_{i_n})$ (which include the identifiers $\mathbf{i} = (i_1, \ldots, i_n)$) and a universe of keywords $\mathbf{w}$. The data structure is constructed using the following procedure, which we denote as $\mathsf{buildIndex}(\mathbf{f})$:

1. Assume a total order on the documents $\mathbf{f} = (f_{i_1}, \ldots, f_{i_n})$, imposed by the ordering of the identifiers $\mathbf{i} = (i_1, \ldots, i_n)$. Build a red-black tree T on top of $i_1, \ldots, i_n$. At the leaves, store pointers to the appropriate documents. We assume the documents are stored separately, e.g., on disk. Note that this is a slight modification of a red black tree since the tree is constructed on top of the identifiers but the leaves store pointers to the files.
2. At each internal node u of the tree, store an m-bit vector $\mathtt{data}_u$. The i-th bit of $\mathtt{data}_u$ accounts for keyword w_i, for $i = 1, \ldots, m$. Specifically, if $\mathtt{data}_u[i] = 1$, then there is at least one path from u to some leaf that stores some identifier j, such that f_j contains w_i;
3. We guarantee the above property of vectors $\mathtt{data}_u$, by computing $\mathtt{data}_u$ as follows: for every leaf l storing identifier j, set $\mathtt{data}_l[i] = 1$ if and only if document f_j contains keyword w_i. Now let u be an internal node of the tree T with left child v and right child z. The vector $\mathtt{data}_u$ of the internal node u is computed recursively as follows:
$$\mathtt{data}_u = \mathtt{data}_v + \mathtt{data}_z\,, \tag{1}$$
where $+$ denotes the *bitwise* boolean OR operation.

To search for a keyword w in a KRB tree T one proceeds as follows. Assuming that w has position i in the m-bit vectors stored at the internal nodes, check the bit at position i of node v and examine v's children if the bit is 1. When this traversal is over, return all the leaves that were reached.

The intuitive reason the KRB tree is so useful for our purposes is that it allows *both* keyword-based operations (by following paths from the root to the leaves) and file-based operations (by following paths from the leaves to the root). As we will see later, this property is useful for handling updates efficiently. We now have the following:

Lemma 1 (KRB tree data structure). *Let* $\mathbf{f} = (f_{i_1}, \ldots, f_{i_n})$ *be a set of* n *documents containing keywords from a dictionary of* m *keywords* $\mathbf{w} = (w_1, \ldots, w_m)$*. Then there exists a dynamic data structure for keyword search such that:* (a) *the space complexity of the data structure is* $O(mn)$*;* (b) *constructing the data structure takes time* $O(mn)$*;* (c) *the search time for a keyword* w *is* $O(r \log n)$*, where* r *is the number of documents containing* w*;* (d) *the time to insert and delete a document* f *is* $O(q \log n)$*, where* q *is the number of unique keywords contained in document* f*;* (e) *search and updates take parallel* $O(\frac{r}{p} \log n)$ *and* $O(\frac{q}{p} \log n)$ *time, respectively, with* p *processors in the concurrent-read-exclusive-write (CREW) model of parallel computation.*

Proof. Since the underlying red-black tree has space complexity $O(n)$ and we have to store at each node a bit-vector of m bits, it follows that the space complexity of the KRB tree is $O(mn)$. Now due to the property of Relation 1, given the document collection $\mathbf{f}$ one can start building the data structure following a postorder traversal. Since a postorder traversal visits $O(n)$ nodes and the time spent at each node is $O(m)$ (to compute the OR of two m-bit vectors), the time required for constructing the data structure is $O(mn)$.

Recall that sequential search for a keyword w (corresponding to position i of the m-bit vectors) proceeds as follows: while the bit at position i of node v is 1, examine v's children. Therefore the search procedure will traverse as many paths as the documents containing keyword w, namely r paths. Since the maximum height of the red-black tree is maintained to be $O(\log n)$ [5], the search time is $O(r \log n)$.

The parallel search is executed as follows. Let $0, 1, \ldots, p-1$ be the processors that are available. Processor 0 queries the root r of the tree for a specific keyword. If the search is to be continued in both the subtrees T_u and T_v of the processor's children u and v, processor 0 continues with one subtree (say T_u) and assigns the other subtree T_v to be explored by another processor. The same algorithm is recursively applied for nodes u and v. However, if at some point during the search no more processors are available (i.e., all p processors are working—this test can be achieved with concurrent read of the same data structure by all processors, that is why we require CREW model), the current processor simply selects one of its two possible children to continue, marks the other child c as "unexplored" and pushes c into a local stack of unexplored nodes so that it can be explored later. All p processors terminate their execution in $O(\log n)$ time, outputting p documents containing the queried keyword. In the second round, each processor i starts over by popping (and removing) a node c from the processor's local stack and by resuming the search from that node c. Again, during that round, each processor pushes nodes it cannot explore into its local stack. At the end of the second round, at least another p documents are retrieved in $O(\log n)$ time. Eventually, after r/p rounds of logarithmic time, all documents are retrieved (and all the local stacks are guaranteed to be empty). Therefore, search executes in parallel $O(\frac{r}{p} \log n)$ time.

The sequential update time follows from the complexity of the update of the red-black tree [5]. Since it involves bit operations between q independent vector positions, it can be implemented in $O(\frac{q}{p}\log n)$ parallel time. This completes the proof. □

We now show the following corollary, establishing the number of processors required for improving the *optimal* sequential performance:

Corollary 1 (Number of processors). *With $\omega(\log n)$ processors available, parallel searches take $o(r)$ time and parallel updates take $o(q)$ time in a KRB tree.*

KRB-Based Dynamic SSE. We now describe in detail our KRB-based parallel and dynamic SSE construction. Let $\mathbf{f} = (f_{i_1}, \ldots, f_{i_n})$ be the set of documents and $\mathbf{w} = (w_1, \ldots, w_m)$ be the set of keywords. We use the following cryptographic primitives:

1. A pseudo-random function $\mathtt{G} : \{0,1\}^k \times \{w_1, \ldots, w_m\} \to \{0,1\}^k$;
2. Another pseudo-random function $\mathtt{P} : \{0,1\}^k \times \{w_1, \ldots, w_m\} \to \{0,1\}^k$;
3. A random oracle $\mathtt{H} : \{0,1\}^k \times \{0,1\} \to \{0,1\}$.

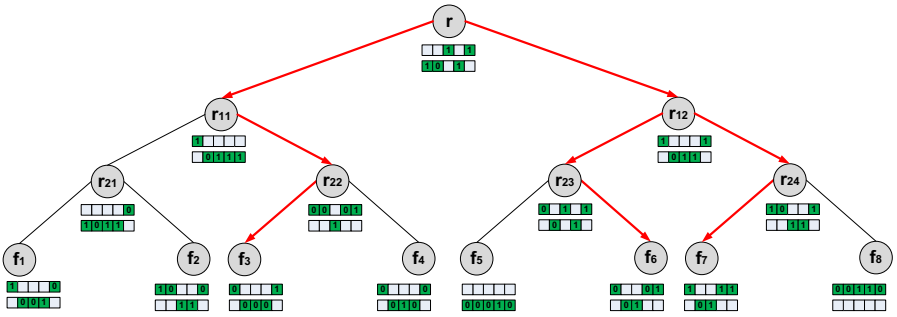


Fig. 1. The construction of a dynamic symmetric searchable encryption (DSSE) scheme using the KRB tree data structure, for a collection of $n = 8$ documents indexed over $m = 5$ keywords. Note that for each node v we store two vectors. The encryption of the actual bit of position i at node v is stored to either hash table λ_{0v} or hash table λ_{1v}, depending on the output of the random oracle. The red arrows indicate the search for keyword 5, returning documents f_3, f_6, f_7. Note that the two searches displayed can be parallelized.

We now describe the algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{SrchToken}, \mathsf{Search}, \mathsf{UpdHelper}, \mathsf{UpdToken}, \mathsf{Update}, \mathsf{Dec})$ of the DSSE scheme from Definition 1 in detail:

Algorithm $K \leftarrow \mathsf{Gen}(1^k)$: Generate three random k-bit strings K_1, K_2 and r. Instantiate one private-key CPA-secure encryption scheme for encrypting documents by calling $K_3 \leftarrow \mathcal{E}.\mathsf{Gen}(1^k; r)$. Set $K := (K_1, K_2, K_3)$;

Algorithm $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}(K, \delta, \mathbf{f})$: Let $\delta \leftarrow \mathsf{buildIndex}(\mathbf{f})$ be a KRB tree and proceed as follows:

1. Instantiate a second private-key CPA-secure encryption scheme $\mathcal{R}$. Derive a secret key SK_i per keyword w_i by calling $\mathsf{SK}_i = \mathcal{R}.\mathsf{Gen}(1^k; \mathtt{G}_{K_2}(w_i))$, for $i = 1, \ldots, m$.
2. For $1 \leq j \leq n$, let $c_{i_j} \leftarrow \mathcal{E}.\mathsf{Enc}(K_3, f_{i_j})$, outputting a vector of ciphertexts $\mathbf{c}$;

3. Store $\mathbf{c}$ on disk (note that the identifiers $\mathbf{i}$ remain the same) and then delete $\mathbf{f}$. For every node v of the KRB tree T, that has identifier $\mathtt{id}(v)$, do the following: **(a)** Instantiate two (k, m) keyword hash tables λ_{0v} and λ_{1v}—see Section 2 for the definition of a (k, m) keyword hash table. Store λ_{0v} and λ_{1v} at v; **(b)** For every $i = 1, \ldots, m$, set $\lambda_{\mathsf{b}v}[\mathtt{P}_{K_1}(w_i)] \leftarrow \mathcal{R}.\mathsf{Enc}(\mathsf{SK}_i, \mathtt{data}_v[i])$, where $\mathsf{b} = \mathtt{H}(\mathtt{P}_{K_1}(w_i), \mathtt{id}(v))$ is a bit computed as the output of a random oracle and $\mathtt{data}_v$ is the vector at node v in T; **(c)** Store a random string at $\lambda_{|1-\mathsf{b}|v}[\mathtt{P}_{K_1}(w_i)]$ (namely at each node v, the bit b dictates which hash table (either λ_{1v} or λ_{0v}) contains the actual entry for keyword w_i); **(d)** Delete vector $\mathtt{data}_v$.
4. Output $\gamma := T$ and $\mathbf{c} := (c_{i_1}, \ldots, c_{i_n})$.

Algorithm $\tau_s \leftarrow \mathsf{SrchToken}(K, w_i)$**:** Call $\mathcal{R}.\mathsf{Gen}(1^k; \mathtt{G}_{K_2}(w_i))$ to output the secret key SK_i and output the search token $\tau_s := (\mathtt{P}_{K_1}(w_i), \mathsf{SK}_i)$;

Algorithm $\mathbf{i}_w \leftarrow \mathsf{Search}(\gamma, \mathbf{c}, \tau_s)$**:** Parse τ_s as (τ_1, τ_2). Call $\mathsf{search}(r)$, where r is the root of the KRB tree T. Let v and z be the left and right child of a node u respectively. Algorithm $\mathsf{search}(u)$ is recursively defined as follows:

1. Output a bit $\mathsf{b} = \mathtt{H}(\tau_1, \mathtt{id}(u))$ and compute $\mathsf{a} = \mathcal{R}.\mathsf{Dec}(\tau_2, \lambda_{\mathsf{b}u}[\tau_1])$;
2. If $\mathsf{a} = 0$, return;
3. If u is a leaf, set $\mathbf{c}_w := \mathbf{c}_w \cup c_u$, where c_u is the ciphertext corresponding to file identifier u (and also stored at node u). Else call $\mathsf{search}(v)$ and $\mathsf{search}(z)$.

Output $\mathbf{c}_w$ (note here that, by Lemma 1, this algorithm can be parallelized to execute in $O(\frac{|\mathbf{c}_w|}{p} \log n)$ time, where p is the number of processors).

Algorithm $\mathsf{info}_{i,u} \leftarrow \mathsf{UpdHelper}(i, u, \gamma, \mathbf{c})$**:** The update u in this algorithm refers to the document f_i and is either an insertion of document f_i or a deletion of document f_i (with identifier i). To compute the information $\mathsf{info}_{i,u}$, the algorithm performs the structural update[7] on the KRB tree T. Note that in order to perform the structural update, no access to the actual content of the documents is required, since such an update is only based on the identifier i. The information $\mathsf{info}_{i,u}$ consists of the portion $T(u)$ of the KRB tree T that is *accessed* during the update. In other words $T(u)$ suffices to perform the update, in absence of the rest of the tree $T - T(u)$. Moreover, the size of $T(u)$ is $O(m \log n)$ in the worst case, given that a red-black tree update takes $O(\log n)$ time in the worst case (see Lemma 1) and therefore it can "touch" as many bits (multiplied by the size m of the encrypted vectors that are stored at the tree nodes). To give an example, in Figure 1, $\mathsf{info}_{3,u}$, where u is "deletion of file f_3", contains the nodes r_{22}, r_{11} and r (along with the encrypted vectors stored at them).

Algorithm $\tau_u \leftarrow \mathsf{UpdToken}(K, f_i, \mathsf{info}_{i,u})$**:** If the update u is an insertion of document f_i, compute first an encryption $c_i \leftarrow \mathcal{E}.\mathsf{Enc}(K_3, f_i)$ of the added document. Now let $\mathsf{info}_{i,u}$ contain the specific portion $T(u)$ of the KRB tree, as returned by $\mathsf{UpdHelper}$. Perform the structural update on $T(u)$ and let $T'(u)$ be the new subtree after the update.

[7] The structural update involves the necessary rotations that are performed during an update of a red-black tree, so that its height can be maintained to be logarithmic. The details of such operations are described in the book by Cormen, Leiserson, Rivest and Stein (CLRS) [5].

Every node v of $T'(u)$ that has new/modified ancestors (compared to its structure in $T(u)$) must also change its encrypted local information, since this is always computed as a function of its ancestors (in a sense, its encrypted local information is going to be recomputed and rerandomized). Also it changes its identifier from $\mathtt{id}(v)$ to $\mathtt{id}(v')$. Specifically, for every such node $v \in T'(u)$ we do the following:

1. Instantiate two *new* (k, m) keyword hash tables with random entries λ_{0v} and λ_{1v}. Store λ_{0v} and λ_{1v} at v;
2. Update the new hash tables by setting $\lambda_{\mathsf{b}v}[\mathtt{P}_{K_1}(w_i)] \leftarrow \mathcal{R}.\mathsf{Enc}(\mathsf{SK}_i, \mathtt{data}_v[i])$ $(i = 1, \ldots, m)$, where $\mathsf{b} = \mathtt{H}(\mathtt{P}_{K_1}(w_i), \mathtt{id}(v'))$ is a bit computed as the output of the random oracle and $\mathtt{data}_v$ is the *updated* vector due to the update;
3. Output $\tau_u := (T'(u), c_i)$.

Algorithm $(\gamma', \mathbf{c}') \leftarrow \mathsf{Update}(\gamma, \mathbf{c}, \tau_u)$**:** On input τ_u, just *copy* the new information $T'(u)$ to the already structurally updated KRB tree T (note that T was structurally updated by the UpdHelper algorithm) and output the new encrypted index γ' and the new set of ciphertexts $\mathbf{c}'$.
Algorithm $f_i \leftarrow \mathsf{Dec}(K, c_i)$**:** Output the plaintext $f_i := \mathcal{E}.\mathsf{Dec}(K_3, c_i)$.

Correctness, security and main result. In this section we prove that our scheme is correct (Lemma 2) and secure (Lemma 3). Then we give the final result (Theorem 1).

Lemma 2 (Correctness). *The dynamic searchable symmetric encryption scheme presented above is correct according to Definition 2.*

Proof. Note that the Update algorithm modifies the encrypted KRB tree in a way that Relation 1 is satisfied, since the correct value of the bit at each node v is stored as indicated by the random oracle. Since the Search algorithm will query the same random oracle, it follows that when searching for a keyword w, it will follow the tree paths that lead to the correct set of documents.

Lemma 3 (Security). *The dynamic searchable symmetric encryption scheme presented above is $(\mathcal{L}_1, \mathcal{L}_2)$-secure in the random oracle model and according to Definition 5 (CKA-2 security), where $\mathcal{L}_1$ leaks the number of the keywords, the number of the documents, the identifiers of the documents and the size of each document; and $\mathcal{L}_2$ leaks the search pattern and the access pattern, as defined in Definitions 3 and 4.*

The proof of Lemma 3 can be found in the Appendix. We now state our final theorem.

Theorem 1 (Parallel and dynamic SSE scheme). *There exists a dynamic searchable symmetric encryption scheme for a collection of n documents indexed over a set of m keywords such that: (a) it is correct according to Definition 2 and secure according to Definition 3 and in the random oracle model; (b) searches for keywords w can be performed in sequential $O(r \log n)$ time or parallel $O(\frac{r}{p} \log n)$ time in the CREW model of parallel computation, where r is the number of documents containing w and p is the number of processors available; (c) additions or deletions of a document f_i can be performed in sequential $O(m \log n)$ time or parallel $O(\frac{m}{p} \log n)$ time, with one interaction, and have $O(m \log n)$ communication complexity; (d) the encrypted data structure γ has size $O(mn)$ and the local space needed is $O(1)$.*

Proof. Correctness follows from Lemma 2, security follows from Lemma 3 and most complexities (including the parallel ones) follow from Lemma 1. Note that the update complexity is not $O(q \log n)$, as in Lemma 1—it is $O(m \log n)$, since we do not want to reveal *which keywords are contained in the file of the update*. Also, our scheme has interactive updates due to the algorithm UpdHelper in the definition. □

In the following corollary we give the minimum number of processors required so that the parallel operations of our scheme outperform the *optimal* sequential complexity.

Corollary 2 (Number of processors). *With $\omega(\log n)$ processors available in the above scheme, parallel searches take $o(r)$ time. Similarly, with $\omega(\frac{m}{q} \log n)$ processors available, parallel updates take $o(q)$ time.*

4 Extensions and Optimizations

Improving the Space Complexity. We note here that the space complexity of our encrypted KRB tree is $O(mn)$, where m is the number of the keywords and n is the number of documents. Since $m \ll n$ in practical scenarios, the space complexity can be kept low in practice. However, one way to reduce the size of the data structure is to use a tree of depth 2 with internal nodes of degree $O(\sqrt{n})$. Similarly to KRB trees, one can store m-bit vectors at the internal nodes of such a tree and perform search, add and delete operations in the same manner. Encryption of these trees can also be handled with the same algorithm that encrypts KRB trees. The space complexity of this structure is $O(n + m\sqrt{n})$, which is $O(n)$ as long as $m \leq \sqrt{n}$. Note however that this construction increases the search time and communication complexity (for updates) to $O(r\sqrt{n})$, where r is the number of the documents that contain w.

Supporting I/O-Efficient Search. In the case of very large indexes that cannot fit into main memory[8], our approach can be implemented in an I/O-efficient way by using a B-tree instead of a red-black tree. With such a structure, search always requires $\log_B n$ I/Os. Note that after the disk page has been loaded into main memory, the encrypted search in main memory can be parallelized. In order now to store the B siblings of a B-tree node in a single disk page of x bits, one has to choose B such that $2Bm \leq x$. This is because at each node of the B-tree we need to store two encrypted m-bit vectors, where m is the size of the universe of keywords.

Verifiability of Encrypted Searches. Our model assumes an adversary that is curious but *honest*. However, we note that our scheme can be potentially extended to support verifiability of results in a scenario where the adversary is malicious. This could be achieved by turning our KRB tree into a hash-based KRB tree (e.g., using Merkle hash trees [15]) and maintaining hashes at the internal nodes of the KRB tree. For verifying the result the client could access a logarithmic number of hashes and verify the search computation step-by-step, in logarithmic time (for that, the client also needs to store and update the root hash of the tree). In this way, a client could be assured that no encrypted

[8] E.g., Amazon Common Crawl Corpus, `http://aws.amazon.com/datasets/41740`.

documents have been omitted from the results, unless the adversary is able to break the collision resistance of the hash function. We defer a more formal description of such a scheme and a proof of security to future work.

Acknowledgments. The second author was supported by Intel through the ISTC for Secure Computing. Part of this work was performed while the second author was interning at Microsoft Research. The authors would like to thank Elaine Shi, Dawn Song, Emil Stefanov and Tom Roeder for useful discussions.

References

1. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
2. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
3. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
4. Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 577–594. Springer, Heidelberg (2010)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press (2009)
6. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. In: Computer and Communications Security (CCS), pp. 79–88 (2006)
7. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Symposium on Theory of Computing (STOC), pp. 169–178 (2009)
8. Goh, E.-J.: Secure indexes. IACR Cryptology ePrint Archive, 2003:216 (2003)
9. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. Journal of the ACM 43(3), 431–473 (1996)
10. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
11. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Computer and Communications Security (CCS), pp. 965–976 (2012)
12. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC (2008)
13. Kurosawa, K., Ohtaki, Y.: UC-secure searchable symmetric encryption. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 285–298. Springer, Heidelberg (2012)
14. Lorch, J.R., Mickens, J.W., Parno, B., Raykova, M., Schiffman, J.: Toward practical private access to data centers via parallel ORAM. IACR Cryptology ePrint Archive, 2012:133 (2012)
15. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)
16. Park, D.J., Kim, K., Lee, P.J.: Public key encryption with conjunctive field keyword search. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 73–86. Springer, Heidelberg (2005)
17. Shi, E., Bethencourt, J., Chan, T., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: IEEE Symposium on Security and Privacy (SSP), pp. 350–364 (2007)
18. Song, D., Wagner, D., Perrig, A.: Practical techniques for searching on encrypted data. In: IEEE Symposium on Security and Privacy (SSP), pp. 44–55 (2000)

19. Stefanov, E., Shi, E., Song, D.: Towards practical oblivious ram. In: Network and Distributed System Security Symposium, NDSS (2012)
20. van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 87–100. Springer, Heidelberg (2010)

Appendix

Proof sketch of Lemma 3. We describe a simulator $\mathcal{S}$ that interacts with an adversary $\mathcal{A}$ in an execution of an $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k)$ experiment as described in Definition 5. Given the leakage $\mathcal{L}_1(\delta, \mathbf{f})$, it constructs $(\gamma, \mathbf{c})$ as follows. It simulates the encrypted files $\mathbf{c} = (c_{i_1}, \ldots, c_{i_n})$ using the simulator $\mathcal{S}_{\mathcal{E}}$, which is guaranteed to exist by the CPA-security of $\mathcal{E}$, together with the value n and the size of each file (both of which are included in the leakage function). To simulate γ, it constructs a red black tree T using the identifiers $\mathbf{i} = (i_1, \ldots, i_n)$ included in the leakage function. It then picks m random addresses $\mathtt{addr}_i \in \{0,1\}^k$ and generates m keys $(\mathsf{SK}_1, \ldots, \mathsf{SK}_m)$ for $\mathcal{R}$ to be used for encrypting the entries of the bit-vectors at the internal nodes of T. Note that m is also included in the leakage function.

Now, for every node v of the tree T the simulator sets up two (k, m) keyword hash tables Λ_{0v} and Λ_{1v} as follows: for every $i = 1, \ldots, m$, the simulator stores (a) either an encryption of a "0" at $\Lambda_{0v}[\mathtt{addr}_i]$ and an encryption of a "1" at $\Lambda_{1v}[\mathtt{addr}_i]$ or, (b) an encryption of "1" at $\Lambda_{0v}[\mathtt{addr}_i]$ and an encryption of a "0" at $\Lambda_{1v}[\mathtt{addr}_i]$. This is decided, for every node, by flipping a coin. To encrypt bit $bit_i \in \{0,1\}$ at position $\mathtt{addr}_i$, the simulator uses $\mathcal{R}$, outputting the ciphertext $\mathcal{R}.\mathsf{Enc}(\mathsf{SK}_i, bit_i)$ for all $i = 1, \ldots, m$. The simulator also stores this information locally: For each node v of T it stores a vector $\mathtt{state}_v$ such that $\mathtt{state}_v[i] = j \in \{0,1\}$ if and only if the encryption of "1" was stored at vector Λ_{jv}, for all $i = 1, \ldots, m$.[9] Finally, the simulator outputs the encrypted KRB tree γ to the adversary. γ consists of the red black tree T and the vectors Λ_{0v} and Λ_{1v} for every node v of T. We continue by distinguishing two cases, one for adaptive queries and one for adaptive updates. Before that we recall that given a search query for keyword w that takes place at time t, the leakage $\mathcal{L}_2(\delta, \mathbf{f}, w, t)$ consists of the search pattern $\mathcal{P}(\delta, q, t)$ and the access pattern $\Delta(\delta, \mathbf{f}, w, t)$ (from Definitions 3 and 4):

1. *Adaptive queries*: Suppose the simulator receives a new query q for a keyword w_i. Using $\mathcal{L}_2(\delta, \mathbf{f}, w_i, t)$ the simulator knows whether this query has appeared before (due to the search pattern $\mathcal{P}(\delta, q, t)$), and if so, it outputs the same token τ_s. Else, the simulator picks an address $\mathtt{addr}_i$ that has not been used before. The simulated token is $\tau_s = (\mathtt{addr}_i, \mathsf{SK}_i)$. After the adversary receives the token $\tau_s = (\mathtt{addr}_i, \mathsf{SK}_i) = (\tau_1, \tau_2)$ he is able to execute the search by using the algorithm $\mathsf{Search}(\gamma, \mathbf{c}, \tau_s)$, in a way that Search is accessing *exactly* the same locations contained in the access pattern $\Delta(\delta, \mathbf{f}, w_i, t)$, therefore *always* returning the correct answer. We show, that this can be achieved by appropriate use of the random oracle.

[9] Note that the simulator could also retrieve that information without storing it, by using the initial randomness he used to compute it.

Namely, the output b of the random oracle at Step 1 of the $\mathsf{Search}(\gamma, \mathbf{c}, \tau_s)$ algorithm (with reference to node v) is programmed by the simulator in the following manner: if v is contained in the path to an identifier contained in the access pattern $\Delta(\delta, \mathbf{f}, w_i, t)$, then b is the bit such that $\mathcal{R}.\mathsf{Dec}(\tau_2, \Lambda_{\mathsf{b}v}[\tau_1]) = 1$. Otherwise, b is the bit such that $\mathcal{R}.\mathsf{Dec}(\tau_2, \Lambda_{\mathsf{b}v}[\tau_1]) = 0$. Note that the appropriate bit can be determined by the simulator, since it is storing the state vector $\mathtt{state}_v$, for every node v of the tree T.

2. *Adaptive updates*: Suppose the simulator receives a new query $q = u = f_i$ for inserting/deleting document f_i with identifier i. If u is an insertion, the simulator uses $\mathcal{S}_{\mathcal{E}}$ to simulate c_i (note that while f_i is not available to the simulator, its identifier is revealed through the updated leakage $\mathcal{L}_1(\delta, \mathbf{f})$). Now let $\mathsf{info}_{i,u}$ be the helper information that is returned during the protocol by $\mathsf{UpdHelper}$. We recall that $\mathsf{info}_{i,u}$ consists of a certain subtree $T(u)$ of T, namely the portion of the red-black tree T that is *accessed* during the update. The simulator can now compute $T'(u)$ (since he knows the identifier of the new document) which is the new subtree after the update. For every node v' that changes its structure in the subtree $T'(u)$ (e.g., it obtains a new child), and has a new identifier $\mathtt{id}(v')$, the simulator does the following (similar actions taken in the setup phase of the simulation):
   (a) it instantiates two (k, m) keyword hash tables with random entries $\Lambda_{0v'}$ and $\Lambda_{1v'}$ and stores $\lambda_{0v'}$ and $\lambda_{1v'}$ at v;
   (b) For every $i = 1, \ldots, m$, the simulator *reinitializes* the keyword hash tables $\Lambda_{0v'}$ and $\Lambda_{1v'}$ by storing (a) either an encryption of a "0" at $\Lambda_{0v'}[\mathtt{addr}_i]$ and an encryption of an "1" at $\Lambda_{1v'}[\mathtt{addr}_i]$ or, (b) an encryption of "1" at $\Lambda_{0v'}[\mathtt{addr}_i]$ and an encryption of a "0" at $\Lambda_{1v'}[\mathtt{addr}_i]$. This is decided by flipping a coin. Again, to encrypt bit $b_i \in \{0, 1\}$ at position $\mathtt{addr}_i$, the simulator executes $\mathcal{R}.\mathsf{Enc}(\mathsf{SK}_i, b_i)$. Also the simulator updates its state $\mathtt{state}_{v'}$ accordingly.
   (c) Output $\tau_u := (T'(u), c_i)$.

After the update, if the adversary searches for keywords for which it has old tokens, the simulator can again control the search by programming the random oracle appropriately since it gets the updated leakage $\Delta(\delta, \mathbf{f}, w_i, t)$ for each previous keyword for which a token was issued at time t.

It remains to show that for all PPT adversaries $\mathcal{A}$, the outputs of a $\mathbf{Real}_{\mathcal{A}}(k)$ and of an $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(k)$ experiment are negligibly close. This holds for the following reasons. The keys used in the tokens and the ones used to encrypt the hash table elements are indistinguishable from real keys since they are constructed with PRFs which are indistinguishable from random functions. Therefore, the CPA-security of $\mathcal{E}$ and $\mathcal{R}$ guarantee that the adversary cannot distinguish between the real and simulated encryptions of the files and bit vectors, respectively. Finally, the answers returned by simulator to the adversary's random oracle queries are consistent and are appropriately distributed. □

GMW vs. Yao? Efficient Secure Two-Party Computation with Low Depth Circuits

Thomas Schneider and Michael Zohner

Engineering Cryptographic Protocols Group (ENCRYPTO),
European Center for Security and Privacy by Design (EC SPRIDE),
Technische Universität Darmstadt, Germany
{thomas.schneider,michael.zohner}@ec-spride.de

Abstract. Secure two-party computation is a rapidly emerging field of research and enables a large variety of privacy-preserving applications such as mobile social networks or biometric identification. In the late eighties, two different approaches were proposed: Yao's garbled circuits and the protocol of Goldreich-Micali-Wigderson (GMW). Since then, research has mostly focused on Yao's garbled circuits as they were believed to yield better efficiency due to their constant round complexity.

In this work we give several optimizations for an efficient implementation of the GMW protocol. We show that for semi-honest adversaries the optimized GMW protocol can outperform today's most efficient implementations of Yao's garbled circuits, but highly depends on a low network latency. As a first step to overcome these latency issues, we summarize depth-optimized circuit constructions for various standard tasks. As application scenario we consider privacy-preserving face recognition and show that our optimized framework is up to 100 times faster than previous works even in settings with high network latency.

Keywords: GMW protocol, optimizations, privacy-preserving face recognition.

1 Introduction

Generic secure two-party computation allows two parties to jointly compute any function on their private inputs without revealing anything but the result. Interestingly, two different approaches have been introduced at about the same time in the late eighties: Yao's garbled circuits [33] and the protocol of Goldreich-Micali-Wigderson (GMW) [11, 12]. Both protocols allow secure evaluation of a function that is represented as a Boolean circuit and, in their basic form, provide security against semi-honest adversaries who honestly follow the protocol but try to learn additional information from the observed messages – this widely used model allows to construct highly efficient protocols and is the focus of our paper.

As Yao's protocol has a constant number of rounds and requires *Oblivious Transfers (OTs)* only for the inputs of one of the parties while the secure evaluation of a gate requires only symmetric cryptographic operations, it was believed

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 275–292, 2013.

to be more efficient than the GMW protocol, which requires an interactive OT for each AND gate (see for example [13, Sect. 1.2]). In fact, many subsequent works presented improvements to Yao's protocol and showed that it can be made truly practical and applied to a large variety of privacy-preserving applications, e.g., [14, 15, 19, 20, 24, 29].

In a recent work [5], it was shown that by implementing OTs efficiently using symmetric cryptographic primitives, the semi-honest version of the GMW protocol can outperform previous secure computation protocols with $n \geq 3$ parties. However, for the two-party case, the authors of [5] state that they expect their implementation to be roughly a factor of two slower than today's fastest implementation of Yao's garbled circuit protocol of [15]. For stronger active adversaries, it has been shown in [27] that an extension of the GMW protocol achieves a performance that can compete with garbled circuit-based protocols.

1.1 Outline and Our Contributions

In this work we show that the GMW protocol is truly practical in the setting with two semi-honest parties and in fact has some advantages over Yao's garbled circuits protocol. Our contribution is threefold:

Depth-Efficient Circuits. As the GMW protocol requires interaction for the secure evaluation of AND gates, the dependence on the latency can be reduced by using circuit constructions that have both, low size and depth. We give a summary of such circuit constructions for standard functionalities in §3.

Implementation-Specific Optimizations. We extend the implementation of the GMW protocol of [5] with various optimizations to yield better performance in the setting with two parties. Our optimizations are described in §4 and include load balancing (to distribute the workload equally among the parties) and processing multiple gates in parallel. Overall, our optimized implementation yields a more than 10 times faster runtime compared to the implementation of [5] for an example circuit with about 800,000 gates.

Performance Evaluation. Finally, in §5, we compare the optimized GMW protocol with today's most efficient techniques for garbled circuits on a conceptual level (independent of the implementation). Afterwards, we experimentally measure and evaluate the performance of our implementation for two example applications: a mobile social network [5] and privacy-preserving face recognition using Eigenfaces [9, 14, 30] and the Hamming distance [28] in a desktop environment that has a network with high bandwidth. In settings with low network latency our implementation is able to process up to 1,000,000 AND gates (1-out-of-4 OTs) per second in the setup phase and 50,000,000 AND gates per second in the online phase. Our evaluation indicates that the GMW protocol is a noticeable alternative to previous protocols based on garbled circuits.

1.2 Related Work

Circuit Optimizations. Since the communication complexity of garbled circuits is independent of the depth of the evaluated circuit, cryptographic literature

mostly focused on minimizing the multiplicative size of circuits, i.e., the number of AND gates where XOR gates are free, e.g. [4,18,29]. Most of the existing work on minimizing circuit depth originates from the area of electrical engineering (cf. [8,31,32]), where the circuit depth directly affects the computation latency. Our setting for circuit optimization is slightly different from this as we can evaluate XOR gates for free, can store intermediate results indefinitely, and have unbounded fan-out.

Garbled Circuit Frameworks. Starting with Fairplay [24] that demonstrated the practical feasibility of Yao's garbled circuits, various frameworks for secure two-party computation have been developed. The currently fastest garbled circuits framework with security in the semi-honest model is [15]. A garbled circuit framework secure against malicious adversaries was given in [20].

GMW Frameworks. The implementation of [5] showed that for semi-honest adversaries the GMW protocol can outperform previous secure multi-party computation frameworks for circuits that can be efficiently represented as Boolean circuits. However, they show this only for the setting with $n \geq 3$ parties and expect that their implementation requires about twice the computation time of [15] in the two-party setting (cf. [5, Sect. 1.2]). The GMW protocol was recently extended into a new approach for secure two-party computation with security against malicious adversaries in [27].

Other Frameworks. For completeness we mention that there exist also other approaches for secure two-party computation (beyond garbled circuits and the GMW protocol) that evaluate arithmetic instead of Boolean circuits. These approaches use additively homomorphic encryption as implemented in the VIFF framework [6] or the SPDZ framework [7] and can even be combined with garbled circuits as implemented in [14]. Although these approaches are well-suited for outsourcing computations in scenarios where one party has substantially more computing power than the other party (e.g., cloud computing), they involve relatively expensive public-key operations in the online phase. Hence, these works are orthogonal to the protocols we consider that require only fast operations per gate and allow to distribute the workload equally among the parties.

2 Preliminaries

2.1 Oblivious Transfer

Oblivious Transfer (OT) is a cryptographic protocol executed between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$ in which $\mathcal{R}$ obliviously selects one of the inputs provided by $\mathcal{S}$. More specifically, in 1-out-of-n OT_ℓ^m, $\mathcal{S}$ provides m n-tuples $(x_{11}, \ldots, x_{1n}), \ldots, (x_{m1}, \ldots, x_{mn})$ of ℓ-bit strings; $\mathcal{R}$ provides m selection numbers $r_1, \ldots, r_m$ with $1 \leq r_i \leq n$ and obtains x_{jr_j} $(1 \leq j \leq m)$ as output. The widely used Naor-Pinkas OT protocol [25] is secure against semi-honest adversaries under the Decisional Diffie-Hellman (DDH) assumption in the random-oracle model and requires both parties to perform $\mathcal{O}(m)$ modular exponentiations. The following two techniques can be used to substantially speed up OTs.

OT pre-computations [2] allows to pre-compute the OTs on random inputs and later on in the online phase use these pre-computed values as one-time pads to run the OT on the actual inputs. In the online phase, $\mathcal{R}$ sends one message of size $m \log_2 n$ bits to $\mathcal{S}$ who sends back a message of size mnl bits.

OT extensions [16,22] allow to perform a large number of m OTs (OT_ℓ^m) using a small number of t base OTs on t-bit keys (OT_t^t), where t is a security parameter. The marginal cost for each additional OT is a small number of evaluations of a cryptographic hash function (modeled as random oracle) and of a pseudo-random function. More specifically, for each of the m OTs, $\mathcal{S}$ computes n hash evaluations and $t(1 + \log_2 n)$ pseudo-random bits, whereas $\mathcal{R}$ computes 1 hash evaluation and $t(1 + n)$ pseudo-random bits. The communication complexity is one message from $\mathcal{R}$ to $\mathcal{S}$ of size mnt bits and one in the opposite direction of size $mn\ell$ bits.

2.2 Approaches for Secure Two-Party Computation

We summarize the main approaches for secure two-party computation of Boolean circuits next: Yao's garbled circuits (§2.2.1) and the GMW protocol (§2.2.2).

2.2.1 Yao's Garbled Circuits Protocol [33]. The basic idea of Yao's garbled circuits is to let one party, called creator, encrypt the function to be computed. For this, the plain values are mapped to random-looking symmetric keys and for each gate an encryption table is generated that allows to compute the gate's output key given its input keys. The creator then transmits the encrypted circuit along with the corresponding encrypted inputs to the other party, called evaluator. The creator sends his encrypted inputs directly to the evaluator and the evaluator obtains his encrypted inputs obliviously via 1-out-of-2 OT. The evaluator then uses the encrypted inputs to evaluate the encrypted function gate by gate. Finally, the creator provides a mapping from the encrypted output to plain output. As the evaluation of Yao's garbled circuits is performed non-interactively, the resulting protocol has a constant number of rounds.

The following extensions enhance the speed of Yao's garbled circuits protocol: point-and-permute [24], free XOR [19], efficient encryption with a cryptographic hash function [23], garbled row reduction [26, 29], and pipelining [15]. Put together, these techniques allow "free" evaluation of XOR gates (i.e., no communication and negligible computation), interweaving circuit generation and evaluation, and per non-XOR gate 4 evaluations of a cryptographic hash function for the creator, transmissions of an encrypted gate table with 3 entries, and 1 evaluation of a cryptographic hash function for the evaluator.

2.2.2 GMW Protocol [11, 12]. In the GMW protocol two parties interactively compute a function using secret-shared values. For this, the value v of each input and intermediate wire is shared among the two parties with a 2-out-of-2 secret sharing scheme such that each party holds a random-looking share v_i with $v = v_1 \oplus v_2$. As XOR is an associative operation, XOR gates can be securely evaluated locally by XORing the shares. For secure evaluation of AND gates,

the parties run an interactive protocol using one of the two techniques described below. We note that AND gates of the same layer in the circuit can be computed in parallel. Finally, the parties send the respective shares of the output wires to the party that should obtain the output.

Oblivious Transfers. To securely evaluate an AND gate on input shares x_1, x_2 and y_1, y_2, the two parties can run a 1-out-of-4 OT_1^1 protocol. Here, the chooser inputs its shares x_1, y_1 and the sender chooses a random output share z_2 and provides four inputs to the OT protocol such that the chooser obliviously obtains its output share $z_1 = z_2 \oplus ((x_1 \oplus x_2) \wedge (y_1 \oplus y_2))$. As described in §2.1, all OTs can be moved into a pre-processing phase such that the online phase is highly efficient (only two messages and inexpensive one-time-pad operations).

Multiplication Triples. An alternative method to securely evaluate an AND gate on input shares x_1, x_2 and y_1, y_2 are *multiplication triples* [1]. Multiplication triples are random shares a_i, b_i, c_i satisfying $(c_1 \oplus c_2) = (a_1 \oplus a_2) \wedge (b_1 \oplus b_2)$. They can be generated in the setup phase using a 1-out-of-4 OT_1^1 protocol in a similar way to the OT-based solution described above. In the online phase the parties use these pre-generated multiplication triples to mask the input shares of the AND gate, exchange $d_i = x_i \oplus a_i$ and $e_i = y_i \oplus b_i$, and compute $d = d_1 \oplus d_2$ and $e = e_1 \oplus e_2$. The output shares are computed as $z_1 = (d \wedge e) \oplus (b_1 \wedge d) \oplus (a_1 \wedge e) \oplus c_1$ and $z_2 = (b_2 \wedge d) \oplus (a_2 \wedge e) \oplus c_2$. The advantage of multiplication triples over the OT-based solution is that, per AND gate, each party needs to send only one message (independent of each other) and the size of the messages is slightly smaller ($2 + 2$ bits instead of $2 + 4$ bits).

2.3 Evaluation Metrics and Notation

Motivated by the fact that both approaches for secure two-party computation summarized in §2.2 provide free XORs, we consider the *(multiplicative) size* $\mathbf{S}(\text{C})$ of a circuit C as the number of AND gates in C and the *(multiplicative) depth* $\mathbf{D}(\text{C})$ as the maximum number of AND gates on any path from an input to an output of C.

We denote a bit sequence of length ℓ bits as x^ℓ and use x_i to refer to its i-th bit, starting with the least-significant bit x_1. $\bar{x}^\ell$ denotes the bitwise complement of x^ℓ. $x^\ell_{1...n}$ is a sequence of n ℓ-bit values $x^\ell_1, x^\ell_2, ..., x^\ell_n$. A circuit C that processes n values of $(\ell_1, \ell_2, ..., \ell_n)$-bits each is denoted by $\text{C}^{((\ell_1,\ell_2,...,\ell_n),n)}$. If all inputs n have the same length ℓ, we shorten the notation to $\text{C}^{(\ell,n)}$. When n is clear from the context we use C^ℓ instead. We use the standard notations for binary operations, i.e. concatenation $||$, bitwise XOR $\oplus$, bitwise AND $\wedge$, and bitwise OR $\vee$.

3 Circuit Constructions with low Depth and Size

In the following we briefly summarize circuit building blocks that have low depth and only a slightly larger size compared to constructions that are optimized for size only. A detailed list of constructions is given in Appendix A, Tab. 7 (due to space restrictions we defer details to the full version).

3.1 Addition

The standard method for adding two ℓ-bit numbers is the *Ripple-carry* adder ADD_{RC}^{ℓ} that has linear size and depth, $\mathbf{S}(\text{ADD}_{RC}^{\ell})$=$\mathbf{D}(\text{ADD}_{RC}^{\ell}) = \ell$ [18]. In the following we summarize techniques for addition in sub-linear depth.

3.1.1 Ladner-Fischer Adder. The *Ladner-Fischer* adder ADD_{LF}^{ℓ} [21] is a so-called parallel prefix adder that adds two ℓ-bit values x^{ℓ} and y^{ℓ} in logarithmic depth. The idea of parallel prefix adders is to evaluate multiple carry-bits in parallel. During the computation of the sum, the Ladner-Fischer adder computes a parity bit $p_{i,j}$ and a carry bit $c_{i,j}$ in each node at bit position i ($1 \leq i \leq \ell$) and level j ($0 \leq j \leq \lceil \log_2 \ell \rceil$). At level 0, the Ladner-Fischer adder computes $p_{i,0} = x_i \oplus y_i$ and $c_{i,0} = x_i \wedge y_i$. Then, for every node at level $j > 0$, the parity and carry bit are computed as $p_{i,j} = p_{i,j-1} \wedge p_{k,j-1}$ and $c_{i,j} = (p_{i,j-1} \wedge c_{k,j-1}) \vee c_{i,j}$, where k is the node that propagates the carry-bit to position i. Lastly, at level $\lceil \log_2 \ell \rceil + 1$, the sum $s^{\ell+1}$ is computed as $s_{\ell+1} = c_{\ell,\lceil \log_2 \ell \rceil}$, $s_i = p_{i,0} \oplus c_{i-1,\lceil \log_2 (i-1) \rceil}$ for $1 < i \leq \ell$, and $s_1 = p_{1,0}$. The Ladner-Fischer adder has size $\mathbf{S}(\text{ADD}_{LF}^{\ell}) = 1.25\ell\lceil \log_2 \ell \rceil + \ell$ and depth $\mathbf{D}(\text{ADD}_{LF}^{\ell}) = 2\lceil \log_2 \ell \rceil + 1$.

3.1.2 Carry-Save Adder. The *carry-save* adder $\text{ADD}_{CSA}^{(\ell,3)}$ [8] converts the sum of three ℓ-bit unsigned integers x^{ℓ}, y^{ℓ}, and z^{ℓ} into two $\ell+1$-bit unsigned integers $p^{\ell+1}$ and $c^{\ell+1}$ such that $p^{\ell+1} + c^{\ell+1} = x^{\ell} + y^{\ell} + z^{\ell}$. To obtain the result of the addition in sub-linear depth, $p^{\ell+1}$ and $c^{\ell+1}$ can again be added using the Ladner-Fischer adder (cf. §3.1.1). The carry-save adder is composed from ℓ 1-bit full-adders that compute the parity $p_i = x_i \oplus y_i \oplus z_i$ and the carry $c_{i+1} = z_i \oplus ((z_i \oplus x_i) \wedge (z_i \oplus y_i))$ in parallel for every bit position $1 \leq i \leq \ell$. Finally, $p_{\ell+1}$ and c_1 are set to 0.

The carry-save adder has linear size and constant depth, allowing three ℓ-bit numbers to be added with a circuit of size $\mathbf{S}(\text{ADD}_{CSA}^{(\ell,3)}) = \ell$+$\mathbf{S}(\text{ADD}_{LF}^{(\ell+1)})$ and depth $\mathbf{D}(\text{ADD}_{CSA}^{(\ell,3)}) = 1$+$\mathbf{D}(\text{ADD}_{LF}^{(\ell+1)})$. Multiple carry-save adders can be combined to create a carry-save network $\text{ADD}_{CSN}^{(\ell,n)}$ that converts n ℓ-bit numbers to two ℓ+$\lceil \log_2 n \rceil$-bit numbers and adds them using $\text{ADD}_{LF}^{\ell+\lceil \log_2 n \rceil}$ with $\mathbf{S}(\text{ADD}_{CSN}^{(\ell,n)})$ $= (\ell-1)(n-2) + \sum_{i=0}^{\lceil \log_2 n \rceil} i\frac{n}{2^i} + \mathbf{S}(\text{ADD}_{LF}^{(\ell+\lceil \log_2 n \rceil)}) \approx \ell n - 2\ell + n - \lceil \log_2 n \rceil +$ $\mathbf{S}(\text{ADD}_{LF}^{(\ell+\lceil \log_2 n \rceil)})$ and $\mathbf{D}(\text{ADD}_{CSN}^{(\ell,n)}) = \lceil \log_2 n \rceil + \mathbf{D}(\text{ADD}_{LF}^{(\ell+\lceil \log_2 n \rceil)})$ [31].

3.2 Squaring

Although the square of a number can be computed with a multiplication circuit, a squaring circuit is smaller by a factor of about two. The standard school method multiplication circuit MUL^{ℓ} [18] computes the product $x^{\ell}x^{\ell}$ as $\sum_{i=1}^{\ell} 2^{i-1}(x^{\ell}x_i)$. Since each x_jx_i with $i \neq j$ is computed twice, $x_jx_i + x_ix_j$ can be simplified to $2x_ix_j$ and x_ix_i can be replaced with x_i [34]. The corresponding depth-efficient squarer circuit SQR_{LF}^{ℓ} has size $\mathbf{S}(\text{SQR}_{LF}^{\ell}) = \ell^2 + 1.25\ell\lceil \log_2 \ell \rceil - 1.5\ell - 2$ and depth $\mathbf{D}(\text{SQR}_{LF}^{\ell})$=$\mathbf{D}(\text{ADD}_{CSA}^{(2\ell,\lceil \ell/2 \rceil)})$+$\mathbf{D}(\text{ADD}_{LF}^{2\ell})$+1 $= 3\lceil \log_2 \ell \rceil + 3$.

3.3 Comparison

A circuit that checks the equality of two values EQ^ℓ has linear size $\mathbf{S}(\mathrm{EQ}^\ell) = \ell - 1$ [19] and can be built in a pairwise tournament fashion to achieve logarithmic depth $\mathbf{D}(\mathrm{EQ}^\ell) = \lceil \log_2 \ell \rceil$. The standard greater than circuit GT^ℓ_S that checks whether one number is greater than another has linear size, but also linear depth $\mathbf{S}(\mathrm{GT}^\ell_S) = \mathbf{D}(\mathrm{GT}^\ell_S) = \ell$ [18]. The greater than operation can be computed in logarithmic depth using the circuit GT^ℓ_{DC} of [10] that recursively splits the ℓ-bit parameters into half. More precisely, let $x^\ell = (x_H || x_L)$ and $y^\ell = (y_H || y_L)$ be two ℓ-bit integers with x_H, y_H being $\lceil \frac{\ell}{2} \rceil$-bit and x_L, y_L being $\lfloor \frac{\ell}{2} \rfloor$-bit unsigned integers. GT^ℓ_{DC} is then recursively computed as $\mathrm{GT}^\ell_{DC}(x^\ell, y^\ell) = \mathrm{GT}^{\lceil \frac{\ell}{2} \rceil}_{DC}(x_H, y_H) \oplus \mathrm{EQ}^{\lceil \frac{\ell}{2} \rceil}(x_H, y_H) \wedge \mathrm{GT}^{\lfloor \frac{\ell}{2} \rfloor}_{DC}(x_L, y_L)$ until x and y are bits for which $\mathrm{GT}^1_{DC}(x_i, y_i) = x_i \wedge \bar{y}_i$. The circuit has size $\mathbf{S}(\mathrm{GT}^\ell_{DC}) = 3\ell - \lceil \log_2 \ell \rceil - 2$ and depth $\mathbf{D}(\mathrm{GT}^\ell_{DC}) = \lceil \log_2 \ell \rceil + 1$.

3.4 Hamming Weight

The Hamming weight circuit CNT^ℓ counts the number of one entries in x^ℓ, i.e., it computes its Hamming weight $d_H(x^\ell) = \sum_{i=1}^{\ell} x_i$. In [4], a Hamming weight circuit CNT^ℓ_{BP} was given that splits a value x^ℓ into three parts of length $m = \lceil \frac{\ell - 1}{2} \rceil$, $n = \lfloor \frac{\ell - 1}{2} \rfloor$, and one bit, respectively: $x^\ell = (x^m || x^n || x_1)$. CNT^ℓ_{BP} is then computed recursively as $\mathrm{CNT}^\ell_{BP}(x^\ell) = \mathrm{ADD}^{\lceil \log_2 \ell \rceil}_{RC}(\mathrm{CNT}^m_{BP}(x^m), \mathrm{CNT}^n_{BP}(x^n), x_1)$ where x_1 can be provided as carry-in to the addition circuit at no extra cost. The circuit has size $\mathbf{S}(\mathrm{CNT}^\ell_{BP}) = \ell - d_H(\ell)$ and depth $\mathbf{D}(\mathrm{CNT}^\ell_{BP}) = \lfloor \log_2 \ell \rfloor$.

4 Optimizations for Two-Party GMW

The original paper of [5] provides an implementation of the GMW protocol and gives performance numbers for $n \geq 3$ parties. For two parties, they expect that their implementation is slower than the currently fastest garbled circuit implementation of [15] by a factor of two (cf. [5, Sect. 1.2]). We modified and extended their GMW implementation for better efficiency in the two-party setting. In the following we give an overview of our modifications that improved the overall performance the most. We list the modifications in the order they were implemented and summarize the performance numbers in Tab. 1. Hence, the performance numbers for each modification include the improvement of all previous optimizations. In total, we improve the overall runtime of the following example application by more than a factor of 10.

Benchmarking Environment. In our experiments we evaluate the time for the *setup phase* (Naor-Pinkas OTs and OT extension), *online phase* (sharing of inputs, circuit evaluation, and combining output shares), and the *overall time* (circuit construction, setup phase, and online phase). We perform the comparison on an unoptimized 512-bit multiplication circuit C with $\mathbf{S}(\mathrm{C}) = 800{,}227$ and

$\mathbf{D}(\mathrm{C}) = 38$ using the average time of 100 executions. For the Naor-Pinkas OT we use the group $\mathbb{Z}_p^*$ with $|p| = 512$ bit, also used in [5]. For the OT extensions we set the security parameter to $t = 80$. The server and client run on two 2.5 GHz Intel Core2Quad CPU (Q8300) Desktop PCs with 4 GB RAM each that are connected via Gigabit LAN with a ping latency of 0.2 ms.

Table 1. Time improvements for the individual optimizations

Optimization	None [5]	MT §4.1	PRF §4.2	LB §4.3	GMP §4.4	Bytewise §4.4	SHA-1 §4.4	SIMD §4.5
Setup Phase [s]	13.39	13.82	11.41	7.41	6.87	1.68	0.89	0.84
Online Phase [s]	0.73	0.70	0.70	0.71	0.70	0.31	0.32	0.012
Overall Time [s]	14.19	14.52	12.16	8.14	7.60	2.10	1.35	0.85

4.1 Multiplication Triples

To reduce the impact of the latency during the online time we use Beaver's multiplication triples [1] instead of pre-computed OTs for secure evaluation of AND gates (cf. §2.2.2). This results in a slightly slower setup phase (13.82 s instead of 13.39 s) due to the additional overhead for computing c_2 (instead of using random inputs). However, the online phase gets more efficient (0.70 s instead of 0.73 s) as we only require one instead of two interaction steps and 4 instead of 6 bits sent per AND gate.

4.2 Using AES Instead of SHA as Pseudo-Random Function

The original implementation of [5] used SHA-1 not only for instantiating the random oracle, but also for generating pseudo-random values in the OT extension. To reduce the computational complexity, we use AES in the counter mode as pseudo-random function (PRF). More precisely, we benchmarked the SHA-1 implementation of PolarSSL that was used in [5] against the SHA-1 implementation (sha1-x86_64) and the AES128 (aes-x86_64) implementation of OpenSSL v. 1.0.1c. The results for 10^9 iterations are depicted in Tab. 2. This decreased the number of expensive hash function calls per AND gate (to instantiate the random oracle) from 3.5 to 1 for the receiver $\mathcal{R}$ and from 4.5 to 4 for the sender $\mathcal{S}$; additionally, $\mathcal{R}$ performs 3.1 AES calls and $\mathcal{S}$ performs 0.65 AES calls per AND gate (to instantiate the PRF). Overall, using AES as PRF decreased the setup time from 13.82 s to 11.41 s. We note that the performance could be improved even further by using the Intel AES New Instructions (AES-NI) provided by recent CPUs.

4.3 Load Balancing

In the OT extension protocol executed in the setup phase, the sender and the receiver have different computational workload (cf. §4.2). As the multiplication

Table 2. Comparison of SHA-1 and AES128 implementations for 10^9 iterations

Algorithm	Iterations/s	Bits/s
SHA-1 PolarSSL	$1.30 \cdot 10^6$	$2.08 \cdot 10^8$
SHA-1 OpenSSL	$3.65 \cdot 10^6$	$5.83 \cdot 10^8$
AES128 OpenSSL	$4.99 \cdot 10^6$	$6.39 \cdot 10^8$

triples used in the online phase (cf. §4.1) are symmetric, we can run the OTs to generate them in the setup phase in either direction. Hence, to balance the workload, we run two instantiations of the OT protocol (each for half of the AND gates) in parallel with the roles reversed. With this optimization, each party has the same workload: 2.5 SHA-1 invocations and 1.8 AES invocations per AND gate. Note that now we also need to run the Naor-Pinkas OT protocol for the seed OTs twice, which however amortizes fairly quickly (35 ms computation time and 10 kByte to be transferred). Since the original implementation of [5] already used multi-threading for implementing the OT extensions, we now use four parallel threads during the setup phase (two for each role such that one thread evaluates the pseudo-random function in the first round and the other the random oracle in the second round of the protocol). Overall, load balancing decreased the setup time from 11.41 s to 7.41 s.

4.4 Implementation-Specific Optimizations

In order to further speed up the execution time we performed several implementation-specific optimizations as summarized next.

Arithmetic. For modular arithmetics within the Naor-Pinkas base OTs we replaced the Number Theory Library (*NTL*) v. 5.5.2 used in [5] with the GNU Multiple Precision Arithmetic Library (*GMP*) v. 5.0.5. This decreased the time for the Naor-Pinkas base OTs from 590 ms using NTL to 35 ms using GMP for modular operations on 512 bit values.

Bytewise Operations. A major bottleneck was the bitwise processing order during the OT extension step and the online phase. We reduced the impact of the processing order by performing the operations bytewise instead of bitwise. For the setup phase we thereby gained a decrease in time from 6.87 s to 1.68 s. For the online phase we achieved a decrease in time from 0.70 s to 0.31 s.

SHA-1. Afterwards, the evaluation of SHA-1 for the random oracle became the major bottleneck for the OT extension. We replaced the implementation of SHA-1 from PolarSSL used in [5] with an assembler implementation of OpenSSL v. 1.0.1c (sha1-x86_64). This decreased the setup time from 1.68 s to 0.89 s.

4.5 Single Instruction Multiple Data (SIMD) Operations

The Sharemind framework [3] for secure three-party computation showed that Single Instructions Multiple Data (SIMD) operations can result in substantial

performance gains. The idea of SIMD operations is to replace the evaluation of n identically copies of the same sub-circuit on one-bit values by one evaluation of the sub-circuit on n-bit values. This optimization reduces the overall computation time and the memory footprint as the circuit needs to be generated only once. SIMD operations are especially beneficial in data mining applications [3].

We show the benefit of SIMD operations by running the benchmarking circuit C 32 times in parallel on different inputs, resulting in a circuit C_{par} with $\mathbf{S}(C_{par}) = 32 \cdot \mathbf{S}(C) = 25{,}607{,}264$ and $\mathbf{D}(C_{par}) = \mathbf{D}(C) = 38$. The evaluation without SIMD operations required 36.44 s (i.e., amortized 1.13 s for each circuit C), of which 26.76 s (0.84 s) were spent in the setup phase and 2.87 s (0.09 s) in the online phase. Using the SIMD operations, the overall time decreased to 27.34 s (0.85 s), with similar setup time of 26.74 s (0.82 s) but 7.5 times faster online time of 0.38 s (0.012 s). In our implementation the RAM requirement of one gate without the SIMD extension is 14 Byte. C has 3,168,202 gates (including XOR gates) in total and requires 42.3 MByte of memory. To store C_{par} without the SIMD extension we therefore would need 1.32 GByte of memory. The SIMD extension increases the size per gate to 25 Byte plus one bit for each parallel exection and adds a negligible management overhead for the conversion between bitwise and SIMD operations. In total, the SIMD circuit for C_{par} requires only 88.6 MByte of memory, which corresponds to 6.5% of the non-SIMD variant.

5 Evaluation

We consider two application scenarios to evaluate the benefits of depth-optimized circuits (§3) and the optimizations of the GMW protocol (§4): privacy-preserving mobile social networks (§5.1) and privacy preserving face-recognition (§5.2).

Conceptual Performance Comparison. We first give a conceptual comparison between the GMW protocol, optimized as described in §4, with today's most efficient techniques for garbled circuits, as summarized in Tab. 3. Both techniques provide free XOR gates. Unlike garbled circuits, for each AND gate, the GMW protocol allows to shift *all* usage of symmetric cryptography into the setup phase (cf. §2.2.2) and to distribute the workload *equally* among client C and server S (cf. §4.3). Since GMW operates on single bits during the online phase, multiple gates can be evaluated in one instruction (cf. §4.4 and §4.5), whereas garbled circuits need to evaluate each gate individually. However, the total communication per AND gate for the GMW protocol is slightly larger. The memory requirement during secure evaluation of the circuit is smaller for the GMW protocol as for every wire of the circuit each party only needs to store a 1-bit share instead of an 80-bit wire label. Also the inputs are cheaper in the GMW protocol as only one random bit needs to be chosen and sent to the other party, whereas in garbled circuits a random 80-bit key needs to be chosen and sent to the evaluator (using OT for evaluator's inputs). The main disadvantage of the GMW protocol is its need for interaction during evaluation of AND gates which becomes an inevitable performance bottleneck for high network latency as shown in our practical experiments next.

Table 3. Conceptual Comparison between state-of-the-art Garbled Circuits (using Point-and-Permute, Free XORs, Garbled Row Reduction, and Pipelining) and the GMW Protocol (using optimizations of §4) for security parameter $t = 80$ bits. C: client, S: server, SHA: SHA-1, AES: AES128.

Properties	Garbled Circuits	Optimized GMW
Free XOR	yes	yes
per AND gate:		
setup computation	-	C&S: 2.5×SHA+1.8×AES
setup communication [bit]	-	C→S&C←S: 162
online computation	C: 1×SHA; S: 4×SHA	negligible
online communication [bit]	C←S: 240	C→S&C←S: 2
per wire storage C&S [bit]	80	1
per input: rnd bits | comm. [bit]	80 | C: OT resp. S: 80	1 | 1

Benchmarking Environment. We measure runtimes for different (round-trip) latencies that are typical for the following network types: LAN (0.2 ms), intra-country internet (10 ms), and trans-atlantic internet (100 ms). The latency on the network interfaces was enforced by changing the traffic control settings on Linux using the `tc` command. We used the same benchmarking environment as described in §4, but use a subgroup of order q with $|q| = 128$ and $|p| = 1024$ in the Naor-Pinkas OT to match the parameters of previous works. The time for the Naor-Pinkas OT on 1024 bit values in the LAN setting is 2,950 ms for the original implementation of [5] and 170 ms for our implementation.

5.1 Mobile Social Networks

In the mobile social network scenario of [5] one party inputs a database of N users U_i $(1 \leq i \leq N)$ that each have a set of interests H_i (represented as ℓ-bit vector) and a m-bit location L_i. The other party is a user U who can identify the user U_k that shares the most interests among all users U_i within a certain distance δ^{m+1}. As a result, U obtains in a privacy-preserving way only the identity k of U_k and 0 otherwise. We give more details on the circuit in §C.

In our experiments we use $N = 128$ users, $m = 32$-bit locations L_i, and $\ell = 255$-bit sets of interests H_i. We build two versions of the circuit, a size-efficient version MSN_S with $\mathbf{S}(\text{MSN}_S) = 89{,}321$ and $\mathbf{D}(\text{MSN}_S) = 98$ and a depth-efficient version MSN_D with $\mathbf{S}(\text{MSN}_D) = 203{,}102$ and $\mathbf{D}(\text{MSN}_D) = 68$. We compare the performance on both circuits for the original implementation of [5] to our implementation in Tab. 4.

For the LAN setting (0.2 ms) our implementation outperforms the original implementation by factor 13 (factor 3 in the online phase). Due to the multiplication triple optimization (cf. §4.1), the online time of our implementation increases only half as much as the original implementation with rising latency. While the online time of MSN_D is marginally higher than MSN_S in the LAN setting (0.2 ms), the latency becomes the dominant factor in the overall execution time for the

Table 4. Runtimes for secure evaluation of mobile social networks

Framework	[5]						This work					
Circuit	MSN_S			MSN_D			MSN_S			MSN_D		
Latency [ms]	0.2	10	100	0.2	10	100	0.2	10	100	0.2	10	100
Setup phase [s]	4.65	4.82	6.27	6.48	6.62	8.24	0.30	0.51	1.62	0.42	0.62	1.92
Online phase [s]	0.15	1.12	9.94	0.23	0.92	7.04	0.05	0.54	4.92	0.06	0.40	3.55
Overall time [s]	4.83	5.97	16.24	6.73	7.57	15.31	0.36	1.06	6.58	0.50	1.03	5.50

intra-country internet setting (10 ms) and the trans-atlantic internet setting (100 ms), i.e., MSN_D is better for settings with higher latency.

5.2 Privacy-Preserving Face Recognition

To show the performance of our framework on large circuits we benchmark it on circuits for privacy-preserving face recognition. Here, the client holds a query face and wants to determine whether it matches one of the faces input by the server. We consider two commonly used face-recognition algorithms: Eigenfaces (§5.2.1) and the index-based Hamming distance scheme of SCiFI [28] (§5.2.2).

5.2.1 Using Eigenfaces. Here, the client inputs a query face image Γ with N pixels of b-bits each. The server inputs M faces projected into a K-dimensional feature space $\Omega_1, ..., \Omega_M$. If Γ matches one of the faces in the database, the client receives the index i_{min} of the closest match (see §D for details). A privacy-preserving protocol based on additively homomorphic encryption was given in [9] and subsequently improved by combining it with garbled circuits [14, 30].

We use the same parameters as [9, 14, 30]: $N = 10{,}304$; $b = 8$; $K = 12$. Using the depth-optimized circuits of §3 results in a circuit for 320 faces in the database FR^{320}_{Eig} with $\mathbf{S}(\text{FR}^{320}_{Eig}) = 14{,}887{,}713$ and $\mathbf{D}(\text{FR}^{320}_{Eig}) = 120$ and a circuit for 1,000 faces FR^{1000}_{Eig} with $\mathbf{S}(\text{FR}^{1000}_{Eig}) = 22{,}811{,}392$ and $\mathbf{D}(\text{FR}^{1000}_{Eig}) = 128$.

The performance of our implementation in comparison with previous works is shown in Tab. 5 (similar machines connected via LAN). Our implementation outperforms previous work by at least factor 8 in the online phase for a database with 320 faces (factor 10 for 1,000 faces) while maintaining a fast setup time. Also, our implementation scales very well with increasing database size due to the SIMD operations of §4.5 (≈ 0.33 ms online time per face in the database).

5.2.2 Using Hamming Distance. In the Hamming distance based algorithm proposed in [28], the client maps his query face to an ℓ-bit index vector that represents the characteristics of the face. Face recognition is done by computing the Hamming distance between the client's index vector and each of the server's M index vectors of faces in the database and checking whether this distance is below a pre-computed, face-specific threshold. The original protocols proposed in SCiFI [28] were based on additively homomorphic encryption and later improved by the garbled circuits framework of [15].

Table 5. Runtimes for Eigenfaces-based face recognition (on similar machines)

Faces in Database	320				1,000			
Framework	[9]	[14]	[30]	This work	[9]	[14]	[30]	This work
Setup phase [s]	18	38.1	n/a	15.7	n/a	83.4	n/a	24.0
Online phase [s]	22	41.5	8.4	1.1	n/a	56.2	13	1.3
Overall time [s]	40	76.9	n/a	17.7	n/a	139.6	n/a	26.3

Table 6. Runtimes for index-based face recognition (on similar machines)

Framework	[28]	[15]				This work					
Faces in Database	100	100		320		100		320		50,000	
Latency [ms]	LAN	0.2	100	0.2	100	0.2	100	0.2	100	0.2	100
Setup phase [s]	213	0.40	0.67	0.40	0.67	0.30	1.09	0.49	1.35	44.85	55.87
Online phase [s]	31	1.75	2.18	5.14	6.34	0.006	0.53	0.01	0.62	0.90	3.02
Overall time [s]	244	8.79	9.85	42.9	44.5	0.31	1.64	0.51	2.01	45.98	59.68

We construct a circuit $\mathrm{FR}_{SCI}^{(\ell,N)}$ for the SCiFI recognition algorithm that consists of N parallel instantiations of the Boyar-Peralta count circuit CNT_{BP}^{ℓ} (cf. §3.4)[1] and the size-optimized greater than circuit $\mathrm{GT}_{S}^{\lceil \log_2 \ell+1 \rceil}$ (cf. §3.3) with $\mathbf{S}(\mathrm{FR}_{SCI}^{(\ell,N)}) = N(\ell + \lceil \log_2 \ell + 1 \rceil - d_H(\ell))$ and $\mathbf{D}(\mathrm{FR}_{SCI}^{(\ell,N)}) = \lceil \log_2 \ell + 1 \rceil + 1$. Similar to previous works we choose $\ell = 900$, resulting in a circuit with 906 AND gates per face in the database and a depth of 11. Tab. 6 shows a performance comparison between the original protocols of [28] and the framework of [15] and our framework, both evaluating the SCiFI circuit $\mathrm{FR}_{SCI}^{(900,N)}$.

The original SCiFI protocols [28] require a constant setup time of 213 s and an online time of 0.31 s per face in the database. Their main advantage is that the online phase can be parallelized on multiple servers. For the circuit-based approaches we can observe that our framework outperforms [15] in online time by factor 300 for 100 faces (factor 500 for 320 faces) in the LAN setting and by factor 4 (factor 10 for 320 faces) in the trans-atlantic internet setting. Most notable is the sub-linear scaling in the database size due to the SIMD operations (cf. §4.5), which enable our framework to process large-scale databases within a very short online time (18 μs per face in the LAN setting and 60 μs per face in the trans-atlantic internet setting). In contrast, the main performance bottleneck of the [15] framework is the time for generating large circuits, i.e., the difference between overall time and setup plus online time.

Acknowledgement. This work was supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE and by the Hessian LOEWE excellence initiative within CASED.

[1] This circuit has about half the size of the count circuit proposed in [15].

References

1. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992)
2. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995)
3. Bogdanov, D., Jagomägis, R., Laur, S.: A universal toolkit for cryptographically secure privacy-preserving data mining. In: Chau, M., Wang, G.A., Yue, W.T., Chen, H. (eds.) PAISI 2012. LNCS, vol. 7299, pp. 112–126. Springer, Heidelberg (2012)
4. Boyar, J., Peralta, R.: Concrete multiplicative complexity of symmetric functions. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 179–189. Springer, Heidelberg (2006)
5. Choi, S.G., Hwang, K.-W., Katz, J., Malkin, T., Rubenstein, D.: Secure multiparty computation of Boolean circuits with applications to privacy in on-line marketplaces. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 416–432. Springer, Heidelberg (2012)
6. Damgård, I., Geisler, M., Krøigaard, M., Nielsen, J.B.: Asynchronous multiparty computation: Theory and implementation. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 160–179. Springer, Heidelberg (2009)
7. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012)
8. Earle, L.G.: Latched carry-save adder. IBM Technical Disclosure Bulletin 7(10), 909–910 (1965)
9. Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 235–253. Springer, Heidelberg (2009)
10. Garay, J., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007)
11. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press (2004)
12. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Symposium on Theory of Computing (STOC 1987), pp. 218–229. ACM (1987)
13. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols: Techniques and Constructions, 1st edn. Springer (2010)
14. Henecka, W., Kögl, S., Sadeghi, A.R., Schneider, T., Wehrenberg, I.: TASTY: Tool for Automating Secure Two-partY computations. In: Computer and Communications Security (CCS 2010), pp. 451–462. ACM (2010)
15. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: Security Symposium. USENIX (2011)
16. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
17. Kerschbaum, F.: Automatically optimizing secure computation. In: Computer and Communications Security (CCS 2011), pp. 703–714. ACM (2011)

18. Kolesnikov, V., Sadeghi, A.R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 1–20. Springer, Heidelberg (2009)
19. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
20. Kreuter, B., Shelat, A., Shen, C.H.: Billion-gate secure computation with malicious adversaries. In: Security Symposium. USENIX (2012)
21. Ladner, R.E., Fischer, M.J.: Parallel prefix computation. Journal of the ACM 27(4), 831–838 (1980)
22. Li, B., Li, H., Xu, G., Xu, H.: Efficient reduction of 1 out of n oblivious transfers in random oracle model. Cryptology ePrint Archive, Report 2005/279 (2005)
23. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing two-party computation efficiently with security against malicious adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
24. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: Security Symposium, pp. 287–302. USENIX (2004)
25. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. Journal of Cryptology 18(1), 1–35 (2005)
26. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Electronic Commerce (EC 1999), pp. 129–139. ACM (1999)
27. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012)
28. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: SCiFI - a system for secure face identification. In: Symp. on Security and Privacy, pp. 239–254. IEEE (2010)
29. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
30. Sadeghi, A.R., Schneider, T., Wehrenberg, I.: Efficient privacy-preserving face recognition. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 229–244. Springer, Heidelberg (2010)
31. Savage, J.E.: Models of Computation: Exploring the Power of Computing, 1st edn. Addison-Wesley Pub., Boston (1997)
32. Sklansky, J.: An evaluation of several two-summand binary adders. IRE Transactions on Electronic Computers EC-9(2), 213–226 (1960)
33. Yao, A.C.: How to generate and exchange secrets. In: Foundations of Computer Science (FOCS 1986), pp. 162–167. IEEE (1986)
34. Yoo, J.T., Smith, K.F., Gopalakrishnan, G.: A fast parallel squarer based on divide-and-conquer. IEEE Journal of Solid-State Circuits 32, 909–912 (1995)

A Summary of Circuit Building Blocks

Table 7. Size and Depth of Circuit Constructions (d_H: Hamming weight)

Circuit	Size **S**	Depth **D**
Addition		
Ripple-carry ADD/SUB$^{\ell}_{RC}$	ℓ	ℓ
Ladner-Fischer ADD$^{\ell}_{LF}$	$1.25\ell\lceil\log_2 \ell\rceil + \ell$	$2\lceil\log_2 \ell\rceil + 1$
LF subtraction SUB$^{\ell}_{LF}$	$1.25\ell\lceil\log_2 \ell\rceil + 2\ell$	$2\lceil\log_2 \ell\rceil + 2$
Carry-save ADD$^{(\ell,3)}_{CSA}$	$\ell + \mathbf{S}(\text{ADD}^{\ell})$	$\mathbf{D}(\text{ADD}^{\ell})+1$
RC network ADD$^{(\ell,n)}_{RC}$	$\ell n - \ell + n - \lceil\log_2 n\rceil - 1$	$\lceil\log_2 n - 1\rceil + \ell$
CSA network ADD$^{(\ell,n)}_{CSA}$	$\ell n - 2\ell + n - \lceil\log_2 n\rceil$ $+\mathbf{S}(\text{ADD}^{\ell+\lceil\log_2 n\rceil}_{LF})$	$\lceil\log_2 n - 1\rceil$ $+\mathbf{D}(\text{ADD}^{\ell+\lceil\log_2 n\rceil}_{LF})$
Multiplication		
RCN school method MUL$^{\ell}_{RC}$	$2\ell^2 - \ell$	$2\ell - 1$
CSN school method MUL$^{\ell}_{CSN}$	$2\ell^2 + 1.25\ell\lceil\log_2 \ell\rceil - \ell + 2$	$3\lceil\log_2 \ell\rceil + 4$
RC squaring SQR$^{\ell}_{RC}$	$\ell^2 - \ell$	$2\ell - 3$
LF squaring SQR$^{\ell}_{LF}$	$\ell^2 + 1.25\ell\lceil\log_2 \ell\rceil - 1.5\ell - 2$	$3\lceil\log_2 \ell\rceil + 3$
Comparison		
Equality EQ$^{\ell}$	$\ell - 1$	$\lceil\log_2 \ell\rceil$
Sequential greater than GT$^{\ell}_{S}$	ℓ	ℓ
D&C greater than GT$^{\ell}_{DC}$	$3\ell - \lceil\log_2 \ell\rceil - 2$	$\lceil\log_2 \ell\rceil + 1$
Selection		
Multiplexer MUX$^{\ell}$	ℓ	1
Minimum MIN$^{(\ell,n)}$	$(n-1)(\mathbf{S}(\text{GT}^{\ell})+\ell)$	$\lceil\log_2 n\rceil(\mathbf{D}(\text{GT}^{\ell})+1)$
Minimum index MIN$^{(\ell,n)}_{IDX}$	$(n-1)(\mathbf{S}(\text{GT}^{\ell})+\ell+\lceil\log_2 n\rceil)$	$\lceil\log_2 n\rceil(\mathbf{D}(\text{GT}^{\ell})+1)$
Set Operations		
Set union $\cup^{\ell}$	ℓ	1
Set intersection $\cap^{\ell}$	ℓ	1
Set inclusion $\subseteq^{\ell}$	$2\ell - 1$	$\lceil\log_2 \ell\rceil + 1$
Count		
Full Adder count CNT$^{\ell}_{FA}$	$2\ell - \lceil\log_2 \ell\rceil - 2$	$\lceil\log_2 \ell\rceil$
Boyar-Peralta count CNT$^{\ell}_{BP}$	$\ell - d_H(\ell)$	$\lfloor\log_2 \ell\rfloor$
Distances		
Manhattan distance DST$^{\ell}_{M}$	$2\mathbf{S}(\text{SUB}^{\ell})+\mathbf{S}(\text{ADD}^{(\ell,3)})+1$	$\mathbf{D}(\text{SUB}^{\ell})+\mathbf{D}(\text{ADD}^{(\ell,3)})+1$
Euclidean distance DST$^{\ell}_{E}$	$2\mathbf{S}(\text{SUB}^{\ell})+2\mathbf{S}(\text{SQR}^{\ell})$ $+\mathbf{S}(\text{ADD}^{(2\ell,4)})+2\mathbf{S}(\text{MUX}^{\ell})$	$\mathbf{D}(\text{SUB}^{\ell})$ $+\mathbf{D}(\text{SQR}^{\ell})+3$

B Depth Efficient Distance Circuits

B.1 Manhattan Distance

The Manhattan distance DST$^{\ell}_{M}$ between two points $p_1 = (x^{\ell}_1, y^{\ell}_1)$ and $p_2 = (x^{\ell}_2, y^{\ell}_2)$ is the distance in a two dimensional space allowing only horizontal and vertical moves and is computed as $|x^{\ell}_1 - x^{\ell}_2| + |y^{\ell}_1 - y^{\ell}_2|$. [5] give such a circuit DST$^{\ell}_{M,C}$ with size $\mathbf{S}(\text{DST}^{\ell}_{M,C}) = 9\ell$ and depth $\mathbf{D}(\text{DST}^{\ell}_{M,C}) = 2\ell + 2$. They use 4 multiplexer circuits MUX$^{\ell}$ (cf. [18]), 2 GT$^{\ell}_{S}$ circuits (§3.3), 2 SUB$^{\ell}_{RC}$ circuits (cf. [18]), and one ADD$^{\ell}_{RC}$ circuit (§3.1).

Optimization. We build a more efficient Manhattan distance circuit DST_M^ℓ as

$$x^{\ell+1} = \mathrm{SUB}^\ell(x_1^\ell, x_2^\ell), \qquad y^{\ell+1} = \mathrm{SUB}^\ell(y_1^\ell, y_2^\ell)$$
$$b_1^\ell = (x_{\ell+1}||x_{\ell+1}||...)^\ell \oplus (x_{\ell...1})^\ell, \quad b_2^\ell = (y_{\ell+1}||y_{\ell+1}||...)^\ell \oplus (y_{\ell...1})^\ell$$
$$\mathrm{DST}_M^\ell(p_1, p_2) = \mathrm{ADD}^{(\ell,3)}(b_1^\ell, b_2^\ell, 0^{\ell-2}||x_{\ell+1} \wedge y_{\ell+1}||x_{\ell+1} \oplus y_{\ell+1}).$$

We can choose between the size-optimized Ripple-carry and the depth-optimized Ladner-Fischer instantiations of SUB^ℓ and ADD^ℓ (§3.1). Using the Ripple-carry adder yields $\mathrm{DST}_{M,RC}^\ell$ with $\mathbf{S}(\mathrm{DST}_{M,RC}^\ell) = 4\ell + 1$ and $\mathbf{D}(\mathrm{DST}_{M,RC}^\ell) = 2\ell + 2$. The Ladner-Fischer variant $\mathrm{DST}_{M,LF}^\ell$ has approximately $\mathbf{S}(\mathrm{DST}_{M,LF}^\ell) = 3.75\ell\lceil\log_2 \ell\rceil + 7\ell + 1$ and $\mathbf{D}(\mathrm{DST}_{M,LF}^\ell) = 4\lceil\log_2 \ell\rceil + 6$.

B.2 Euclidean Distance

The Euclidean distance DST_E^ℓ between two points $p_1 = (x_1^\ell, y_1^\ell)$ and $p_2 = (x_2^\ell, y_2^\ell)$ is computed as $\sqrt{(x_1^\ell - x_2^\ell)^2 + (y_1^\ell - y_2^\ell)^2}$. Since computing the square root is very inefficient, the square of the Euclidean distance is often used instead (cf. [9]). We propose an efficient (squared) Euclidean distance circuit DST_E^ℓ as

$$x^{\ell+1} = \mathrm{SUB}^\ell(x_1^\ell, x_2^\ell), \qquad y^{\ell+1} = \mathrm{SUB}^\ell(y_1^\ell, y_2^\ell)$$
$$a^\ell = (x_{\ell+1}||x_{\ell+1}||...)^\ell \oplus (x_{\ell...1})^\ell, \qquad b^\ell = (y_{\ell+1}||y_{\ell+1}||...)^\ell \oplus (y_{\ell...1})^\ell$$
$$c^\ell = \mathrm{MUX}^\ell((0||0||...)^\ell, a^\ell, x_{\ell+1}), \qquad d^\ell = \mathrm{MUX}^\ell((0||0||...)^\ell, b^\ell, y_{\ell+1})$$
$$\mathrm{DST}_E^\ell(p_1, p_2) = \mathrm{ADD}^{(2\ell,4)}(\mathrm{SQR}^\ell(a^\ell), c^\ell||x_{\ell+1}, \mathrm{SQR}^\ell(b^\ell), d^\ell||y_{\ell+1}).$$

where $\mathrm{MUX}^\ell((0||0||...)^\ell, a^\ell, x_{\ell+1})$ is a multiplexer that selects $(0||0||...)^\ell$ if $x_{\ell+1}$ is 0, and a^ℓ else. Note that adding $c^\ell||x_{\ell+1}$ and $d^\ell||y_{\ell+1}$ in the last step can be done as part of an addition network (§3.1.2), requiring only $2\ell+2$ additional AND gates and a constant overhead in depth. DST_E^ℓ can be instantiated with depth or size efficieny in mind. The size-efficient variant $\mathrm{DST}_{E,RC}^\ell$ uses the Ripple-carry adder and has size $\mathbf{S}(\mathrm{DST}_{E,RC}^\ell) = 2\ell^2 + 6\ell + 2$ and depth $\mathbf{D}(\mathrm{DST}_{E,RC}^\ell) = 3\ell + 2$. The depth-efficient variant $\mathrm{DST}_{E,LF}^\ell$ uses the Ladner-Fischer adder and has $\mathbf{S}(\mathrm{DST}_{E,LF}^\ell) = 2\ell^2 + (7.5\ell + 2)\lceil\log_2 \ell\rceil + 11\ell - 5$ and $\mathbf{D}(\mathrm{DST}_{E,LF}^\ell) = 5\lceil\log_2 \ell\rceil + 9$.

C Mobile Social Network Circuit

The mobile social network circuit $\mathrm{MSN}^{((\ell,m),N)}$ has two steps. First, the Manhattan distance between U and each user U_i in the database is computed and compared to the threshold δ^{m+1}, resulting in a bit $c_i = \mathrm{GE}^{m+1}(\delta^{m+1}, \mathrm{DST}_M^m(L_r, L_i))$ which is used to multiplex between 0 and the size of the set intersection as s_i=$\mathrm{MUX}^{\lceil\log_2 \ell\rceil}(0^{\lceil\log_2 \ell\rceil}, \mathrm{CNT}^\ell(\cap^\ell(H_r, H_i)), c_i)$, where MUX is a multiplexer circuit (cf. [18]) and $\cap$ is a circuit for set-intersection (cf. [5]). Next, the index k

of the user U_k with maximum value s_i is determined using the MAX_{IDX} circuit (cf. [18]): $\mathrm{MSN}^{((\ell,m),N)}(H_{1\ldots N}, L_{1\ldots N}) = \mathrm{MAX}_{IDX}^{(\lceil \log_2 \ell \rceil, N)}(s_{1\ldots N})$. The original circuit MSN_C in [5] has approximately size $\mathbf{S}(\mathrm{MSN}_C^{((\ell,m),N)}) = N(10m + 3\ell + 2\lceil \log_2 \ell \rceil + \lceil \log_2 N \rceil)$ and depth $\mathbf{D}(\mathrm{MSN}_C^{((\ell,m),N)}) = 2m + \lceil \log_2 N \rceil(\lceil \log_2 \ell \rceil + 1) + 7$.

Using the optimized circuit building blocks CNT_{BP} (§3.4), $\mathrm{DST}_{M,RC}$ (§B.1), and GT_S (§3.3) we obtain a size-efficient variant MSN_S with approximately size $\mathbf{S}(\mathrm{MSN}_S^{((\ell,m),N)}) = N(5m + 2\ell + 3\lceil \log_2 \ell + 1 \rceil + \lceil \log_2 N \rceil - d_H(\ell) + 2)$ and depth $\mathbf{D}(\mathrm{MSN}_S^{((\ell,m),N)}) = 2m + \lceil \log_2 N \rceil(\lceil \log_2 \ell + 1 \rceil + 1) + 4$. Alternatively, using $\mathrm{DST}_{M,LF}$ (§B.1) and GE_{DC} (§3.3) yields a depth-efficient variant MSN_D which has size $\mathbf{S}(\mathrm{MSN}_D^{((\ell,m),N)}) = N(3.75m\lceil \log_2 m \rceil + 9m + 3\ell + 4\lceil \log_2 \ell + 1 \rceil + \lceil \log_2 N \rceil)$ and depth $\mathbf{D}(\mathrm{MSN}_D^{((\ell,m),N)}) = 5\lceil \log_2 m \rceil + \lceil \log_2 N \rceil(\lceil \log_2 \lceil \log_2 \ell + 1 \rceil \rceil + 2) + 8$. The difference between the two circuits is that the size of the depth-efficient circuit is larger by approximately $N(3.75m\lceil \log_2 m \rceil + 4m + \ell)$ whereas its depth decreases logarithmically in m and $\log_2 \ell$.

D Face Recognition Circuit

The circuit for face recognition FR implements the Eigenface recognition algorithm (cf. [30] for a detailed description) by first projecting the face image Γ into the K-dimensional Eigenface space. The projection is performed by computing for $i = 1, \ldots, K$: $\omega_i = \sum_{j=1}^{N} u_{i,j}\Gamma_j - \sum_{j=1}^{N} u_{i,j}\Psi_j$ where the server inputs the Eigenfaces $u_{i,j}$ and the average face Ψ and locally precomputes $-\sum_{j=1}^{N} u_{i,j}\Psi_j$. Afterwards, the square of the Euclidean distance between the ω_i and the server's projected faces $\Omega_1, ..., \Omega_M$ is computed as $D_j = \sum_{i=1}^{K}(\Omega_{j,i} - \omega_i)^2$, for $j = 1, \ldots, M$. Finally, the minimum distance D_{min} is selected and compared to a pre-determined threshold τ, input by the server. If $D_{min} \leq \tau$ the client learns j_{min} and $\perp$ otherwise.

As squaring is cheaper than multiplying (cf. §3.2), we optimize computation of the projection phase by computing $\sum_{j=1}^{N} u_{i,j}\Gamma_j$ as $\sum_{j=1}^{N} \frac{(u_{i,j}+\Gamma_j)^2 - u_{i,j}^2 - \Gamma_j^2}{2}$. Although this form might look more complex at a first glance, the resulting circuit requires only around half the number of AND gates compared to [30]. Using the optimization of [17], the server locally computes and inputs $-\sum_{j=1}^{N} u_{i,j}^2$ and the client locally computes and inputs $-\sum_{j=1}^{N} \Gamma_j^2$.

As building blocks we use a generalization of the Euclidean distance circuit DST_E (§B.2) to K dimensions, ADD_{CSA} and ADD_{RC} (§3.1), GT_{DC} (§3.3), and a circuit for computing the minimum index $\mathrm{MIN}_{IDX,DC}$ (cf. [18]).

The Untapped Potential of Trusted Execution Environments on Mobile Devices

N. Asokan[1], Jan-Erik Ekberg[2], and Kari Kostiainen[3]

[1] University of Helsinki
asokan@acm.org
[2] Nokia Research Center
jan-erik.ekberg@nokia.com
[3] ETH Zurich
kari.kostiainen@inf.ethz.ch

A *trusted execution environment* (TEE) is a secure, integrity-protected processing environment, consisting of processing, memory and storage capabilities. It is *isolated* from the "normal" processing environment, sometimes called the *rich execution environment* (REE), where the device operating system and applications run. TEEs can make it possible to build REE applications and services with better security and usability by partitioning them so that sensitive operations are restricted to the TEE and sensitive data, like cryptographic keys, never leave the TEE. In our daily lives, we encounter more and more services that use dedicated hardware tokens to improve their security: one-time code tokens for two-factor authentication, wireless tokens for opening doors in buildings or cars, tickets for public transport, and so on. Mobile devices equipped with TEEs have the potential for replacing these many tokens thereby improving the usability for users while also reducing the cost for the service providers without hampering security.

Chances are that the mobile device in your pocket sports a hardware-based TEE. Chances are, too, that you have not come across too many applications that actually make use of TEE functionality. In this article, we explain why this situation came to pass and what the future may hold.

Security in mobile world had a very different trajectory compared to the world of personal computers [6]. The various stakeholders had strict security requirements, some of which date back to two decades ago, right at the beginning of the explosion of personal mobile communications. Some examples of such requirements are:

- Standardization requirements like ensuring that the device identifier (also known as International Mobile Equipment identifier or IMEI) will resist "manipulation and change, by any means (e.g., physical, electrical and software)" [1],
- Regulatory requirements like ensuring secure storage for radio frequency parameters calibrated during manufacture,
- Business requirements like ensuring that subsidy locks on subsidized mobile phones given to a subscriber as part of a contract with a mobile operator

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 293–294, 2013.

cannot be circumvented; and implementing digital rights management schemes as securely as possible, and
- end user perceptions which had grown accustomed to the reliability of early feature phones (e.g., no blue screen of death).

These requirements incentivized mobile device manufacturers, chip vendors and platform providers to design and deploy hardware and platform security mechanisms for mobile platforms from early on. Hardware-based TEEs were seen as essential building blocks in meeting these requirements. The first mobile phones with hardware-based TEEs appeared almost a decade ago in the form of Nokia phones using processors from Texas Instruments [7]. One way to realize a TEE is by implementing a secure processor mode. The primary example of such an implementation is ARM TrustZone [2]. ARM processors capable of TrustZone power the overwhelming majority of smartphones and tablets today. Almost every smartphone and tablet today contains a TEE like ARM TrustZone (as well as software platform security mechanisms [6]).

Despite such a large-scale deployment, the use of TEE functionality has been largely restricted to its original intended uses. There has been no widely available means for application developers to benefit from existing TEE functionality. Some recent research projects, like the On-board Credentials system [5,4,3] we developed at Nokia Research Center have been deployed in commercial products, but these remain proprietary.

Fortunately, with emerging standardization in Global Platform (`http://www.globalplatform.org`) and the Mobile Working Group of the the Trusted Computing Group (`https://www.trustedcomputinggroup.org/`) this situation is about to change. We expect to see implementaitons of standardized interfaces for accessing and using TEE emerging across different platforms.

References

1. 3GPP: 3GPP TS 42.009 Security Aspects. 3GPP (March 2001), `http://www.3gpp.org/ftp/Specs/html-info/42009.htm`
2. ARM: Building a Secure System using TrustZone©Technology. ARM Security Technology (April 2009), `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html`
3. Ekberg, J.E.: Securing Software Architectures for Trusted Processor Environments. Doctoral dissertation, Aalto University (May 2013)
4. Kostiainen, K.: On-board Credentials: An Open Credential Platform for Mobile Devices. Doctoral dissertation, Aalto University (May 2012)
5. Kostiainen, K., Ekberg, J.E., Asokan, N., Rantala, A.: On-board credentials with open provisioning. In: Proceedings of ACM Symposium on Information, Computer and Communications Security, ASIACCS (2009)
6. Kostiainen, K., Reshetova, E., Ekberg, J.E., Asokan, N.: Old, new, borrowed, blue – a perspective on the evolution of mobile platform security architectures. In: Proceedings of the First ACM Conference on Data and Application Security and Privacy (CODASPY) (February 2011)
7. Srage, J., Azema, J.: M-Shield mobile security technology, TI White paper (2005), `http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf`

STARK
Tamperproof Authentication to Resist Keylogging

Tilo Müller, Hans Spath, Richard Mäckl, and Felix C. Freiling

Department of Computer Science
Friedrich-Alexander-University, Erlangen, Germany

Abstract. The weakest link in software-based full disk encryption is the authentication procedure. Since the master boot record must be present unencrypted in order to launch the decryption of remaining system parts, it can easily be manipulated and infiltrated by bootkits that perform keystroke logging; consequently password-based authentication schemes become attackable. The current technological response, as enforced by BitLocker, verifies the integrity of the boot process by use of the trusted platform module. But, as we show, this countermeasure is insufficient in practice. We present STARK, the first tamperproof authentication scheme that *mutually authenticates* the computer and the user in order to resist keylogging during boot. To achieve this, STARK combines two ideas in a novel way: (1) STARK implements *trust bootstrapping* from a secure token (a USB flash drive) to the whole PC. (2) In STARK, users can securely verify the authenticity of the PC before entering their password by using *one-time boot prompts*, that are updated upon successful boot.

Keywords: Disk Encryption, Evil Maid Attacks, Authentication, TPM.

1 Introduction

Full disk encryption (FDE) protects sensitive data against unauthorized access in the event that a device is physically lost or stolen. The risk of data exposure is reduced by rendering disks unreadable to unauthorized users through encryption technologies like AES [11]. Unlike file encryption, FDE encrypts whole disks automatically on operating system level, without the need to take action for single files. Therefore, FDE is both more secure and more user-friendly than file encryption, and consequently, it can be considered best practice for securing sensitive data against accidental loss and theft.

As stated in a SECUDE survey from 2012 [36], full disk encryption is the most popular strategy for data protection in the majority of U.S. enterprises today. Also, the U.S. government recommends agencies to encrypt data on mobile devices to compensate the lack of physical security outside their agency location [19]. In Ponemon's annual study about the *U.S. Cost of a Data Breach* from 2011 [32], it is stated that "breaches involving lost or stolen laptop computers or other mobile data-bearing devices remain a consistent and expensive threat". This threat must be counteracted by full disk encryption systems like

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 295–312, 2013.

BitLocker for Windows, FileVault for Mac, dm-crypt for Linux, or the cross-platform utility TrueCrypt.

1.1 Threat Model

FDE cannot protect sensitive data when a user logs into the system and leaves it unattended. Likewise, FDE does generally not protect against system subversion through malware. In the case of malware infiltration, adversaries can access data remotely over a network connection with user privileges after the user logged in. Hence, hard disk encryption is only intended for scenarios where an adversary gains *physical access* to a target.

Given physical access, two types of attacks can be distinguished: opportunistic and targeted attacks. In *opportunistic* attacks the adversary steals a computer and immediately tries to retrieve the data. This is the attack scenario which is withstood by all widespread FDE solutions, including BitLocker and TrueCrypt. If an attacker simply grabs the computer and runs, data decryption without having the key or password is a futile task with today's crypto primitives like AES. However, a more careful adversary can carry out a *targeted* attack on the key, password, or access control management. Besides *cold boot attacks* [13,8] and *DMA attacks* over FireWire [5,30], PCIe [7,9] or Thunderbolt [6,33], keylogging attacks via *bootkits* [24,25] are a notable threat.

Software-based FDE needs to modify the master boot record (MBR) of a hard drive in order to present pre-boot environments for user authentication. Commonly, pre-boot screens ask users for credentials in form of a secret password or passphrase, but they can also ask for credentials such as smart cards and tokens. Only after a user is authenticated, the operating system is decrypted and prepared to take over system control.

The MBR of an encrypted hard drive can easily be manipulated because it is necessarily left unencrypted for bootstrapping since CPUs can interpret only unencrypted instructions. As a consequence, bootkits can always be placed in the MBR to subvert the original bootloader with software-based keylogging. Such attacks are also referred to as *evil maid attacks* [16] and typically require access to the target machine twice: Let the victim be a traveling salesman who left his laptop in a hotel room and goes out for dinner. An "evil maid" can gain physical access to her target system unsuspiciously now. She replaces the original MBR with a malicious bootloader that performs keylogging and, later on, the unaware salesman boots up his machine and enters his password. On the next event, the evil maid can access the laptop a second time and reads out the logged passphrase.[1]

As shown by a recent study [29] on the security of *hardware*-based FDE (so-called self-encrypting drives, SEDs), evil maid attacks are generally *not* defeated by SEDs although these drives encrypt the MBR [14]. The reason is that evil maids can alternatively replace the entire disk drive, plug in a tiny bootable USB drive, flash the BIOS image [35,18] or UEFI image [26,3], or even replace the machine

[1] Following this cover story, we refer to female attackers and male victims throughout this paper.

with an identical model. Overall, targeted attacks with repeated physical access (such as evil maid attacks) constitute the threat model of our paper.

1.2 Bootkit Attacks

The original evil maid attack was implemented against TrueCrypt in 2009 [16]. Earlier that year, another bootkit, called the *Stoned Bootkit* [31], had circumvented TrueCrypt as well. Until today, TrueCrypt is vulnerable to these attacks and the program's authors do not plan future improvements. To the contrary, they argue that bootkits "require the attacker to have [...] physical access to the computer, and the attacker needs you to use the computer after such an access. However, if any of these conditions is met, it is actually impossible to secure the computer". [39]

Microsoft's BitLocker, on the other hand, defeats software keyloggers up to a certain degree as it assures the integrity of the boot process by means of the trusted platform module (TPM). TPMs are used to build trusted checksums over sensitive boot parameters, such as firmware code, the BIOS, and the bootloader [40]. BitLocker's decryption key can only be derived if these checksums are in line with the reference configuration from system setup. Otherwise, users cannot decrypt their data and hopefully become suspicious that somebody manipulated their machine.

However, at the end of 2009, even BitLocker was successfully compromised by bootkit attacks. Türpe et al. [42] practically performed *tamper and revert* attacks that (1) tamper with the bootloader for introducing keylogging functionality, (2) let the victim enter his password into a forged text-mode prompt, (3) revert to the original bootloader, and (4) reboot. The victim may wonder about the reboot, but most likely he will enter his password again and proceed as usual – unaware of the fact that his password was already logged.

We reproduced both, the attack against TrueCrypt and the attack against BitLocker, with our own malicious bootloaders. Both attacks are still effective today.

1.3 Related Work: Attempts for Countermeasures

As shown by tamper and revert attacks, TPMs alone are *not* suitable to guarantee a trusted password prompt. Consequently, other countermeasures have been taken into consideration in recent years.

The first countermeasures were *external bootloaders* such as the Anti-Bootkit Project [2] from 2010. But even if the integrity of external USB bootloaders can be assured, because they are never left unattended, this measure is insufficient for several reasons: First, it remains unclear which USB port is preferred for booting. An attacker could plug in her own, tiny USB flash drive at the back of the machine. Second, BIOS passwords can often be reset by removing the onboard battery. This allows an attacker to reconfigure the boot sequence at her convenience, e.g., to boot from MBR. Third, the BIOS or UEFI might be manipulated and display a fake password prompt. And fourth, the entire target machine could simply be replaced with a manipulated model.

Another countermeasure named Anti Evil Maid [17] was introduced in 2011. This project mentions a *mutual authentication* scheme in the context of hard disk encryption for the first time: Only if the boot process behaves with integrity, a secret (user-defined) message can be unsealed by means of the trusted platform module. This message must be shown to the user before he enters his password. If the secret message cannot be shown, the user is strongly advised not to enter his password because the boot process is likely to be compromised.

1.4 Possible Attacks against Anti Evil Maid

Considering travelers who boot their laptops at public places like airports, and in conference and meeting halls, it is unlikely that the authentication message from Anti Evil Maid remains confidential for a long time. It will be present on surveillance cameras, can be spied upon by shoulder surfing, and can perfectly be reconstructed as it is simple ASCII text.

Even worse, the authentication scheme of Anti Evil Maid is not secure if we take the confidentiality of the authentication message for granted. It is also not secure if we additionally assume that the sealed authentication message is placed on an external USB drive. Anti Evil Maid only raises the number of required physical accesses from two to three, because the attack by Türpe et al. [42] against BitLocker can be extended as follows:

1. On the first physical access, an evil maid manipulates the bootloader in a way that it copies USB drives, e.g., into the very last sectors of the hard drive or into another USB drive. Then she lets the victim plug in his USB drive and, after it has been cloned, the bootloader automatically reverts to its original state and reboots. The reboot occurs quickly after the copy procedure, and thus an unaware user may not even recognize suspicious behavior.
2. On the second physical access, the evil maid boots up her target with the recently cloned thumb drive and notes down the appearing authentication message. She builds her own bootloader with this message and overwrites the original bootloader a second time. The remaining attack equals the procedure of tamper and revert: The evil maid lets her victim enter his password and after it has been logged, the bootloader automatically reverts to its original state and reboots again. This time, the victim may wonder about the reboot, but most likely he will re-enter his password and proceed as usual.
3. On the third physical access, the evil maid can read out the logged password and so she gains full access to the data.

If a second password, e.g., in form of a PIN for the TPM, is used instead of external thumb drives, as alternatively proposed [17], the attack works analogously with a second password logger instead of the USB drive cloner. If images instead of simple text messages are used, as alternatively proposed [17], too, the attack works similarly as well, because appearing images can be photographed or digitally retrieved through cold boot attacks.

1.5 Contributions

We present the design and implementation of STARK, an authentication protocol that *fully* defends against targeted attacks with repeated physical access for the first time. The goal of STARK is to reveal boot process manipulations before the user enters his password. To this end, we introduce a *mutual authentication scheme* that proves the integrity of a computer to its users. Similar to Anti Evil Maid, our authentication scheme is based on sealed authentication messages and external USB drives, but beyond that it introduces the concept of a *one-time password prompt*.

Roughly speaking, the authentication sequence of STARK works as follows:

- Upon boot, a user-defined authentication message is displayed. The message is sealed by the TPM and can only be unsealed if the boot process behaves with integrity. If the secret message is not displayed, the user must not enter his password.
- To prevent replay attacks as in Anti Evil Maid, we introduce *monces*: authentication messages that are used only once. Each time an authentication message has been displayed and verified, the user must define a new one. All monces are sealed by means of the TPM and stored on external USB drives. Upon boot, a consumed monce is always overwritten by a new one.
- The USB drive must be handled like a physical key and never be left unattended. The trustworthiness of our authentication scheme depends on the trust we have in the thumb drive (*trust bootstrapping*).
- Additionally, we store a sealed token value on the USB drive, to bind the drive necessarily to the decryption process (*two-factor authentication*).

In short, the trustworthiness of a machine state is authenticated to the user by displaying ever-changing authentication messages. Attacks against those messages must fail because an attacker can only retrieve messages that have already been consumed. To summarize, our contribution is threefold:

1. With STARK, we present a mutual boot authentication protocol between users and computers that is more secure than existing solutions (cf. Sect. 2).
2. We describe a practical implementation of STARK, for which we integrated the protocol into the bootloader of the FDE solution TRESOR [28] (cf. Sect. 3).
3. We give a formal argument for the security of STARK (cf. Appendix A).

Our implementation is open source (under GNU GPL v2) and together with other information available at `http://www1.cs.fau.de/stark/`.

2 STARK Protocol and Design Choices

We now describe the actual STARK protocol. STARK is a protocol for mutual authentication between users and computers. So there are two parties that follow the STARK protocol: The user $\mathcal{U}$ and the computer $\mathcal{C}$. Computer $\mathcal{C}$ must contain a TPM, because STARK requires its sealing capabilities to attest to $\mathcal{U}$ that it behaves with integrity. On the other hand, to attest to $\mathcal{C}$ that $\mathcal{U}$ is who he claims to be, traditional passwords are used.

2.1 The STARK Protocol

We formulate STARK in the standard notation of authentication protocols. STARK requires a bootstrapping phase, called session 0. Authentication sessions are numbered consecutively starting with session 1.

Bootstrapping Phase. In the bootstrapping phase (see Fig. 1), $\mathcal{U}$ exchanges password p and the initial monce m_0 with $\mathcal{C}$ and receives $sealed_{\mathcal{C}}(m_0, t)$ from $\mathcal{C}$; t is a token value that binds the USB drive to the authentication process. Computer $\mathcal{C}$ permanently stores a hash sum $h(p, t)$ over p and the token value. After the setup phase, $\mathcal{U}$ and $\mathcal{C}$ can engage in an infinite sequence of authentication sessions.

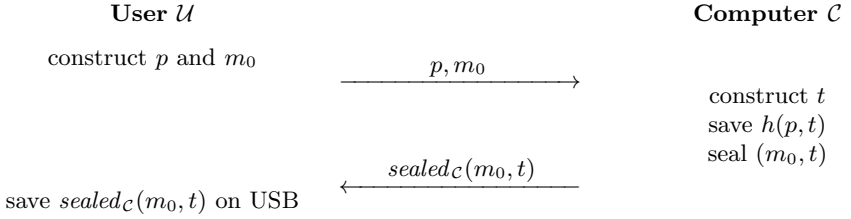


Fig. 1. Session 0 (bootstrapping phase)

Authentication Sessions. In authentication session i (see Fig. 2), $\mathcal{U}$ plugs in his USB drive and boots up $\mathcal{C}$. Computer $\mathcal{C}$ reads out $sealed_{\mathcal{C}}(m_{i-1}, t)$, unseals it, and displays m_{i-1} to $\mathcal{U}$. If $\mathcal{U}$ does not recognize m_{i-1}, $\mathcal{U}$ must abort the protocol and consider $\mathcal{C}$ as compromised. Otherwise, if $\mathcal{U}$ recognizes m_{i-1}, $\mathcal{C}$ is authenticated and $\mathcal{U}$ can safely enter his password.

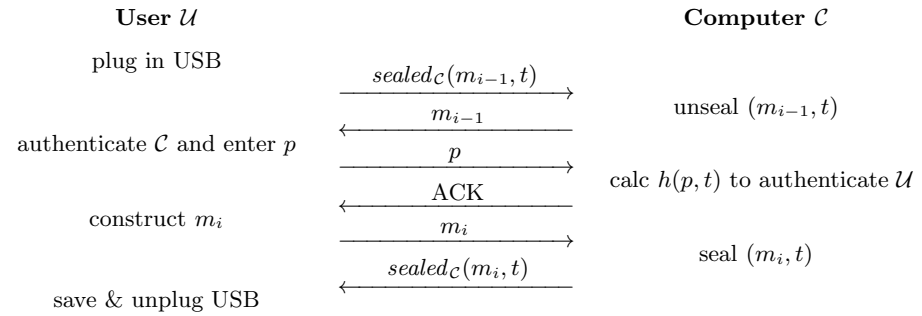


Fig. 2. Session i, $i \geq 1$ (authentication sessions)

Afterwards $\mathcal{C}$ computes $h(p,t)$ and compares it to the permanently saved value. If both values do *not* match, $\mathcal{U}$ is asked to enter his password again. Otherwise, $\mathcal{U}$ is now authenticated to $\mathcal{C}$, i.e., both parties are *mutually* authenticated. Only then $\mathcal{U}$ and $\mathcal{C}$ are safe to exchange a new monce m_i that will be used in session $i+1$. User $\mathcal{U}$ enters m_i, $\mathcal{C}$ seals it and then it is stored on the USB thumb drive. Finally, $\mathcal{U}$ can unplug the thumb drive and $\mathcal{C}$ passes control on to the operating system.

2.2 Security Argument

The central idea behind the security of STARK is that monces disallow replay attacks on the authentication message displayed to the user. To prove the security of the scheme, it is however vital that the user installs a new monce after every successful login without interference by the adversary. In traditional cryptographic protocols, replay attacks are oftentimes defeated by *nonces*, i.e., pseudo-random numbers that are used once. However, compared with traditional authentication schemes, the difficulty in the context of FDE is that one of the communication endpoints is *human*. Most humans have limited computational capabilities and therefore, authentication schemes requiring public/private key computations on both sides must be excluded and we introduce the concept of *monces*, i.e., messages that are used once. Basically, we replace the single authentication messages known from Anti Evil Maid with a series of continuously alternating messages.

In short, we use the sealing capabilities of the TPM to authenticate the computer towards the user. The user then authenticates towards the computer with traditional passwords as well as with a USB token. So upon each boot, the user gives two inputs: his password and a new authentication message m_i. Consequently evil maids can only catch outdated messages. We give a formal correctness argument for STARK in form of an inductive proof in Appendix A.

2.3 Design Components of STARK

We now describe the design choices that influenced STARK. There were two questions we had to answer when designing STARK. The first question was: What is the best way to achieve mutual authentication? Since users authenticate using a password, the problem of mutual authentication lies in having the computer prove its integrity first. We do this using a novel combination of the trusted platform module and one-time messages.

The second question was: How can we best protect the necessary authentication messages? Here we use a USB flash drive to hold the sealed monces. Another design decision, not directly related to the evil maid scenario, regards the use of token values, that are placed on the USB drive to protect users who choose weak passwords (two-factor authentication).

Mutual Authentication by Means of the Trusted Platform Module. In practice all implementations to date that are based upon TPMs fail to attest the trustworthiness of boot parameters in a way that cannot be tampered. An example for this are attacks by Türpe et al. against BitLocker as described above. Even though BitLocker builds upon TPMs, users can easily be tricked into bogus password prompts because BitLocker looks exactly the same on every machine. An attacker can reproduce the original BitLocker prompt perfectly in her own bootloaders and victims have no chance to distinguish malicious from benign behavior. However, TPMs are neither insecure nor useless taken by themselves, they must just carefully be integrated into protocols. In STARK, the TPM is used to protect one-time messages and bind them to the computer at the same time.

Bootstrapping Trust from USB Flash Drives. We store sealed monces on an external USB thumb drive. If the monces were stored on the local hard drive, STARK would be insecure to attackers who simply boot the machine and note down the appearing monce, because such attackers could build their own bootloaders with the exactly same message. If only the laptop *or* the USB drive is missing, monces remain confidential because they can only be unsealed if both entities act together. Practically, that means we can leave PCs unattended when we assure the trustworthiness of USB drives. In this sense, STARK is a protocol that performs *trust bootstrapping* from trusted USB drives.

Two-Factor Authentication with Passwords and Token Values. Besides the machine state and the user password, the FDE decryption key is based on a token value stored on the USB drive. Similar to monces, the token is sealed by means of the TPM and can only be unsealed if the machine is in line with its reference configuration. While all components introduced above defeat bootkit attacks, token values prevent dictionary attacks against weak passwords. Without a token value, adversaries would be able to boot their targets and enter passwords one after another. Tokens are also a protection in scenarios where the user disclosed his password but did not lose the thumb drive. This can be the case due to social engineering [38], skimming and shoulder surfing [23,4], reusing the password in a different system [15], or writing it down. Token values prevent such attacks because they bind USB drives compulsorily to the decryption process.

3 POTTS: A Linux-Based Implementation of STARK

We now present POTTS, our practical implementation of the authentication protocol given above.[2] POTTS ***p**rotects **o**pportunistic and **t**argeted **t**hreat **s**cenarios* through the integration of TRESOR and STARK.

[2] Note that Pepper Potts is the personal secretary of Tony Stark [10].

3.1 Integrating STARK with TRESOR

When an attacker gains physical access to a machine while it is *running* or in *standby mode*, it is not protected by STARK but must be protected by separate mechanisms. Therefore we integrated STARK into TRESOR [28] because TRESOR already defeats another kind of physical access attacks, namely cold boot attacks [13].

Cold boot attacks exploit the *remanence effect* of DRAM [12], meaning that RAM contents fade away gradually over time. Due to the remanence effect, encryption keys can be restored from memory through rebooting a system with malicious USB drives, or by replugging RAM chips physically into another machine. That makes cold boot attacks rather generic and therefore they constitute a threat for *all* software-based FDE technologies to date, including BitLocker and TrueCrypt.

TRESOR, as a countermeasure, runs encryption securely outside RAM. To this end, TRESOR avoids RAM usage entirely and runs the key management, as well as the AES algorithm, solely on the microprocessor. Some processor registers (in detail, these are the debug registers) are permanently used as cryptographic key storage. TRESOR is implemented as a Linux kernel patch because side effects like context switching, which leak information into RAM, cannot be controlled in user mode.

3.2 STARK Protocol Extensions

In the presentation of STARK above we reduced the protocol to the communication of authentication messages between $\mathcal{U}$ and $\mathcal{C}$. However, we ignored some real-world aspects, like error recovery and usability, to make the protocol more amendable for its security analysis (cf. Appendix A). We now make up for those aspects and present protocol extensions of STARK within POTTS.

Data Rescue and Recovery Mechanism. To recover from protocol failures, from hardware configuration changes, and from system compromise, we use two different keys in POTTS: a *key encryption key* (KEK) and a *data encryption key* (DEK). KEK k is composed of token value t and a checksum over password p. KEK k decrypts DEK d which is encrypted using AES and therefore is safe to be stored on hard disk. DEK d is used to decrypt user data.

While k may change frequently, e.g., through password changes, d is designed to be constant. So d prevents cumbersome re-encryption of the disk in the case of password changes since only d must newly be encrypted. Furthermore, d allows for data recovery in the case of password loss, and in the case that USB drives are lost or TPM hardware failures occur. For data recovery, users must store d in plaintext at a physically secure place, e.g., in a vault at home. This allows for migration of the hard disk into other machines and can be consulted in the case of hardware configuration changes that lead to different TPM states.

Not all protocol inconsistencies must necessarily point to a system compromise but can arise from technical problems, too. However, this is hard to differentiate

in practice, and thus we advise users to act carefully: In practice, data can be recovered with d by mounting potentially compromised HDDs on a second, trusted computer as an external device. After data rescue, the HDD in question must be formatted, including the master boot record, and the PC must be reset to factory settings (including the BIOS or UEFI image). Only then, STARK's bootstrapping phase can be rerun securely to set up a new system. Overall, STARK enables users to *identify* a potential system compromise; we do not strictly regulate which actions to take if such a compromise is detected in POTTS.

Key Derivation Function. In the previous section we stated that k is derived from a checksum over the user password p and the token value t. We now specify the key derivation function used in POTTS in detail. We do not use a single SHA-256 checksum over p but run the *password-based key derivation function 2* (PBKDF2) from RSA laboratories [20], which is recommended by NIST [41].

PBKDF2 applies a cryptographic hash to the password along with a salt and repeats the procedure several times to derive the final key (*key stretching*). The salt as well as the additional iterations make brute force attacks on the password difficult. Salt values reduce the risk of precomputed tables (*rainbow tables*) that allow attackers to look up hash values of frequently used passwords. Additionally, several iterations slow down the brute force approach.

PBKDF2 library calls are usually parameterized with *algorithm*, *password*, *salt*, *iterations*, and *keylen*. In POTTS we have chosen the variant

$$k = pbkdf2(\text{HMAC-SHA-256}, p, t, 4096, 256)$$

to derive KEK k from password p. We use the token value t as salt, perform 4096 iterations of SHA-256, and get a final keysize of 256 bits. The decrypted DEK variant d is then derived from its encrypted variant D via

$$d = decrypt(\text{AES-256}, k, D)$$

meaning that the DEK is encrypted with k and AES-256. Decrypted DEK d is a 256-bit value and used to encrypt the disk. In the case that AES-128 or AES-192 is used, superfluous key bits are just ignored.

Usability vs. Security. We now discuss some obvious limitations concerning the tradeoff between security and usability. A reasonable argument against the practical applicability of STARK might be the additional overhead we demand from users by regularly defining new monces. To be able to generate and remember different monces quickly, we advise users to write sentences about their current activities, the places they have been during the day, or their personal sentiments.

However, to increase the usability (at expense of security) in POTTS, we allow users to skip the creation of new monces and to continue with the current monce by pressing F12. Skipping new monces can be harmless in many practical scenarios, because the actual threat situation varies heavily with the physical

environment. Monces that have been displayed on a business trip should always be exchanged, of course, but monces that were displayed at home can mostly be considered as confidential.

If a user skips monces permanently, the authentication scheme of POTTS falls back to that of Anti Evil Maid. But skipping monces carefully, users have a flexible authentication scheme that is both convenient in unthreatened situations and secure in threatened situations. That is, POTTS can be adapted to the user's environment on a daily basis.

The POTTS Protocol. POTTS uses a setup phase in which $\mathcal{U}$ enters secret password p and authentication message m_0. $\mathcal{C}$ generates the 256-bit token t and the 256-bit data encryption key d. $\mathcal{C}$ then seals m_0 and t and gives the sealed tuple $sealed_{\mathcal{C}}(m_0, t)$ back to $\mathcal{U}$ who stores it on USB drive. DEK d is encrypted by means of KEK k, as described above, and can safely be stored inside a crypto footer of the disk drive. The disk drive is encrypted with TRESOR, i.e., with AES-128, -192, or -256.

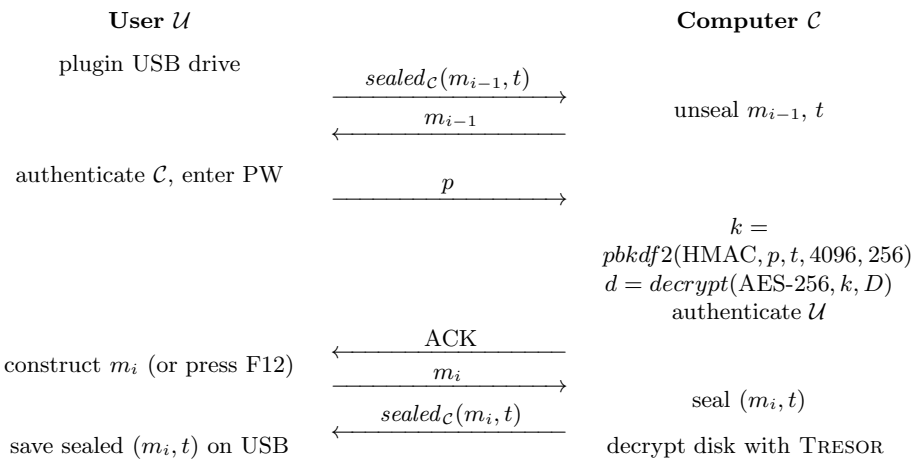


Fig. 3. POTTS authentication session i for $i \geq 1$

After the setup phase, $\mathcal{U}$ and $\mathcal{C}$ can engage in an infinite sequence of mutual authentication sessions. The extended protocol of POTTS for session i, $i \geq 1$ is given in Fig. 3. $\mathcal{C}$ derives KEK k via PBKDF2 from p and t, so that d can be decrypted. With d, a block of the hard disk is decrypted exemplarily in order to verify the correctness of p. If p is wrong, the user is asked to enter his password again.

3.3 POTTS Implementation Details

The integration of STARK into TRESOR was clearly motivated by the fact that we wanted to build an FDE system which is more secure than common FDE solutions in use today. Cold boot attacks do not fall into the defined threat model of this paper, but combining the concepts of STARK with that of TRESOR seemed reasonable. We now give some more information about the implementation of POTTS. The full details are contained in our open source distribution available online.

POTTS is based on *two* Linux kernels: (1) a minimized kernel for the STARK authentication which is loaded as first stage, and (2) the TRESOR patched kernel which is loaded as second stage. The reason behind this design choice was that TPM toolchains like TrouSerS [1] are available for user mode only, and that talking to the TPM in a pre-boot environment turned out to be difficult. In a nutshell, the boot process of POTTS follows the following sequence:

- Users plug in a bootable USB drive that contains *TrustedGRUB*. TrustedGRUB is an extended version of the known bootloader GRUB with TPM support. This makes a secure bootstrap architecture possible because TrustedGRUB measures the whole boot process and makes it verifyable for system integrity checks.
- From TrustedGRUB we boot into the first Linux system. The kernel and the user mode initialization of the first system are optimized for speed and it takes less than 3 seconds to boot it.
- In the first stage we display an *ncurses*-based interface that encapsulates the STARK protocol in a user-friendly way. For example, the interface displays the recently defined monce, asks for the password, and asks for a new monce (if F12 is not pressed). Internally, the STARK protocol is implemented via *tpm_sealdata* and *tpm_unsealdata* from the TPM toolchain. For the key derivation function PBKDF2 we make use of the OpenSSL library.
- With the recently derived key, the second kernel, which is stored on hard drive, is decrypted. After the authentication, we boot into this second kernel via *kexec*, a Linux command that replaces the currently running OS with a new OS without actually rebooting the system.
- After the second stage booted into full (TRESOR-based) Linux, end-users can use their systems as usual. Internally we pass on the decryption key to the TRESOR system over the debug registers of CPU0. Incidentally, *kexec* does not reset the debug registers of CPU0 while booting into a new kernel. All residues of the key and the password are wiped out from RAM between the first and the second stage.

Besides the simplicity of the implementation, the design choice provides us with a flexible system for future enhancements. Since passwords are not entered in pre-boot environments but in full Linux systems, we can easily add functionality to bootstrap trust from smartphones (cf. Sect. 4.2), for example.

4 Limitations and Outlook

To conclude, we give an outlook on current limitations (Sect. 4.1) and possible future enhancements (Sect. 4.2).

4.1 Limitations: Hardware-Based Attacks

STARK defeats traditional evil maid attacks that require physical access to the machine. In practice, however, the borderline from software-only attacks to more complex, hardware-based attacks is blurred. For example, hardware-based attacks can install hidden USB or PS/2 keyloggers, even to laptops [22]. Such keyloggers are generally hard to defeat in software. Academic attempts for detecting hardware keyloggers in software exist [27], but they are hardware-specific and cannot be generalized well. Another countermeasure against hardware keylogging, which is often used by online banking software, is the use of on-screen keyboards or randomized keyboard mappings. However, such countermeasures can in turn be attacked through VGA and DVI loggers [21]. So hardware-based attacks cannot be counteracted well by software solutions. In the abstract model of STARK, attackers can arbitrarily manipulate the software configuration, including the BIOS and the MBR, but they cannot manipulate hardware that is not measured by the TPM.

4.2 Outlook: Taking Advantage of New Hardware

An interesting question is whether the upcoming technology *UEFI secure boot* [43] is an alternative to STARK on future PCs. Secure boot will be supported by Windows 8 for the first time, meaning that the Windows 8 bootloader will be signed by Microsoft, and the underlying UEFI can refuse unsigned bootloaders [37]. While a TPM is a *passive* module and neither halts the computer, nor warns the user when the current configuration differs from its reference configuration [34], secure boot extends the trusted computing architecture by active components. However, UEFI images could still be replaced or manipulated with physical access and, consequently, users can still be tricked into bogus password prompts. Only STARK pursues the idea of authenticating the machine state to the user first. While secure boot stops malicious bootloaders from getting loaded, it does not prove the system state to users. Hence, STARK should be combined with secure boot in the future, but it cannot be fully replaced by secure boot.

Besides that, we plan to extend the idea of *trust bootstrapping*. Currently, we use passive USB devices as confidential storage for sealed monces. For the future, we consider more intelligent devices as confidential, e.g., Android-based smartphones which can always be carried along like physical keys, too, in order to bootstrap trust from them. Such devices have the advantage that they come with computing capabilities that allow for more advanced, RSA-based authentication schemes and can eliminate the need for monces.

Acknowledgments. We would like to thank Stefan Vömel and Johannes Bauer for helpful comments on prior versions of this paper.

References

1. TrouSerS: The open-source TCG Software Stack, http://trousers.sourceforge.net/
2. Ebfe's Anti-Bootkit Project (2010), http://ebfes.wordpress.com/tag/bootloader/
3. Galauner, A.: EFI Rootkits: Pwning your OS before it's even running. Tech. rep., dexlabs.org, SIGINT (2012)
4. Asonov, D., Agrawal, R.: Keyboard Acoustic Emanations. Tech. rep., IBM Almaden Research Center, San Jose, CA. IEEE Symposium on Security and Privacy. IEEE Computer Society (2004)
5. Böck, B.: Firewire-based Physical Security Attacks on Windows 7, EFS and BitLocker. Secure Business Austria Research Lab (August 2009)
6. Break & Enter: Adventures with Daisy in Thunderbolt-DMA-Land: Hacking Macs through the Thunderbolt interface (February 2012)
7. Carrier, B.D., Spafford, E.H.: Getting Physical with the Digital Investigation Process. IJDE 2, 2 (2003)
8. Carbone, Bean, Salois: An in-depth analysis of the cold boot attack. Tech. rep., DRDC Valcartier, Defence Research and Development, Canada, Technical Memorandum (January 2011)
9. Devine, C., Vissian, G.: Compromission physique par le bus PCI. In: Proceedings of SSTIC 2009, Thales Security Systems (June 2009)
10. Favreau, J.: Iron Man (movie), Paramount Pictures (2008)
11. FIPS. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, NIST (November 2001)
12. Gutmann, P.: Data Remanence in Semiconductor Devices. In: Proceedings of the 10th USENIX Security Symposium, Washington, D.C. USENIX Association (August 2001)
13. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest We Remember: Cold Boot Attacks on Encryptions Keys. In: Proceedings of the 17th USENIX Security Symposium, San Jose, CA, pp. 45–60. USENIX Association, Princeton University (August 2008)
14. Intel Corporation. Solid-State Drive 520 Series (2012), http://www.intel.com/content/www/us/en/solid-state-drives/solid-state-drives-520-series.html
15. Ives, B., Walsh, K.R., Schneider, H.: The domino effect of password reuse. Communications of the ACM 47(4) (April 2004)
16. Rutkowska, J.: Evil Maid goes after TrueCrypt. The Invisible Things Lab (October 2009)
17. Rutkowska, J.: Anti Evil Maid, The Invisible Things Lab (September 2011)
18. Heasman, J.: Implementing and Detecting an ACPI BIOS Rootkit. Tech. rep., NGS Consulting, BlackHat Briefings, Europe (2006)
19. Johnson, C.: Protection of Sensitive Agency Information. U.S. Executive Office of the President, Washington, D.C. 20503 (June 2006)
20. Kaliski: PKCS #5: Password-Based Cryptography Specification. In: Request for Comments (RFC): 2898, Internet Engineering Task Force, vol. 2.0. RSA Laboratories (2000)
21. KeeLog. Video Ghost (2012), http://www.keelog.com/hardware_video_logger.html

22. KeyGhost Ltd. PCI/Mini-PCI Hardware Keylogger (2006), http://www.keyghost.com/PCI-MPCI-Keylogger.htm
23. Kuhn, M.G.: Optical Time-Domain Eavesdropping Risks of CRT Displays. Tech. rep., Proceedings 2002 IEEE Symposium on Security and Privacy (SSP 2002), University of Cambridge, Computer Laboratory, Berkeley, California (May 2002)
24. Kumar, N., Kumar, V.: VBootKit 2.0 - Attacking Windows 7 via Boot Sectors. In: Hack In The Box Conference (HITBSecConf), Dubai (April 2009)
25. Li, X., Wen, Y., Huang, M., Liu, Q.: An Overview of Bootkit Attacking Approaches. In: Seventh International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2011), pp. 428–431. IEEE Computer Society (2011)
26. Loukas, K.: De Mysteriis Dom Jobsivs – Mac EFI Rootkits. Tech. rep., assurance, Black Hat Conference Proceedings, USA (2012)
27. Mihailowitsch, F.: Detecting Hardware Keyloggers. In: HITB SecConf, Kuala Lumpur, Malaysia (October 2010); cirosec GmbH. Hack In The Box
28. Müller, T., Freiling, F., Dewald, A.: TRESOR Runs Encryption Securely Outside RAM. In: 20th USENIX Security Symposium, San Francisco, California. University of Erlangen-Nuremberg, USENIX Association (August 2011)
29. Müller, T., Latzo, T., Freiling, F.: Hardware-based Full Disk Encryption (In)Security Survey. Tech. rep., Friedrich-Alexander University of Erlangen-Nuremberg, Technical Report (September 2012)
30. Panholzer, P.: Physical Security Attacks on Windows Vista. Tech. rep., SEC Consult Vulnerability Lab, Vienna (May 2008)
31. Kleissner, P.: Stoned Bootkit. Black Hat, USA (2009)
32. Ponemon Institute, LLC. 2010 Annual Study: U.S. Cost of a Data Breach. Symantec (March 2011)
33. Graham, R.D.: Thunderbolt: Introducing a new way to hack Macs, Errata Security (February 2011)
34. Rutkowska, J., Tereshkin, A., Wojtczuk, R.: Thoughts about Trusted Computing. In: EUSecWest 2009 (May 2009); The Invisible Things Lab
35. Sacco, A.L., Ortega, A.A.: Persistent BIOS Infection: The early bird catches the worm. In: Proceedings of the Annual CanSecWest Applied Security Conference, Vancouver, British Columbia, Canada. Core Security Technologies (2009)
36. SECUDE. US Full Disk Encryption 2011 Survey. Research SECUDE AG (2012)
37. Software Freedom Law Center. Microsoft confirms UEFI fears, locks down ARM devices. Tech. rep. (January 2012)
38. Thornburgh, T.: Social engineering: the Dark Art. Tech. rep., New York, NY, USA, Proceedings of the 1st Annual Conference on Information Security Curriculum Development (InfoSecCD 2004) (2004)
39. TrueCrypt Foundation. TrueCrypt: Free Open-Source On-The-Fly Disk Encryption Software for Windows 7/Vista/XP, Mac OS X and Linux (2012), http://www.truecrypt.org/
40. Trusted Computing Group, Incorporated. TPM Main Specification. Tech. Rep. Specification Version 1.2, rev. 116, TCG Published (March 2011)
41. Turan, M., Barker, E., Burr, W., Chen, L.: Special Publication 800-132: Recommendation for Password-Based Key Derivation. Tech. rep., NIST, Computer Security Division, Information Technology Laboratory (December 2010)
42. Türpe, S., Poller, A., Steffan, J., Stotz, J.-P., Trukenmüller, J.: Attacking the BitLocker Boot Process. In: Chen, L., Mitchell, C.J., Martin, A. (eds.) Trust 2009. LNCS, vol. 5471, pp. 183–196. Springer, Heidelberg (2009)
43. Unified EFI, Inc. Unified Extensible Firmware Interface Specification, Ver. 2.3.1, Errata B ed. (April 2012)

A Formal Security Argument

In this section, we formalize STARK as a security protocol within an abstract system model. To this end, we add a third player: attacker $\mathcal{A}$ who wants to break the authentication scheme between $\mathcal{U}$ and $\mathcal{C}$. Parties $\mathcal{U}$ and $\mathcal{C}$ follow the protocol given above and additional rules given below (Sect. A.1, Sect. A.2). Attacker $\mathcal{A}$ can act arbitrarily under the restrictions described below (Sect. A.3).

In the formal model of STARK, parties communicate by exchanging messages in an abstract sense. This can correspond to typing text on the keyboard, displaying a message on the screen, or attaching a USB thumb drive. Due to the immediate geographical vicinity of all parties, we assume that message exchange is reliable and synchronous, meaning that if a party sends message m, the other party receives m within a short delay.

The senders of individual messages cannot be authenticated, i.e., the receiver of a message does not necessarily know who sent the message. For example, $\mathcal{C}$ cannot distinguish text typed by $\mathcal{A}$ or $\mathcal{U}$. Similarly, a message prompt shown to $\mathcal{U}$ may come from $\mathcal{C}$ (using the original boot loader) or from $\mathcal{A}$ (using a forged boot loader).

A.1 User Model

We now give the rules that $\mathcal{U}$ has to conform to. $\mathcal{U}$ corresponds to a human user sitting in front of the keyboard and wishes to authenticate securely to $\mathcal{C}$. The security of STARK relies on the following rules for $\mathcal{U}$:

- *Completeness*: $\mathcal{U}$ must complete the protocol and define a new m_i as soon as he started session i and consumed m_{i-1}.
- *Singularity*: Every monce m_i chosen by $\mathcal{U}$ must be unpredictable and used only once.
- *Coherence*: $\mathcal{U}$ must not leave the computer during the authentication phase but execute all steps consecutively.
- *Correctness*: If $\mathcal{U}$ cannot authenticate $\mathcal{C}$, $\mathcal{U}$ must abort the protocol entirely and never engage in the protocol with the same participants again.

If any of these rules are violated, the security of STARK cannot be proven, and disk encryption may become insecure. In order to fulfill completeness, e.g., after $\mathcal{C}$ crashed due to a power cut, $\mathcal{U}$ is allowed to restart STARK at any point and to consume m_{i-1} a second time if our assumption about coherence is not violated.

A.2 Computer Model

We now define the possibilities for $\mathcal{C}$. Unlike $\mathcal{U}$, $\mathcal{C}$ is an electronic system that has powerful computation capabilities, including crucial cryptographic primitives. STARK relies on the following properties of $\mathcal{C}$:

- *Integrity*: $\mathcal{C}$ must not be compromised initially during setup phase.

- *Crypto Competence*: We assume that $\mathcal{C}$ can *seal* information in the following sense: Given a precise software configuration c of $\mathcal{C}$, $\mathcal{C}$ can encrypt and sign a message m such that $\mathcal{C}$ itself can decrypt it only when being in c again. It is unfeasible to seal or unseal information for any party besides $\mathcal{C}$ as well as for $\mathcal{C}$ itself if it is not in c.
- *Reliability*: $\mathcal{C}$ waits indefinitely for message m_i and password p to be entered without shutting down. Reading from and writing to a USB thumb drive does not fail, neither do other electronic operations fail.

Intuitively, the crypto competence of $\mathcal{C}$ corresponds to the sealing capabilities of trusted platform modules, meaning that $\mathcal{C}$ is required to have a TPM. Most notably, configuration c encompasses the BIOS settings and the master boot record. The reliability of $\mathcal{C}$ is an interesting property, both in practice as in theory, because it is generally hard to differentiate between malicious and faulty behaviors. This makes secure recovery mechanisms in the case of accidental data corruption difficult (see Sect. 3.2).

A.3 Attacker Model

Attacker $\mathcal{A}$ corresponds to an evil maid who may additionally be equipped with electronic devices. So $\mathcal{A}$ can perform both human actions as well as computationally complex calculations. Overall, $\mathcal{A}$ acts arbitrarily under the following restrictions:

- Attacker $\mathcal{A}$ can let $\mathcal{C}$ unseal any data for her (when she plugs in a USB drive and boots $\mathcal{C}$), but she cannot break cryptography, i.e., she cannot derive the password from a hash, and she cannot seal data herself.
- Attacker $\mathcal{A}$ can inject or replay arbitrary messages before and after the authentication process of STARK, and she can start an authentication process herself. But she cannot send messages to $\mathcal{U}$ or $\mathcal{C}$ within complete protocol parts, i.e., she is not allowed to interrupt the authentication process of $\mathcal{U}$. This corresponds to the event that $\mathcal{A}$ interferes with $\mathcal{U}$ while he boots up $\mathcal{C}$, which we exclude.
- Attacker $\mathcal{A}$ cannot get physical access to the USB drive. Nevertheless, she can make a one-to-one copy of it after manipulating the MBR (as sketched in the attacks from Sect. 1.4); so she may know all monces up to m_{i-2}.

A.4 Security Argument

The security argument is by induction. We argue that over an infinite sequence of sessions, the following invariant is maintained at the beginning of every session i (the security of STARK follows from item 3 of the invariant):

1. User $\mathcal{U}$ knows p and m_{i-1} and owns $sealed_{\mathcal{C}}(m_{i-1})$ on a USB drive.
2. Attacker $\mathcal{A}$ does not know m_{i-1}.
3. Attacker $\mathcal{A}$ does not know p.

Note that token t does not play a role in our security model, because t does not add security to the evil maid scenario, but in other scenarios as described above.

We now prove that STARK maintains the invariant by induction over the number of sessions. The base case (session 0) is straightforward and follows from the setup phase: $\mathcal{U}$ knows m_0 and has it sealed by $\mathcal{C}$, and the attacker $\mathcal{A}$ can neither have the password p nor the monce m_0 because of the initial integrity of $\mathcal{C}$.

For the induction step, assume that the invariant holds at the beginning of session i. There are two possibilities: Either $\mathcal{A}$ manipulates $\mathcal{C}$ and tries to inject messages, or $\mathcal{U}$ starts the protocol with regular behavior of $\mathcal{C}$, possibly after a reboot, and possibly after his USB drive was cloned. We discuss both cases in turn:

- Case 1 ($\mathcal{A}$ manipulates $\mathcal{C}$ and tries to inject messages): Since $\mathcal{A}$ does not know m_{i-1} (from invariant) and m_{i-1} cannot be guessed (from singularity of monces), any value injected by $\mathcal{A}$ towards $\mathcal{C}$ results in a wrong monce displayed to $\mathcal{U}$. $\mathcal{C}$ cannot unseal $\mathit{sealed}_{\mathcal{C}}(m_{i-1})$ because it is not in line with its reference configuration (from crypto competence of $\mathcal{C}$). Consequently, the protocol will be aborted and never restarted with the engaged parties (from correctness of user behavior).
- Case 2 ($\mathcal{U}$ starts the protocol with regular behavior of $\mathcal{C}$, possibly after a reboot): $\mathcal{A}$ may know $\mathit{sealed}_{\mathcal{C}}(m_{i-1})$, and hence, $\mathcal{A}$ can learn m_{i-1} from $\mathcal{C}$. However, $\mathcal{A}$ cannot inject any messages because $\mathcal{U}$ started the protocol (from coherence of user behavior). $\mathcal{U}$ successfully authenticates towards $\mathcal{C}$ using his password p, and completes the protocol by defining m_i (from completeness of user behavior), such that $\mathit{sealed}_{\mathcal{C}}(m_i)$ is stored on a USB drive (from reliability of $\mathcal{C}$).

In both cases, session i completes and session $i+1$ commences. Before the new session starts, observe the following points:

1. From session i, $\mathcal{U}$ knows m_i and has a sealed version of it. Of course, $\mathcal{U}$ still knows p.
2. Attacker $\mathcal{A}$ does not know m_i. Indeed, $\mathcal{A}$ may know m_{i-1}, but $\mathcal{A}$ cannot exploit it to fool $\mathcal{U}$ because $\mathcal{U}$ already defined m_i.
3. Attacker $\mathcal{A}$ does not know password p (from invariant).

Overall the invariant still holds at the beginning of session $i+1$, concluding the proof. Crucial for our argument is that *complete protocol phases* restrict the behavior of adversary $\mathcal{A}$. As explained above, we assume that $\mathcal{U}$ completes session i, meaning that when $\mathcal{U}$ executes the first step of a session, $\mathcal{U}$ does not abort but runs the protocol to completion. Without the completeness property, STARK would be insecure.

Risks of Offline Verify PIN on Contactless Cards

Martin Emms, Budi Arief, Nicholas Little, and Aad van Moorsel

School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
{martin.emms,budi.arief,n.little,aad.vanmoorsel}@ncl.ac.uk

Abstract. Contactless card payments are being introduced around the world allowing customers to use a card to pay for small purchases by simply placing the card onto the Point of Sale terminal. Contactless transactions do not require verification of the cardholder's PIN. However our research has found the redundant verify PIN functionality is present on the most commonly issued contactless credit and debit cards currently in circulation in the UK. This paper presents a plausible attack scenario which exploits contactless verify PIN to give unlimited attempts to guess the cardholder's PIN without their knowledge. It also gives experimental data to demonstrate the practical viability of the attack as well as references to support our argument that contactless verify PIN is redundant functionality which compromises the security of payment cards and the cardholder.

Keywords: Contactless Payments, Verify PIN, NFC, EMV, Chip & PIN, Credit Card, Debit Card, Card Payment.

1 Introduction

The EMV[1] specifications [5][6] control the operation of 1.62 billion of payment cards and 23.8 million of Point of Sale terminals worldwide [15]. EMV payments can be contact transactions commonly termed Chip & PIN or contactless transactions also known as Near Field Communication (NFC).

Contact payments require the cardholder to insert their card into the Point of Sale terminal and enter their PIN to authorise the transaction. Contact transactions can be any value up to the card limit or available balance on the card. Contactless payments are designed to be a convenient way to pay for low value transactions (currently up to £20 per transaction in the UK) with a card rather than cash. Designed to be faster than a traditional Chip & PIN transaction, the card is simply placed in close proximity (approximately 4cm) to the Point of Sale terminal to authorise the payment, PIN entry is not required.

In the UK the EMV specification for contact transactions supports PIN verification locally by the card (*offline*) and PIN verification remotely by the bank's computers (*online*). The specifications for contactless transactions specifically exclude the use of *offline* PIN verification (full details in [6] Book A section 5.9.3 and [10] section 2.4

[1] Europay, MasterCard, Visa is a collaboration between Visa, MasterCard, American Express and JCB to create an interoperable card payment system.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 313–321, 2013.

point 5). Contactless *offline* PIN verification requires the PIN to be transmitted wirelessly to the card which poses a security risk from eavesdropping.

The EMV specification only permits PIN entry in contactless transactions made using NFC enabled mobile devices. PIN entry is not permitted for contactless card transactions. Mobile device payments are controlled by Consumer Device CVM[2] rules, which permit *online* PIN verification, but not *offline* PIN (full details in [6] Book C3 sections 2.1 and 5.7).

This paper examines the security implications of the verify PIN functionality intended for Chip & PIN operation also being available over the contactless interface, where it can be accessed without the cardholder's knowledge or consent. Surprisingly many of the contactless cards currently in circulation in the UK allow access to *offline* verify PIN.

The attack scenario presented draws upon research carried out into the predictability of PINs [2] which shows that there is a subset of PINs that are much more commonly used; meaning guesses from this subset are much more likely to be successful.

The implementation work builds upon related investigations into the vulnerability of EMV contactless payment cards to various attacks, such as skimming [7][8] and transaction relay [4][9]. These papers show that the wireless interface makes contactless payment cards vulnerable to new modes of attack that were not present in Chip & PIN. Other research [3][11] show that the EMV protocol sequence can be manipulated to produce erroneous behaviour in the cards and the Point of Sale terminals.

In what follows, we first introduce the attack scenario then the technology used and finally the performance results demonstrating the practicality of the attack. A critical part of our software implementation is the ability to find and attack EMV payment cards contained in a wallet with various other contactless cards. Our software implements the ISO-14443 part 3 protocol sequence for card initialisation and anti-collision. It can identify multiple cards, select each card in turn and communicate with each card once selected.

2 Attack Scenario

The attack scenario outlined in this paper is presented as supporting evidence of our assertion that allowing contactless access to *offline* verify PIN represents a tangible threat to a large number of EMV payment cards currently in circulation in the UK.

Newcastle University, like many other companies and institutions, uses NFC enabled identity cards to control access to our buildings. When entering the building, many of us place our whole wallet on the door access reader as it is quicker and easier than taking the access card out of the wallet. This gives an attacker the opportunity to access the other cards in the wallet, communicating with any contactless payment cards also present.

Given that the person will enter the building on a regular basis and that the number of available PIN attempts is reset each time the payment card is used in a Point of Sale terminal or ATM, the attacker can have unlimited attempts to guess a card's PIN.

[2] Cardholder Verification Method is used to approve the transaction either by PIN or by signature.

In our experimental implementation of the attack scenario we make use of (i) a protocol sequence which exploits the verify PIN functionality (ii) the ability to access multiple cards in a single wallet presented to the door access reader (iii) a strategy for guessing PINs [2] which will yield greatest number of correct guesses.

2.1 PIN Verify Protocol Sequence

The full protocol sequence (Fig 1) is designed to guess the PIN without locking the card. Locking occurs when all of the available PIN attempts are used (i.e. the card is locked when the counter for PIN attempts remaining becomes zero). The protocol uses the minimum number of commands possible so that it can be completed quickly (<500ms) to avoid arousing the suspicions of the cardholder. Moreover, to avoid locking the card we need to keep at least one PIN attempt remaining on the card. The protocol sequence is therefore limited to a maximum of two guesses each time the cardholder uses the door. However, over time the attack has multiple chances to run the protocol sequence as the person will regularly return to the door access reader and each time the card is used in a Point of Sale terminal or ATM, the PIN attempt counter is reset, giving more chances for further guesses.

The PIN verify protocol sequence described above ensures that at least one PIN attempt is left on the card. However the logic can be changed to create a nuisance attack

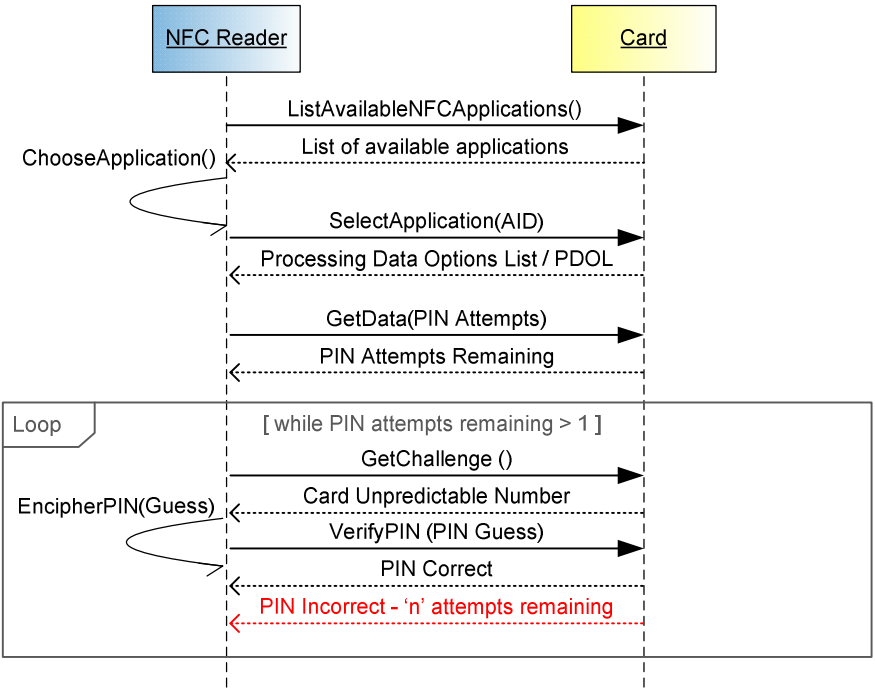


Fig. 1. - Verify PIN protocol sequence

which wipes out all of the available PIN attempts on all of the EMV payment cards in the wallet. This would not yield any financial gain, but there are many malicious attacks performed purely for the nuisance value. A card that has zero PIN attempts remaining cannot be reactivated at the Point of Sale terminal and the cardholder must go to a bank ATM.

2.2 Reading Multiple Cards

The scenario requires reader software capable of distinguishing between multiple NFC cards in a wallet, allowing it to locate the EMV payment cards (implementation details can be found in section 3.2). This also gives the potential to look for additional data such as the cardholder's birthday on the other cards in the wallet, such as loyalty cards which may hold personal data unencrypted. Bonneau et al. [2] shows that knowing the person's birthday increases the chances of guessing their PIN within 6 guesses from 1.94% to 8.23%.

2.3 PIN Guessing Strategy

The attack scenario presented accesses the card each time the cardholder enters the building. This gives it potentially unlimited guesses at the PIN over time, two guesses each time the door access is used. Bonneau et al. [2] presents a survey containing a study of 1,351 respondents, 805 of which detailed the respondents' choice of PIN and their reason for choosing it. The survey shows that 23% of respondents chose a memorable date (birthday and anniversary) as their PIN. The paper goes further and identifies a list of PINs which are statistically more likely; using this list, the paper calculates that given 6 guesses, the chance of correctly guessing the PIN is 1.94%, which rises to 8.23% if the birthday of the cardholder is known. This research is backed up by a recent news story [14] where a burglar stole a wallet in which he found a driving licence and two ATM cards, he correctly guessed the PIN from the date of birth on the driving licence and was able to obtain £1,000 from a nearby ATM.

3 Software Implementation

The experimental work in preparing this paper includes (i) an implementation of the verify PIN protocol sequence which makes multiple attempts to guess the PIN of any EMV payment card detected in the wallet (ii) a multiple card reader implementation which will identify and communicate with all of the contactless cards in the wallet.

The experiments were performed using an ACR122-U contactless card reader [1] and the Java™ Smart Card I/O API [13].

3.1 Verify PIN Implementation

The UML sequence diagram (Fig 1) illustrates the protocol sequence required to perform the verify PIN attack sequence. The sequence employs the minimum number of commands which achieve two contactless verify PIN attempts, this minimises total execution time (on average 457.2ms) for the sequence. Minimising execution time is

important to ensure that the attack is not easily detected by the cardholders using the door access.

The protocol sequence is initiated when the multiple card reader (section 3.2) detects an EMV payment card in the wallet. The protocol sequence therefore starts with the EMV payment card in the `active` state ready to accept commands (see Table 1 for a full explanation of the possible card states). Once the reader has established communication with the card, it reads the number of PIN attempts remaining using `GetData(PIN Attempts)`. It then calls the verify PIN command in a loop. The card responds with 0x9000 if the PIN is correct or 0x63Cn if the PIN is incorrect, where 'n' is the number of PIN attempts remaining. The loop is repeated until the correct PIN is guessed or only one PIN attempt remains.

We observed that the contactless PIN is the same as the contact PIN, this was confirmed by changing the card's contact PIN using an ATM and verifying that the contactless PIN had also changed.

3.2 Multiple Card Reader Implementation

EMV contactless payment cards are compliant with the ISO-14443:Part 3 which defines the disambiguation and activation sequence. Disambiguation involves obtaining the Unique Identifiers (UID) of each of the cards in the NFC field. Once this is complete, the UID is used to activate each card individually. The card is then `ready` to accept commands. For successful communication only one card can be `active` at any one time. Table 1 below describes the transitions between the different states `idle, ready, active` and `halt` which allow the reader to successfully communicate with an individual card when there are multiple cards in the field.

Table 1. ISO-14443 Card State Transitions

State	Description
`idle`	Upon entry to the NFC field all cards will power up into the `idle` state.
`ready`	The reader transmits `REQA` / `WUPA` command putting the cards into the `ready` state. Once all of the cards are in the `ready` state the anti-collision loop sequence can begin.
`active`	The anti-collision loop sequence is an iterative process used by the NFC reader to find the UID of the next card in the field. The anti-collision command is repeatedly sent to all cards until only one card answers with a complete UID and no collisions. The UID is then used in the `Select` command which moves that card into the `active` state. At this point the reader can communicate with the card using the card type specific protocol (EMV, MIFARE etc.) or instruct the card to `halt` and store the UID for future use.
`halt`	To communicate with the next card in the NFC field the reader must `halt` the currently `active` card. Cards can be re-awakened from the `halt` state using the `WUPA`.

The process of communicating with multiple cards is as follows:

1. the anti-collision loop finds the UID of each card in turn
2. Select(UID) moves the card with the given UID into the `active` state
3. the `active` card is now ready for communication with the reader, only one card at a time can be `active`
4. `halt` is used to stop communicating with the card and move to the next card

The current implementation of the disambiguation and activation sequence is compatible with all ISO-14443:Part 3 compliant cards. Once disambiguation is complete each card type has its own specific communication protocol. We have implemented protocol sequences for three commonly available card types: EMV payment cards, MIFARE classic door access cards and MIFARE DESFire travel pass cards. Communication with the implemented card types is not affected if an unknown card type is also present in the NFC field, the unknown card type is simply ignored once the disambiguation process has identified its UID. The software utilises hardware commands specific to the NXP PN532 chipset [12] to perform the anti-collision loop, disambiguation and card selection.

4 Results

The test results in this section focus on the time taken to perform each of the steps involved in performing the attack scenario presented in section 2. These results are presented to support our assertion that the delay introduced by the attack would not arouse the suspicions of the users of the door access system.

4.1 Verify PIN Protocol Sequence

Based on the data obtained in our tests the average time taken to perform the complete protocol sequence (Fig 1) was only 457.2ms; thereby strengthening the case that the door access reader attack scenario can be implemented without raising the suspicions of the users of the door access system.

Table 2. Verify PIN command execution times

Command	average (ms)	standard deviation (sec)
`ListAvailableNFCApplications()`	18.4	12.7
`SelectApplication(AID)`	19.2	5.5
`GetData(PIN attempts)`	29.8	17.9
`GetChallenge()`	24.6	7.0
`VerifyPIN(incorrect PIN)`	175.8	7.2
`GetChallenge()`	12.2	6.8
`VerifyPIN(correct PIN)`	177.2	9.6
`Complete Protocol Sequence`	457.2	24.9

The time taken to perform each of the commands in the verify PIN protocol sequence is detailed in Table 2, which shows the average time and standard deviation calculated from 20 test runs performed using EMV payment cards issued by a UK bank.

The results show that 77.2% of the total time was taken by the card responding to the `VerifyPIN()` command. It is also interesting to note that there is no significant difference between a correct PIN (177.2ms) and an incorrect PIN (175.8ms).

4.2 Multiple Card Identification

For the multiple card identification tests we used three of the more popular contactless card types: EMV payment cards, MIFARE classic door access cards and MIFARE DESFire travel pass cards. The test results in Table 3 show the average time (over 60 test runs) to identify each card, when there are multiple cards in the NFC field. Results of the tests show the identification of each card takes longer when more cards are in the field.

Table 3. Multiple Card Identification Times

Cards in NFC Field	2 cards	3 cards	4 cards	5 cards
Identification of Each Card (ms)	214.36	285.82	305.95	358.30
Standard Deviation (ms)	16.91	16.66	72.54	53.87

The maximum number of cards that the ACR-122U reader (used in our tests) can identify in the NFC field varies by card type. Table 4 shows the maximum number of each card type that the reader could identify and communicate with. The first three rows show tests with a single card type in the NFC field. The following three rows represent wallets containing a mixture of card types, with at least one EMV payment card and one MIFARE classic door access card (as the attack scenario described is based on wallets containing these two cards).

Table 4. Maximum Cards in NFC field

	EMV payment	MIFARE classic	MIFARE DESFire
Single card type	2 cards	5 cards	4 cards
Multiple card types	2 cards	1 card	1 card
	1 card	1 card	
	1 card	3 cards	

4.3 Total Attack Time

Table 5 illustrates the total time taken by the verify PIN attack on two example wallets: *wallet 1* containing one MIFARE classic door access card and one EMV

payment card; and *wallet 2* containing one MIFARE classic, one EMV and one MIFARE DESFire travel pass. The complete sequence identifies all of the cards present in the wallet and then performs two PIN guesses on the EMV payment card.

Table 5. Multiple Card Identification and Communication Time

Scenario	Identify Card (ms)	Communication (ms)	Total (ms)
wallet 1	428.73	457.20	855.93
wallet 2	643.09	457.20	1070.29

In summary, the test results (Table 3) show that it is possible to attack a wallet containing multiple card types. Moreover, Table 5 shows that for both *wallet 1* and *wallet 2*, the total attack time of around 1 second is fast enough to avoid detection by the cardholder. The attack should also delay the green light that signifies the card has been read and delay the opening of the door. This will reassure the cardholder that the system is operating normally (if a little slowly) and allows time for the attack to complete.

5 Conclusion

The attack scenario described in this paper exploits contactless verify PIN to give potentially unlimited attempts to guess the cardholder's PIN without their knowledge, this significantly increases the odds that the attack will guess their PIN correctly. The implementation work has successfully built and tested software that proves this attack scenario is technically viable. The timing tests prove that the attack protocol sequence can be performed in less than 1 second (*wallet 1*), making it possible to access the payment cards in the wallet without arousing the suspicions of the cardholder.

It is our assertion that the attack scenario and experimental implementation work presented in this paper make a compelling case that contactless verify PIN can be misused to find out the PIN of the card without the knowledge of the cardholder. This significantly impacts the underlying security assumption of the Chip & PIN payment system, that an attacker can only gain knowledge of the cardholder's PIN through the negligence or collaboration of the cardholder. Moreover, offline verify PIN is not required in the processing of contactless transactions and is therefore redundant functionality. These findings suggest that it would be prudent to remove the contactless verify PIN functionality. It would also help to educate cardholders remove their card from their wallet before placing it on a reader.

References

1. Advanced Card Systems: ACR122U NFC Reader Application Programming Interface (2011), `http://www.acs.com.hk/drivers/eng/API_ACR122U_v2.00.pdf` (accessed January 29, 2013)

2. Bonneau, J., Preibusch, S., Anderson, R.: A birthday present every eleven wallets? The security of customer-chosen banking PINs. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 25–40. Springer, Heidelberg (2012)
3. Choudary, O.S.: The Smart Card Detective: a hand-held EMV interceptor, Cambridge (2010)
4. Drimer, S., Murdoch, S.: Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. In: USENIX Security Symposium (2006)
5. EMVCo. EMV Specifications for Payment Systems, Books 1,2,3 and 4, Version 4.3 (2011)
6. EMVCo. EMV Contactless Specifications for Payment Systems, Books A,B,C-1,C-2,C-3,C-4 and D, Version 2.2 (2012)
7. Emms, M.: Practical Attack on Contactless Payment Cards. In: HCI 2011 Workshop - Heath, Wealth and Identity Theft (2011)
8. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Potential Misuse of NFC Enabled Mobile Phones with Embedded Security Elements as Contactless Attack Platforms. In: International Conference for Internet Technology and Secured Transactions (2009)
9. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones (2011)
10. MasterCard: PayPass - M/Chip Acquirer Implementation Requirements (2006)
11. Murdoch, S., Drimer, S., Anderson, R., Bond, M.: Chip and PIN is Broken. In: IEEE Symposium on Security and Privacy (2010)
12. NXP PN532 User Manual (2007), http://www.adafruit.com/datasheets/pn532um.pdf (accessed January 29, 2013)
13. Oracle: Java Smart Card I/O API (2012), http://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html (accessed January 29, 2013)
14. Willey, G.: PIN Number burglar used victims' card. Newcastle Evening Chronicle (April 27, 2012)
15. Worldwide EMV Deployment (2011), http://www.emvco.com/about_emvco.aspx?id=202 (accessed January 29, 2013)

How to Attack Two-Factor Authentication Internet Banking

Manal Adham[1], Amir Azodi[1,3], Yvo Desmedt[2,1], and Ioannis Karaolis[1]

[1] University College London, UK
[2] The University of Texas at Dallas, USA
[3] Hasso Plattner Institute, Germany

Abstract. Cyber-criminals have benefited from on-line banking (OB), regardless of the extensive research on financial cyber-security. To better be prepared for what the future might bring, we try to predict how hacking tools might evolve. We briefly survey the state-of-the-art tools developed by black-hat hackers and conclude that automation is starting to take place. To demonstrate the feasibility of our predictions and prove that many two-factor authentication schemes can be bypassed, we developed three browser rootkits which perform the automated attack on the *client's computer*. Also, in some banks attempt to be regarded as user-friendly, security has been downgraded, making them vulnerable to exploitation.

Keywords: Two-Factor Authentication, Browser Rootkits, Internet Banking, Online Banking.

1 Introduction

A lot of research has been done on the cyber-security of financial transactions, both from a cryptographic as well as from a computer security viewpoint. Due to work like [9], wholesale transactions are now relatively secure. The challenge today is to secure the PC-based transactions. We see that identity theft has been for more than a decade the most frequent white collar crime. We know that attacks against US and UK Internet banking systems have been quite successful [18]. Tools like the Zeus malware kit [17] and the SilentBanker Trojan [21], have been helping criminals perform fraudulent on-line bank transactions.

Since research should be ahead of hackers, the main goal of this paper is to predict the next generation of OB attacks and how they may worsen the already questionable security. We survey the tools hackers use and see that, attacks against on-line banking systems (OBS) are getting more sophisticated and automated. The hacker community independently worked on automation, deploying Trojans and malware kits that overpass the requirement of having humans to manually perform the fraudulent transactions [24] (the work done in [1–3] was done in 2010-2011). To demonstrate the feasibility of a fully automated attack, three advanced browser rootkits (for Firefox Browsers) were developed, targeting the Internet banking systems of NatWest, HSBC and a European Bank which we

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 322–328, 2013.

were unable to contact, (hereinafter after called "Bank A"). An attack against Barclays' system is also proposed. [1] Security of these major banks is largely dependent on the commonly used two-factor authentication schemes. By successfully circumventing and exploiting the weaknesses of such systems, we hope to raise awareness on how dangerous the current situation is.

Aside from security in the absolute sense, we consider auxiliary factors like usability. This is a critical factor that can drive a bank to lower its security threshold in order to satisfy its customers' desires (*e.g.*, Barclays).

2 Background

2.1 Attack Strategies and Techniques

We distinguish between three main Internet banking attack vectors that can be used alone or in combination. Firstly, a *Credential Stealing* attack, is where fraudsters try to gather users' credentials, either with the use of a malicious software or through phishing [10]. Secondly, a *Channel Breaking* attack, involves intercepting the communication between the client side and the banking server, by masquerading as the server to the client and vice versa [20]. Finally, a *Content Manipulation* (also called man-in-the-browser (MiTB) [16]) attack, takes place in the application layer between the user and the browser. The adversary is granted with privileges to read, write, change and delete browser's data "on the fly", whilst the legitimate user is seamlessly unaware.

A *Browser Rootkit*, is a content manipulation technique, which is capable of completely changing the browser's display and behavior [7]. It is basically a malicious *browser extension*, which are used to extend and/or customize the browser's functionality [4]. To control unauthorized installation of browser extensions, modern browsers like Firefox, have employed numerous but weak security measures, which can be easily bypassed [7].

2.2 Related Work and Attacks

A Firefox malicious extension called BROWSERSPY was developed, which once installed on the victim's machine, begins collecting personal data that is later sent to the remote attacker [22].

Zeus [17], *SilentBanker* [21] and *URLZone1* [5], are infamous Trojans, which have been successfully used against on-line banking systems to steal millions of dollars [14, 18]. They are primarily used to steal login credentials and card numbers with their security codes, but can also change transaction details on the fly. This is commonly achieved by adding form fields at the browser level. The stolen data is sent to a command and control server (C&C server) controlled by the remote attacker, who is then able to *manually* connect to the victims on-line banking accounts.

[1] NatWest, HSBC and Barclays were informed about the findings of this study.

In [24], sophisticated and automated attacks against OBSs, built on the established Zeus and SpyEye tactics are presented. In contrast with Zeus/SpyEye, these attacks offer automated bypass of two-factor physical authentication, server-side automation to conceal the attacks, techniques to avoid fraud detection, techniques to hide evidence, and finally, techniques against standard security software.

An attack against Helios 2.0 e-voting system is described in [7]. The attack exploits Adobe Acrobat/Reader vulnerabilities to install a browser rootkit on the voters' machines. This rootkit changes the selected legitimate candidate vote to the one chosen by the extension's author, and also misleads the voters to believe they have voted for their desired candidate.

3 Authentication Systems Used in Internet Banking

Currently, several banks implement an additional layer of secondary authentication, required for monetary transactions. Four common classes of authentication schemes are usually employed.

One-Time-Password (OTP) schemes, generate passwords using pseudo-random generators or mathematical algorithms in order *to authenticate the users*. However, they offer no security against man-in-the-middle (MiTM) or MiTB attacks [11]. *Bank A* uses the DigiPass GO3 token for authenticating its customers' monetary transactions [23].

In *Full Transaction Authentication* schemes, the OTP generated *to authenticate the transaction* is cryptographically bound to the transaction data. For example, *Barclays'* PINsentry device, which adopts the CAP protocol, implements this scheme when in 'Sign mode" [19]. However, since August 2011, the implementation of PINsentry has been relaxed and *favors usability* rather than security. Specifically, account holders no longer need to use PINsentry for authenticating transactions when the payee is in their beneficiary list. It is evident that pressures from unhappy customers, who are unaware of security issues, have influenced Barclays to *downgrade its security* [8, 15].

Partial Transaction Authentication is a "relaxed" form of full transaction authentication, where only part of the transaction data are required during authentication. Example implementations of these systems include *NatWest* card reader (challenge/response mode) and *HSBC* security key. Here the cryptograms generated to authenticate transactions are based only on the last four digits of the payees' account numbers [19].

Finally, some OBSs do not require authentication of monetary transactions (*i.e., No Secondary Authentication* is required). This situation is true for NatWest and HSBC customers, who can perform payment transactions to payees who have previously been authenticated and to governmental/financial institutions deemed trustworthy by the banks.

4 Automation of the Attack

Traditional black-hat hackers' software requires manual intervention to perform fraudulent transactions, limiting the damage that can be caused. In the recent

past, more automated and sophisticated attacks have been deployed [24].In the future, hackers may adapt their tools to take advantage of the weaknesses in modern OBSs and fully automate their attack. In order to demonstrate the current shortcomings of OBSs, we have implemented three browser rootkits that undermine the OB security of HSBC, NatWest and Bank A. An attack against Barclay's OBS is also proposed.

4.1 Automatic Fraudulent Transaction

Once the browser rootkit is installed on the victims' machines, it passively monitors their browsing activities, and activates once they successfully log onto the targeted OB.

In order to achieve successful fraudulent transactions, some conditions must first be met. In the case of *HSBC* and *NatWest*, the last 4 digits of the beneficiary's account number must match the respective last 4 digits of an account controlled by the attacker (as this paper is a proof of concept, the attacker's account numbers are stored in the browser rootkits. To counter traceability, different account numbers, which can be mule accounts, can be fetched from different locations). In the case of *Bank A*, a freshly generated OTP must be typed. When the necessary condition is met, the respective browser rootkit can successfully execute a fraudulent transaction by changing the transaction data "on the fly" (*i.e.*, the beneficiary's account number and amount), whilst displaying what the user expects to see (*i.e.*, the intended by the user transaction data). In order to keep the attacks undetectable for a longer period, a small percentage of the holdings in an account is taken each time.It is more likely that the victim will disregard a few missing dollars compared to a few hundred missing dollars.

At the time this study began, *Barclays'* OBS required a fresh full authenticating signature from PINsentry for every monetary transaction, thus no browser rootkit was feasible on its own to successfully attack the given system (full transaction authentication for every transaction beats browser rootkit attacks). However, the fact that account numbers in the user's beneficiary list need no signature, creates a vulnerability. Specifically, if the attacker is successful in tricking the victim to purchase or buy a fake service (*e.g.*, through scareware [13]), then the account number used by the attacker will be listed in the victim's beneficiary list. The browser rootkit will then be capable to perform fraudulent transactions to the attacker's account number, since money transfers to that account will require no signature! Similarly with *NatWest and HSBC*, money transfers to a host of businesses (deemed trustworthy) do not require a transaction verification step. In this case, the browser rootkit can be instructed to perform funds transfers to these hosts, hidden from the user's view. Although there is no direct financial gain for the attacker, this remains a vulnerability of this specific OBS and other similar.

4.2 Hiding the Attacks

The developed browser rootkits cover up all traces of their activities. Specifically, they store the information pertaining to the fraudulent transactions in browser

cookies and on the victim's hard drive (under Firefox's "Profiles" directory). All fraudulent activity is successfully hidden from the victim by first, having the browser displaying the expected by the victim account balance rather than the actual one (which obviously has less money since the browser rootkit fraudulently transfered money from it), and secondly, by removing all fraudulent activity from the victims' statements. This is achieved by exploiting DOM elements of the HTML page. The fraudulent activity is kept hidden from the user no matter when and how many times the user logs out and back in again. As long the malicious extension is installed, the attack remains hidden.

4.3 Advantages

The biggest advantage of the developed browser rootkits is that they are fully automated. Once they are successfully installed, they are capable of performing fraudulent transactions and covering their traces at the same time, without any further human intervention or instructions from a remote C&C server, as in the case of Zeus and the like Trojans [5,17,21]. This is feasible since the attacks are against *fixed* banks, with the browser rookits explicitly designed to attack the given OB systems of the respective targeting banks. In addition, the adversarial bank account numbers are stored in the browser rootkits code.

Additionally, no theft of banking information is necessary. Our browser rootkits are clever enough to remain hidden and submit when successful fraudulent transactions are possible.

Finally, in contrast with Trojans of black-hat hackers that create files and modify the registry keys in the victims' machines, our developed browser rootkits are simple browser extensions written in a few hundred lines of code. Browser rootkits are really hard detected by any antivirus program, since once installed they become a part of the browser. In contrast, OS rootkits and Trojans are much easier to detect with an updated antivirus or an Intrusion Detection System.

5 Future Work

Real deployments of two-factor authentication schemes suffer from security issues as a result of bad design, underestimating client-side attacks and lack of usability. An ideal scheme must accommodate for the trade-off between usability and security. Clearly, new ideas that focus on security with minimal user interaction are very important. For instance, in [11], a low-cost hardware token based on PIN/TAN system is proposed for protecting e-banking systems against possible situations where the adversary has full control over the user's computer. Given that a secure element (*i.e.*, something the user has) has become essential to reach the standards of modern secure authentication schemes, the question of whether we can get rid of them may no longer stand. However, whether a universal security device for all OBSs could be deployed still remains, and depends on the willingness, effort and cooperation of the financial institutions. In addition, as the use of smartphones has transformed our daily lives, it is possible

to use mobile phones to generate transaction authentication signatures. Mobile phones are in fact used as a second authentication, but the question is whether they can improve the current processes. Finally, some advances that could be addressed in a future work may be the use of Out-of-Band (OOB) communication channels [6]. In such a setting, the device is able to communicate with the authenticating servers through the secure OOB channel and display the information for confirmation.

6 Conclusions and Reflections

Two-factor authentication (excluding full transaction verification) is still inadequate to deal with browser rootkit attacks. Although the original user-unfriendly approach of Barclays shows that if criminals would automate their attacks, certain banks are ready to roll out their modifications and annul most of the attacks, the hardware/software used by most banks, as HSBC, NatWest and Bank A, may not allow them to switch quickly. We finally observe that full transaction verification may not fully address all security concerns. The information displayed on the PC including; account numbers, name, balance and transaction details, do *not* remain private! Indeed, a browser rootkit can leak all this information to an attacker who could use it to physically target rich users, use identity theft techniques, etc. To deal with this problem banks need to carefully consider what they display on the browser. Finally, this study has multiple precautions to ensure that no malware was released in the wild. No bank servers were violated. A longer version of this paper is available in [12].

Acknowledgments. We would like to thank the referees for their feedback and the shepherd of the paper Ahmad-Reza Sadeghi for his effort. We would also like to thank Hans van Tilburg and Karl Brincat from VISA, who helped us contact HSBC, NatWest and Barclays. Finally, we also thank Natasha Lewis (Director of Legal Services - UCL), Stephen Brett (External Legal Advisor) and Diran Solanke (Head of Research Contracts) for their advice.

References

1. Adham, M.: Barclays, NatWest and Halifax Internet Banking Security, with a Simulation of Browser Exploit, MSc Thesis, University College London (2011)
2. Azodi, A.: Simulation of an Attack on HSBC's Two-factor Authentication with Transaction Verification Online Banking System, MSc Thesis, University College London (2011)
3. Karaolis, I.: Automating the Hacking of Internet Banking: Simulation of an Attack for ...[5] Internet Banking, MSc Thesis, University College London (2011)
4. Mozilla Developer Centre. Extensions, `https://developer.mozilla.org/en-US/docs`
5. Chechik, D.: Malware Analysis Trojan Banker URLZone/Bebloh (September 2009), `http://goo.gl/z7YSV`
6. Chou, D.: Strong User Authentication on the Web (August 2008), `http://goo.gl/6xbky`

7. Estehghari, S., Desmedt, Y.: Exploiting the client vulnerabilities in internet E-voting systems: hacking Helios 2.0 as an example. In: Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections, EVT/WOTE 2010, pp. 1–9 (2010)
8. PistonHeads Web Forum. Barclays PINsentry, what a dumb POS (2008), http://goo.gl/wRoJV
9. Greenlee, F.M.B.: Requirements for key management protocols in the wholesale financial services industry. IEEE Communications Magazine 23(9), 22–28 (1985)
10. Jakobsson, M.: Modeling and Preventing Phishing Attacks. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, p. 89. Springer, Heidelberg (2005)
11. Li, S., Sadeghi, A.-R., Heisrath, S., Schmitz, R., Ahmad, J.J.: hPIN/hTAN: A Lightweight and Low-Cost E-Banking Solution Against Untrusted Computers. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 235–249. Springer, Heidelberg (2012)
12. Adham, M., Azodi, A., Desmedt, Y., Karaolis, I.: How To Attack Two-Factor Authentication Internet Banking (2013), http://goo.gl/YsA6j
13. Microsoft Safety & Security Centre. Watch out for fake virus alerts, http://goo.gl/YEZMT
14. Mills, E.: Banking Trojan steals money from under your nose (September 2009), http://goo.gl/tuDfJ
15. moneysavingexpert Web Forum. Stupid Barclays PINsentry Thingymajic. Can I Log Into My Online Account Without It? (2010), http://goo.gl/eqaey
16. RSA White Paper. Making Sense of Man-in-the-browser Attacks: Threat Analysis and Mitigation for Financial Institutions (2010), http://goo.gl/NRcez
17. Ragan, S.: Overview: Inside the Zeus Trojan's source code (May 2011), http://goo.gl/nsvpG
18. RiskAnalytics LLC: $70 Million Stolen From U.S. Banks With Zeus Trojan (October 2010), http://goo.gl/XkSgq
19. 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)
20. Schneier, B.: Two-Factor Authentication: Too Little, Too Late. Commun. ACM 48(4), 136 (2005)
21. Symantec. Banking in Silence (June 2009), http://goo.gl/aj61F
22. Ter Louw, M., Lim, J.S., Venkatakrishnan, V.N.: Extensible Web Browser Security. In: Hämmerli, B.M., Sommer, R. (eds.) DIMVA 2007. LNCS, vol. 4579, pp. 1–19. Springer, Heidelberg (2007)
23. VASCO. DIGIPASS GO 3 (August 2012), http://goo.gl/EmLFy
24. Marcus, D., Sherstobitoff, R.: Dissecting Operation High Roller (2012), http://www.mcafee.com/us/resources/reports/rp-operation-high-roller.pdf

CAge: Taming Certificate Authorities by Inferring Restricted Scopes

James Kasten, Eric Wustrow, and J. Alex Halderman

The University of Michigan
{jdkasten,ewust,jhalderm}@eecs.umich.edu

Abstract. The existing HTTPS public-key infrastructure (PKI) uses a coarse-grained trust model: either a certificate authority (CA) is trusted by browsers to vouch for the identity of *any* domain or it is not trusted at all. More than 1200 root and intermediate CAs can currently sign certificates for any domain and be trusted by popular browsers. This violates the principle of least privilege and creates an excessively large attack surface, as highlighted by recent CA compromises. In this paper, we present CAge, a mechanism that browser makers can apply to drastically reduce the excessive trust placed in CAs without fundamentally altering the CA ecosystem or breaking existing practice. CAge works by imposing restrictions on the set of top-level domains (TLDs) under which each CA is trusted to sign certs. Our key observation, based on an Internet-wide survey of TLS certs, is that CAs commonly sign for sites in only a handful of TLDs. We show that it is possible to algorithmically *infer* reasonable restrictions on CAs' trusted scopes based on this behavior, and we present evidence that browser-enforced inferred scopes would be a durable and effective way to reduce the attack surface of the HTTPS PKI. We find that simple inference rules can reduce the attack surface by nearly a factor of ten without hindering 99% of CA activity over a 6 month period.

Keywords: TLS, HTTPS, Certificate Authorities, Authentication.

1 Introduction

Every day, millions of Internet users rely on HTTPS to secure their online transactions against malicious eavesdroppers or tampering through man-in-the-middle attacks. HTTPS relies on a public-key infrastructure (PKI) based on certificate authorities (CAs) that are trusted by the browser. Each server presents an X.509 certificate tying its public key to its domain name. This certificate is digitally signed by a CA, which is responsible for verifying the site's identity.

CA-signed certificates cannot protect users from compromise of the CAs themselves. Several recent high-profile attacks on CAs resulted in the signing of fraudulent certificates. For instance, in 2011, an attacker breached the security of the Dutch CA DigiNotar and created certificates for dozens of popular sites, including `*.google.com` [3]. An ISP in Iran subsequently abused this latter certificate to conduct man-in-the-middle attacks against Google services.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 329–337, 2013.

Preventing DigiNotar-style attacks is difficult, because there are currently very few technical restrictions on what domains trusted CAs can sign for. Once the CA convinces a browser (or another CA) that they are trustworthy, they are given an almost unrestricted capability to vouch for any domain name they choose. This ability leads to an enormous attack surface: an attacker who compromises any one of over 1200 CAs can then impersonate any website that relies on HTTPS. This violates the principle of least privilege: DigiNotar should not have had the capability to sign certificates for Google, nor should a CA run by a small university be allowed to sign certificates for foreign government agencies. In other words, each CA's trust should come with a limited scope.

One way to limit the scope of CA trust is to designate a set of top-level domains (TLDs), such as `.com` or `.uk`, within which each CA may sign. Indeed, we present data that suggests that most CAs currently only sign certificates for sites in a small number of TLDs, and conversely, that sites in most TLDs utilize only a small set of CAs. Many CAs appear to sign exclusively for domains belonging to a single organization, and others appear to operate within a specific country, sector, or both. Although this suggests that TLD-based restrictions could be fruitful, realizing them within the existing PKI is a challenge. The X.509 name constraints extension (see Section 2) introduced the ability to explicitly declare such restrictions in new CA certificates, but has seen almost no adoption.

Rather than relying on each CA to explicitly declare a TLD scope, we explore the possibility that browser makers could *infer* such scopes without CA participation. We propose a mechanism called CAge that creates a profile of each CA based on the TLDs of publicly visible certificates it has previously signed. These restrictions can be implemented without cooperation from the CAs, at the risk that CAs will change their behavior over time and begin signing for certificates outside their previous pattern. Empirically, we find that this rate of change is quite low, that inferred scopes generated with simple algorithmic rules would result in a low false-positive rate, and that the CAge approach would allow browser makers to dramatically reduce the attack surface of the HTTPS PKI.

For further details, see the full version of this paper, which is available online at https://jhalderm.com/papers/.

2 Related Work

There have been several prior proposals for addressing CA shortcomings [3,11,14]. Multi-path probing [2,10,13] has been suggested as a way to reduce reliance on CAs; however, it necessitates the availability and access to trusted notaries. Browser extensions have also been proposed to pin previously seen certificates or CAs to domains [6,9,12].

Scopes on CA signing have previously been proposed through X.509 Name Constraints [4], a certificate extension with the ability to restrict CAs to a particular set of domains. However, this approach has yet to see significant adoption due to several practical impediments, including lack of direct browser authority

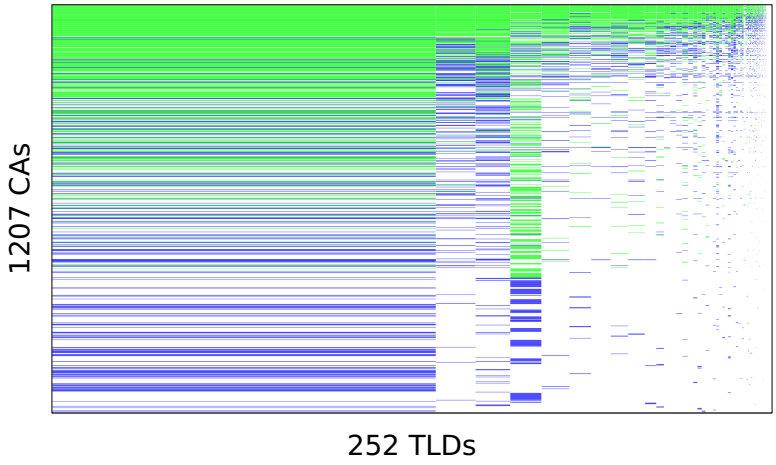


Fig. 1. This matrix shows which trusted CAs have signed certs for at least one domain (blue) or at least ten domains (green) in each TLD. Columns are scaled by fraction of valid certs (left is `.com`). Note sparseness of CA signing practices.

over intermediate CAs, the long lifetime of root CA certs, and the need to replace site certs when an issuing CA implements constraints. In contrast, CAge can be applied immediately by browsers without CA collaboration.

3 Analyzing the CA Infrastructure

Currently, important aspects of browser trust behavior and CA signing practices are surprisingly opaque to outside observers. Certificate chaining conceals the full set of trusted CAs by including an unknown number of intermediate authorities. Furthermore, CAs typically do not publish the domains for which they have issued certificates, obfuscating their signing patterns.

To understand these aspects of the HTTPS PKI, we analyzed a large corpus of certificates collected for another recent study [7] using an Internet-wide scan of HTTPS servers. We determined which certificates would be trusted by major web browsers and extracted the set of trusted CAs. See the full paper for details.

The number of certificates signed by each CA varied considerably; the top 20 CAs were responsible for more than 80% of valid certificates. Over 90% of all signed `.com` domain names used certificates issued by just 25 CAs.

Despite this lopsided distribution of CA size, 1207 CAs had the ability to issue trusted certificates for any domain name. To examine how much of this authority each CA exercised, we extracted the set of domain names that each CA had directly issued certificates for, and then examined the set of TLDs to which these domains belonged. We find that 89% of CAs had signed for domains in fewer than 10 unique valid TLDs [8], with the majority (65.8%) of CAs signing for domains in either zero or one TLD.

Although `.com` accounts for 51% of signed domains, fewer than 35% of trusted CAs had signed a certificate for even a single `.com` domain, and only 20% had signed for 10 or more such certificates. There were 787 CAs that had *never* signed for a `.com` domain. Similarly, fewer than 11% of CAs had signed certificates in the `.uk` TLD, and only 6.6% had signed for 10 or more in the `.uk` domain.

Many CAs belong to private companies and organizations and are used for domains under their control. More than 200 German universities and research institutions control browser-trusted CAs, as do corporations such as Ford, Disney, and Wells Fargo. We observed that such CAs generally limit their public signing practices to a few specific second-level domains. Other smaller CAs appear to focus their business within a specific geographic region and tend to sign domains under a country-specific TLD.

4 Our Proposal

In this section, we propose CAge, a browser-based approach that restricts CA signing to TLDs in which they have *already* signed. CAge consists of two phases: In the initialization phase, we collect certificates from an Internet-wide scan and infer rules from the observed current CA signing practices. Browsers then apply these rules in the enforcement phase to restrict CAs to the inferred scopes and handle exceptions. See the full paper for more details and for a description of our browser extension prototype.

4.1 Initialization and Rule Inference

Prior to deploying CAge, the browser maker needs to develop an initial set of restricted scopes to apply to existing CAs; however, creating justifiable rules for existing CAs necessitates knowledge of current CA practice. A comprehensive survey of public HTTPS servers (like that completed by Heninger et al. [7]) can be performed to determine the observable list of intermediate CAs and the domains for which they have directly signed certificates.

After scanning and collecting the raw data, we infer rules and restrictions for the CAs, based on current practices. As stated earlier, there are many CAs that have never signed for particular top-level domains. If a user is later presented such a certificate, this may indicate that the certificate is fraudulent, and the user should be alerted. As a first approach, CAge can generate the inferred scopes by looking at the TLDs that each CA has previously signed for. Under the simplest form of this approach, the inferred rules will allow a CA to sign for domains in a given TLD only if that CA has signed for a domain in that TLD before.

Rules are stored for each CA in the form of a set of regular expressions that governs the domains the CA is trusted to sign. This allows for the rule inference to be improved with more sophisticated algorithms in the future. In general, rule inference should be generated from an algorithm taking the CAs and their signed domains as input and producing the CA restrictions as output. CAs could be constrained to second-level domains or more specific rules could be required

for larger TLDs, factoring in the cost of false positives and both the size and brittleness of the rule set.

4.2 Enforcement and Exception Handling

Once CAge has inferred CA signing rules from the collected scans, CAge relies on browsers to enforce these rules during certificate validation. Browsers have a strong incentive to protect their users from fraudulent certificates, making them a natural place to enforce these restrictions.

Normally, browsers verify that HTTPS certificates have a valid signed chain to a trusted root. With CAge, browsers additionally compare the domain to the set of regular expression rules inferred for that certificate's intermediate (signing) CA. If the domain does not fall within the allowed rules for the given CA, CAge alerts the user with a warning explaining that the website's origin is certified by an unusual source. CAge also asks the user if they want to send the violation to the browser developers for further inspection. This feedback allows the browser to potentially verify the authenticity of the certificate via other means, while respecting the privacy of its users.

4.3 Updating

Keeping the rule set accurate and current is crucial to keeping a low false positive rate and avoiding user habituation to clicking through warning messages. The CAge rule set must be updated as CA policies change and new CAs emerge. Luckily, browser makers are in a good position to provide updates to users, based on newly discovered certificates reported collectively by users. Updates to the CAge rules can also be pushed to users through browser update mechanisms.

The update mechanism must be carefully designed to avoid being gamed by attackers. For example, we might be tempted to regenerate the inferred rules based on any newly signed domain. However, in that case, an attacker who compromised a CA that was not allowed to sign for a domain in `.com` could simply purchase a certificate from that CA for a `.com` domain the attacker legitimately controlled. The inferred TLD rules would then update to allow this CA to sign for `.com`, and the attacker could use their compromise to sign for other `.com` domains fraudulently.

While CAge would still protect users from illegitimate certificates signed by CAs that do not sign publicly (including private organizations, root CAs and inactive intermediates), attackers can still try to increase the scope of all publicly signing intermediate CAs. For this reason, we propose that the CAge rule set should be updated on a per-domain basis. When a domain exception is reported to browsers, the domain should be added to a "watchlist" where the domain can be manually vetted before the specific certificate is whitelisted and pushed as an update. We show in Section 5.2 that these updates are infrequent and thus enable manual inspection and verification.

Over the long term, new CAs, without any recorded behavior, can be added by browsers after interrogating the CAs about their intended scope and policies.

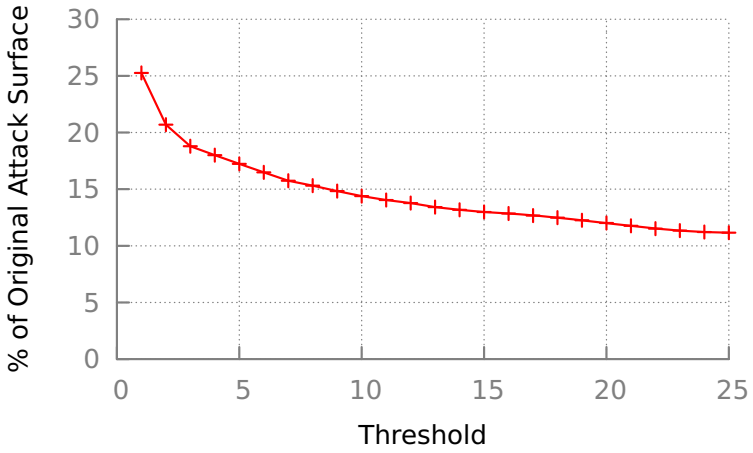


Fig. 2. HTTPS PKI attack surface under the basic CAge inference policy, compared to current practice. Even this simple approach achieves 75–90% reduction.

While this might pose an additional hurdle for new CAs entering the market, ultimately, the authority to say if a particular CA is trusted or not lies with the browser, and users' security interests demand a high level of scrutiny.

5 Evaluation

5.1 Attack Surface Reduction

While restricted scopes would reduce the attack surface across a large number of CAs, they are most effective against small and private CAs. In the Comodo attack that occurred in March 2011 [11], an attacker issued fraudulent certificates for `.com` domains signed by "CN=UTN-USERFirst-Hardware", a relatively large CA which had signed over 25,000 other `.com` certificates previously. Due to these signing practices, CAge would have been unable to protect against the Comodo attack. Similarly, all but two of the top 20 CA certificates have signed domains from over 100 unique TLDs, limiting the usefulness of restricting these large CAs to the TLDs they currently sign.

However, the vast majority of CAs do not sign for such a diverse market, allowing CAge to provide protection during a CA compromise. For instance, CAge would have detected the DigiNotar compromise. The EFF's SSL Observatory [5] data, which was collected a year before the attack, shows that the issuer of the fraudulent `*.google.com` certificate, "DigiNotar Public CA 2025" [1], had not signed certificates for any `.com` domains. Had CAge been implemented at the time, it would have prevented the attack against Internet users in Iran.

In light of these conflicting case studies, we attempt to quantify what kinds of attacks CAge would or would not protect against by developing an attack

surface metric. The goal of this metric is to quantify the relative risk of damage that could be caused by an attacker-compromised CA. We compare this metric under different scenarios; namely, current CA practice (all CAs can sign for all domain names) versus CAge (CAs are restricted to a particular set of domains).

We approximate the attack surface by $\sum_{c \in CAs} domains[c]$. The function $domains[c]$ is the number of existing, validly signed domains for which a certificate signed by a given CA c would be trusted under a given set of policies. (Intuitively, the number of signed valid domains represents the number of protected entities on the Internet.) Under current practice, $domains[c]$ is constant across all CAs and is simply the number of signed valid domains in existence. Under CAge, $domains[c]$ is reduced to domains in TLDs that are allowed under the inferred trust scope for c. For example, if a CA is trusted to sign for only `.com` because it previously signed for 100 of the 1.3 million `.com` domains (and zero domains under other TLDs), then $domains[c]$ for that CA would be 1.3 million.

While this attack surface metric is by no means complete, it provides a first-order approximation that allows us to quantitatively compare the risks of different CA restriction policies. Applied to our data set, the simple CAge rule set inference method described in the previous section yields an attack surface that is 75% smaller than current practice.

We can improve this result by modifying the inference procedure to only allow a CA to sign for domains in a TLD if it has previously signed for a minimum threshold t of unique domains in that TLD. If a CA has signed fewer than t domains in a particular TLD, these domains can either be viewed as suspicious anomalies or whitelisted as individual rules within the rule set. Applied to our scan data with $t = 25$, this policy would reduce the attack surface by 89% compared to current practice.

5.2 Rule Set Durability

Although the attack surface metric provides a quantifiable goal, reducing it is not our only objective. The minimum attack surface would be achieved by pinning every observed domain to the CA that signed its cert, but the result would be an enormous rule set that would require constant updating and lead to an impractical number of false positives. CAge must instead attempt to capture CAs' actual signing policies so as to produce rules that are compact and stable.

In order to test the durability of our inferred rules, we acquired a second scan in April 2012 (6 months after the original scan). Focusing on changes during this interval, we found that the large majority of domains observed in newly issued certificates conformed to our rules, supporting our hypothesis that the TLDs that CAs sign for are generally static. The basic policy, restricting CAs to TLDs they have signed in the past, accommodated 99.84% of new certificates. Most of the 1506 violations that occurred were in unpopular or small TLDs. See the full version of this paper for additional analysis.

6 Conclusion

In this paper, we presented CAge, a mechanism for inferring TLD-based restricted scopes for HTTPS CAs. Based on the empirical observation that the vast majority of browser-trusted CAs do not utilize their unconstrained signing power, we argue that each CA should be restricted to signing for domains within a limited set of TLDs. We show how restrictions can be realized in practice by profiling past CA behavior, and we find that such an approach would dramatically reduce the attack surface of the HTTPS PKI without a high rate of false alarms over time.

While browsers have a positive record of revoking compromised CA certificates once a breach is discovered, we believe much more can be done to proactively mitigate the damage caused by attacks against CAs and to provide defense-in-depth to the HTTPS PKI. Given the relative ease with which CAge could be deployed by browsers, we strongly encourage browser developers to adopt this approach to help combat the growing threats that HTTPS users face.

Acknowledgements. The authors gratefully acknowledge Zakir Durumeric for providing HTTPS certificate data for this study, and we thank the anonymous reviewers for their constructive comments and feedback. This work was supported in part by the National Science Foundation (NSF) under contract numbers CNS 1255153 and DGE 0654014 and an NSF Graduate Research Fellowship.

References

1. Gmail.com SSL MITM Attack by Iranian government (August 2011), `http://pastebin.com/ff7Yg663`
2. Alicherry, M., Keromytis, A.D.: Doublecheck: Multi-path verification against man-in-the-middle attacks. In: ISCC, pp. 557–563. IEEE (2009)
3. Bhat, S.: Gmail users in Iran hit by MITM Attacks. Website (August 2011), `http://techie-buzz.com/tech-news/gmail-iran-hit-mitm.html`
4. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard) (May 2008)
5. EFF. The EFF SSL Observatory, `https://www.eff.org/observatory`
6. Evans, C.: New Chromium security features (June 2011), Website, `http://blog.chromium.org/2011/06/new-chromium-security-features-june.html`
7. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your Ps and Qs: Detection of widespread weak keys in network devices. In: Proceedings of the 21st USENIX Conference on Security Symposium, Security 2012, p. 35. USENIX Association, Berkeley (2012)
8. IANA. Top level domains, `http://data.iana.org/TLD/tlds-alpha-by-domain.txt`
9. Loesch, C.: Certificate patrol. Website, `http://patrol.psyced.org/`
10. Marlinspike, M.: SSL and the future of authenticity, BlackHat USA (August 2011)
11. Richmond, R.: Comodo fraud incident (March 2011), `http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`

12. Soghoian, C., Stamm, S.: Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 250–259. Springer, Heidelberg (2012)
13. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: Improving SSH-style host authentication with multi-path probing. In: USENIX 2008 Annual Technical Conference, pp. 321–334. USENIX Association, Berkeley (2008)
14. Zusman, M.: Criminal charges are not pursued: Hacking PKI, DefCon 17 (August 2009)

Interdependent Privacy: Let Me Share Your Data

Gergely Biczók[1] and Pern Hui Chia[2]

[1] Dept. of Telematics
Norwegian University of Science and Technology
gbiczok@item.ntnu.no
[2] Centre for Quantifiable Quality of Service (Q2S)
Norwegian University of Science and Technology
chia@q2s.ntnu.no

Abstract. Users share massive amounts of personal information and opinion with each other and different service providers every day. In such an interconnected setting, the privacy of individual users is bound to be affected by the decisions of others, giving rise to the phenomenon which we term as *interdependent privacy*. In this paper we define online privacy interdependence, show its existence through a study of Facebook application permissions, and model its impact through an Interdependent Privacy Game (IPG). We show that the arising negative externalities can steer the system into equilibria which are inefficient for both users and platform vendor. We also discuss how the underlying incentive misalignment, the absence of risk signals and low user awareness contribute to unfavorable outcomes.

Keywords: Interdependent Privacy, Application Permissions, Facebook Apps, Externality, Incentive Misalignment, Game Theory.

1 Introduction

In today's networked world, users of online services have become logically interconnected in many ways. Such relationships often involve sharing personal information or opinion via named or unnamed user accounts or pseudonyms. Privacy concerns arise along with data sharing. In such an intertwined setting, the privacy of individual users is bound to be affected by the decisions of others, and could be out of their own control. This gives rise to the phenomenon which we term as *interdependent privacy*. While there is a plethora of online services where privacy interdependence matters, including the blogosphere, forums, photo and video sharing portals, the low-hanging fruits in this context are Online Social Networks (OSNs). A particularly interesting example is Facebook and its platform for third-party apps. Through its app platform, Facebook provides an efficient way to create a lock-in effect for users. However, the structure of the permission system associated with the platform raises some questions [1].

In this paper we take a first step towards understanding interdependent privacy. Our contribution is threefold. First, we define the concept of interdependent

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 338–353, 2013.

privacy in the context of today's networked society. Second, through a study on third party Facebook apps, we point out the permissions causing privacy interdependence, and quantify their prevalence. Third, we present the Interdependent Privacy Game (IPG), and show how positive (network effect) and negative (privacy loss) externalities can shape the behavior of social network users with regard to app usage. Our main finding is that equilibrium outcomes may be inefficient and contrary to best interest of users and/or platform vendor. We discuss how the underlying incentive misalignment, the absence of risk signals and low user awareness contribute to the unfavorable outcome, and hint on designing a possibly better application installation mechanism by mitigating negative externalities. Our intention is to introduce privacy interdependence to the research community, and outline interesting future research directions of both theoretical and experimental nature.

The paper is organized as follows. Section 2 explains the notion of interdependent privacy, while Section 3 exemplifies privacy interdependence on the Facebook application platform through a measurement study. Motivated by the Facebook case study, Section 4 presents a game theoretical model of interdependent privacy and analyzes its potential equilibria. We discuss our findings in Section 5. Section 6 shortly describes related work. Finally, we provide a summary and potential topics for future research in Section 7.

2 Interdependent Privacy

An early definition given by Clarke [2] was that privacy is the interest that individuals have in sustaining a 'personal space', free from interference by other people and organizations. Clarke further outlined four dimensions of privacy: bodily, behavioral, communication and data privacy.*

Online Privacy. As the digital world evolves, and changes online user behavior and expectations, there is no single widely accepted definition of online privacy today. Yet, adapting from Clarke's categorization, we can structure online privacy risks in three dimensions:

- **Personal:** Potential loss of information about a user and his behavioral data.
- **Relational:** Revelation of how a user relate to and communicate with others.
- **Spatial:** Invasion of the virtual space of an online user (e.g. uninvited postings on the user's blog and social media spaces).

* Bodily privacy concerns the integrity of the individual's body including issues such as blood transfusion without consent, and compulsory submission of body fluids or tissues. Meanwhile, behavioral privacy concerns all aspects of human behaviors including sensitive information such as sexual preferences, political activities and religious practices. On the other hand, communication privacy demands for the ability to communicate with intended targets without routine monitoring by others, while data privacy concerns the protection of personal data, and the ability to exercise control over data that is to be processed by others [2].

There exists a rich literature on the protection of online personal and relational privacy. Spatial privacy is another important subject as virtual spaces including blogs and social media spaces (e.g., Facebook's user timeline, LinkedIn's user profile) are being claimed by and associated with the users.

Privacy Interdependence and Externality. Rather than focusing on protecting each of the 3 online privacy dimensions from malicious actors, we present in this article an important aspect yet to be adequately addressed in the community – *privacy interdependence*. Indeed, the protection of personal, relational and spatial privacy of individuals is increasingly dependent on the actions of others, rather than the individuals themselves, in the interconnected digital world.

The interdependence in online privacy is perhaps not a new phenomenon. Alice could easily embarrass Bob by taking and sharing a "funny" photo of Bob through conventional mediums such as posters, emails or blogs. Yet, the advent of online social networking services has made data sharing much easier across networks of users and thus a higher concern for privacy interdependence. How appropriate it could be for an OSN service to allow a user to share an information concerning or on behalf of another user, based on their relationship, for an improved user experience?

Sharing a user's information without his direct consent can lead to the emergence of externalities. We know from [3] that an externality arises when an entity engages in an activity that influences the well-being of a bystander and yet neither pays nor receives any compensation for that effect. If the impact on the bystander is beneficial, it is called a positive externality. On the contrary, a negative side-effect is termed as negative externality. While sharing someone else's information may yield benefits for her (e.g., personalized experience), it is also almost certain to cause a decrease in her utility (i.e., loss of online privacy).

A straightforward example of privacy interdependence in OSN is with photo tagging. Consider the case where Alice tags Bob in a photo and shares it in the OSN without explicit consent from Bob. Both Alice's and Bob's friends gain access to the photo in the default setting. Another excellent example of privacy interdependence is exemplified by the Facebook application platform. How well a user can protect his privacy from third party developers depends not only on his decisions, but also the decisions of his friends. We leverage the case of Facebook applications as the primary example in this article.

3 Case Study: Facebook Application Platform

The Facebook Help Center [4] describes why apps need to access user information before she can use them. As expected, it says that apps look to maximize user experience by collecting personal information. Common ways of using this information are: helping you find friends using the same app, personalizing content, aiding content sharing and quick bootstrapping of required data. It is also stated that apps are not allowed to use information for advertisements or transfer user information without your consent. We will show in this section that the last statement is not entirely valid.

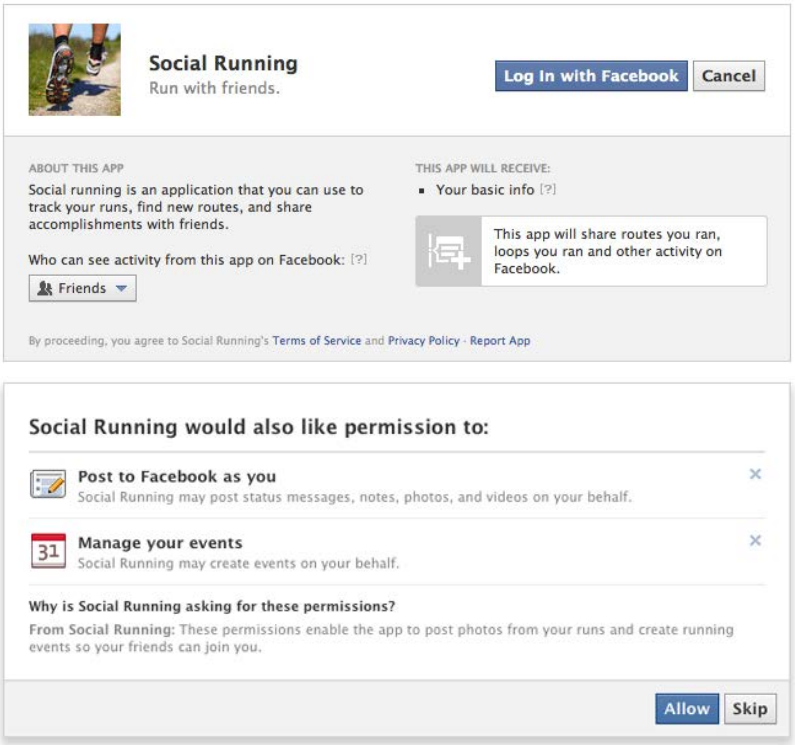


Fig. 1. The new permission request dialog of Facebook – Enhanced Auth Dialog. (Top) First screen displays the non-extended permissions requested by the app. (Bottom) Second screen shows the list of extended permissions requested, and an optional description by the developer to justify the need for extended permissions. Extended permissions can be individually rejected by the user.

Facebook relies on permission-based platform security to apply the least privilege principle to third party applications on the platform. Similarly to the Android mobile platform, third party Facebook apps wanting to access specific user information or account features are required first to ask for user consent to grant the relevant permissions. Chia et al. termed this as a user-consent permission system, as opposed to the centralized permission model of iOS where Apple decides which permissions can be requested by third party apps [5].

Application Permissions on Facebook. Facebook has a total of 65 permissions as of June 2012. They are categorized into 5 different types: basic, user or friend information, extended, open graph, and page permissions [6]. Towards the users, Facebook however distinguishes only between non-extended and extended permissions since early 2012. As shown in Fig. 1, the newly introduced Enhanced Auth Dialog of Facebook presents extended permissions to the users more prominently (on the second screen) as compared to the non-extended permissions (on the first screen).

We are most interested in investigating the interdependence aspect of different app permissions. Table 1 shows the three dimensions of online privacy risks affecting the user himself and his friends (hence the negative externality). Indeed, privacy control with app permissions on Facebook depends not only on the user's own due diligence, but also the discipline of his friends. The table shows the number and percentage of apps exhibiting a particular privacy risk, derived from a data set of 27,029 apps constructed by Chia et al. [5]. The data set was constructed by first downloading the list of all Facebook applications on `socialbakers.com`, and then visiting each of the apps to save the list of permissions requested at install-time.

Personal privacy concerning the potential leak of user interests, birth date, education history and political views can depend on the user's own, or any of his friends' decisions to install a third party application. Facebook has 24 permissions as shown in Table 2 (right), which allow an app to obtain not the personal information of the user himself, but that of his friends. 1.92% of apps request for friends' personal information. While it is a smaller sum compared to 17.15% of apps requesting for the user's own personal information (excluding those that asks for only the `basic` permission), we believe that a majority of users are unaware of the privacy externality (or control dependency) with Facebook app permissions (see [7] for a different angle).

More than just friendship links, we consider relational privacy to include the conversations (chats, messages) and joint events between two users. Relational privacy thus affects both parties, and its protection depends on the actions of both. Installation of an app requesting a set of relational permissions, as listed in Table 3, can reveal the relation between the user and his friends. Note that the `basic` permission reveals the list of friends for a user, while `read_friendlists` reveals the custom lists of friends (e.g., close friends, band members, colleagues, relatives) that a user has made. The permission `manage_friendlists` further allows an app to edit the custom lists of friends. At the same time, `xmpp_login` exposes private chat messages, while `read_stream` allows an app to read the less private messages such as postings by friends onto user's timeline. Excluding the `basic` permission, 1.75% of apps pose the risk of relational privacy breach.

The third dimension is spatial privacy. Again, the protection of the user's digital space depends on his own and his friends' decisions. Third party apps with `publish_actions` and `publish_checkins` permissions can post to the user's own

Table 1. Online privacy dimensions, dependency of privacy control (equivalently, the affected victim), and the number of apps posing the respective risks. Figures in [brackets] exclude apps that request only the single `basic` permission.

Dimension	Dependency (Affecting)	# app	% app
Personal	Self	18204 [4634]	67.35 [17.15]
	Friends	518	1.92
Relational	Both self and friends	18204 [480]	67.35 [1.78]
Spatial	Self	494	1.83
	Friends	6249	23.12

Facebook timeline. Meanwhile, `publish_streams` has been designed to be the superset permission of `publish_actions`, allowing an app to post also onto the friends' spaces. This single permission of `publish_streams`, which has been requested by 23.12% of apps, is the main culprit for uninvited and often disgraceful postings on a user's timeline or feeds, including invitations from spammy apps or obscene postings. We regard this as a violation of spatial privacy.

Table 2. Facebook permissions with implications to personal privacy; control lies with the user himself (left), or depends on the decisions of his friends (right)

Permission	# app	% app
basic	18204	67.35
email	3766	13.93
user_about_me	284	1.05
user_activities	67	0.25
user_birthday	914	3.38
user_checkins	24	0.09
user_education_history	67	0.25
user_events	27	0.10
user_games_activity	5	0.02
user_groups	35	0.13
user_hometown	204	0.75
user_interests	94	0.35
user_likes	314	1.16
user_location	412	1.52
user_notes	12	0.04
user_online_presence	67	0.25
user_photos	574	2.12
user_questions	-	-
user_relationships	77	0.28
user_relationship_details	21	0.08
user_religion_politics	50	0.18
user_status	131	0.48
user_subscriptions	-	-
user_videos	187	0.69
user_website	12	0.04
user_work_history	107	0.40

Permission	# app	% app
friends_about_me	25	0.09
friends_activities	23	0.09
friends_birthday	162	0.60
friends_checkins	15	0.06
friends_education_history	30	0.11
friends_events	7	0.03
friends_games_activity	5	0.02
friends_groups	8	0.03
friends_hometown	44	0.16
friends_interests	33	0.12
friends_likes	51	0.19
friends_location	62	0.23
friends_notes	3	0.01
friends_online_presence	89	0.33
friends_photos	256	0.95
friends_questions	-	-
friends_relationships	19	0.07
friends_relationship_details	8	0.03
friends_religion_politics	20	0.07
friends_status	16	0.06
friends_subscriptions	-	-
friends_videos	75	0.28
friends_website	2	0.01
friends_work_history	29	0.11

4 The Interdependent Privacy Game

Motivated by our measurement findings, we propose a game-theoretic model called the Interdependent Privacy Game (IPG). While IPG is general enough to model most decision scenarios involving privacy interdependence, here we choose to focus on the inter-user effects of app installations on Facebook. We concentrate on the 2-player-1-app case, while also providing some insight for multiple players and apps. The main feature of IPG is the possibly simultaneous emergence of both positive and negative externalities, and the effect of this phenomenon on the stable outcome.

Table 3. Facebook permissions with implications to relational privacy. Control depends on both the user himself and his friends. The `basic` permission reveals the list of friends, while `read_friendlists` exposes the custom lists of friends of the user.

Permission	# app	% app
basic	18204	67.35
read_friendlists	114	0.42
read_mailbox	1	0.00
read_requests	5	0.02
read_stream	356	1.32
rsvp_event	12	0.04
xmpp_login	14	0.05
manage_friendlists	1	0.00
manage_notifications	7	0.03

Table 4. Facebook permissions with implications to spatial privacy. Control lies with the user himself (top), or his friends (bottom). The `publish_actions` permission allows an app to post to the user's spaces (wall, timeline) while `publish_stream` allows an app to post to friends' spaces.

Permission	# app	% app
publish_actions	485	1.79
publish_checkins	9	0.03

Permission	# app	% app
publish_stream	6249	23.12

4.1 Game Structure

Assumptions. We assume that the players are non-cooperative. Players have a connection in the social network, hence they are "friends". We only consider apps that ask for permissions affecting the privacy of the friends of the user. We assume that these permissions are independent of the set of permissions requested by the app directly from the user. For tractability reasons, we focus on the scenario of two players and a single app.

Players. IPG is played by two players. Players embody users of Facebook, who also have an established friend connection.

Strategies. In the simplest form of the game, the decision is whether to install or not to install the given single app. Formally, the strategy space is $S = \{i, n\}$ for both players. Mixed strategies are probability distributions over these (pure) strategies.

Payoffs. The main characteristics of IPG is that both *positive* and *negative externalities* could emerge from the decisions of the two players. The positive externality is the so-called *network effect*: having more users install the same app could actually improve user experience [4]. On the other hand, negative externality can emerge as the other user's decision to install an app imposes privacy risks to this user. As detailed in Section 3, Facebook's app permission model exhibits characteristics of interdependent privacy. Many of the app permissions poses personal, relational or spatial privacy risks to the friends of an app user. Adding to positive and negative externalities, users also have their own valuations of each app. The valuation v can be positive (e.g., fun, useful) or negative (e.g., useless, waste of time, buggy); note that v represent the app's value without network effect, and therefore is independent from other parameters. Formally:

$$\pi_{(s_1,s_2)} = f\left(v, e^+\left(s_1, s_2\right), e^-\left(s_1, s_2\right)\right), \tag{1}$$

Table 5. Payoff matrix for IPG

	(I)nstall	(N)ot
(I)nstall	$(v + e^+ + e^-, v + e^+ + e^-)$	(v, e^-)
(N)ot	(e^-, v)	$(0, 0)$

where s_1 (s_2) is the strategy that player 1 (player 2) plays, $v \in R$ is the user's own valuation for the app, while $e^+ > 0$ ($e^- < 0$) represents the positive (negative) externality. Note that we assume identical players with respect to valuation of both app and externalities, therefore the payoff matrix is symmetric. Payoff values are shown in Table 5.

It is important to emphasize that the positive externality emerges only when both players choose to install the app, while the negative externality appears if either one of them does. Also, the signum of the payoff has a more important message than the exact value; especially, since user valuations are hard to estimate/measure.

4.2 Analysis

In such a 2-player matrix game with parametric payoffs, the equilibrium depends on the relation of payoffs for the possible 4 outcomes. In fact, 4 inequalities (1 for each neighboring outcome-pair) are satisfactory to describe these relations. These are:

$$\pi_{ii} > \pi_{in} \quad \text{iff} \quad e^+ + e^- > 0 \tag{2}$$

$$\pi_{ii} > \pi_{ni} \quad \text{iff} \quad v + e^+ > 0 \tag{3}$$

$$\pi_{in} > \pi_{nn} \quad \text{iff} \quad v > 0 \tag{4}$$

$$\pi_{nn} > \pi_{ni} \quad \text{always.} \tag{5}$$

It can be seen that there are 8 possible cases with regard to the three parametric inequalities. Since we aim to explore the solution space, we assign a single bit to each inequality in the order of Eq. (2)-(3)-(4), and set it to 1 if the inequality holds and to 0 if it does not. After checking for conflicts, we find that Case 101 and 001 are not feasible. This leaves us with 6 potential cases.

Possible Nash Equilibria. Let us identify Nash equilibria (NE) on a case-by-case basis. The first three cases have intuitive outcomes, and a simple NE. We also characterize Pareto-optimality (PO) and social optimality (SO).

- *Case 111.* Unique NE: (i, i). This is an intuitively beneficial outcome: direct valuation is positive, and positive network effects are stronger than the privacy loss; the result is mutual installation. The NE is both PO and SO.

- *Case 110.* Unique NE: (n, n). Strong positive externality cannot overcome the negative direct valuation resulting in not installing the app. The NE is both PO and SO.
- *Case 000.* Unique NE: (n, n). This is the mirror image of Case 111: negative direct valuation and strong negative externality result in not installing the app. The NE is both PO and SO.

In the following case, IPG turns into the classic *prisoner's dilemma (PD)*, when the payoff is negative in equilibrium. Note that cooperating in the original PD is analogous to not installing in IPG.

- *Case 011.* Unique NE: (i, i). When $v < |e^- + e^+|$, the payoff is negative in the NE. Under this condition, the relation of payoffs becomes $\pi_{in} > \pi_{nn} > \pi_{ii} > \pi_{ni}$, which in turn leads to a PD-type game. Hence, the NE is neither Pareto- nor socially optimal. The strategy profile (n, n) is both PO and SO. Putting it differently, users will install the app because they fear that the other player would possibly inflict a negative externality on them.

In the last two cases IPG turns into a *coordination game*.

- *Case 110.* Three possible NE: (i, i), (n, n) and a symmetric mixed NE; in the mixed NE players play i with a positive probability $p = \frac{-v}{e^+}$, while they play n with a probability of $1 - p$. The strategy profile (i, i) is PO and SO if $|e^+| > |v| + |e^-|$; (n, n) is PO and SO otherwise.
- *Case 010.* Three possible NE: (i, i), (n, n) and a symmetric mixed NE; in the mixed NE players play i with a positive probability $p = \frac{-v}{e^+}$, while they play n with a probability of $1 - p$. The strategy profile (n, n) is both PO and SO. All three other cases result in negative payoffs. In fact, (i, i) could yield the worst possible aggregate social payoff (sum of the 2 players), if $|e^-| - e^+ > |e^+| - |v|$. This could be interpreted as the following: users can punish each other with installing the app, even if their own valuation v and their total payoff is negative.

Evolutionary Stability. In order to view the IPG from a different angle, here we apply the evolutionary perspective. We think this is particularly fitting in the case of social networks, since organisms (persons) of the same population (users) do interact (establish connections, comment, chat) with each other, and user behavior could be considered inherent regarding certain actions ("genetic" strategies). While we can consider the same strategies as above, the solution concept here is Evolutionarily Stable Strategy (ESS). ESS is analogous to NE in an evolutionary setting: a genetically determined strategy that tends to persist once it is prevalent in a population [8]. On the other hand, ESS is a refinement of NE, and relies on a stricter definition of equilibrium. Due to this, an ESS is always a NE, but a NE is not always an ESS.

Consider the general, symmetric game shown in Table 6. A well-known result for ESS is that in a two-player. two-strategy, symmetric game, S is an ESS iff (i) $a > c$ or (ii) $a = c$ and $b > d$ [8]. Now, putting it into the context of IPG,

Table 6. General symmetric game

	S	T
S	(a, a)	(b, c)
T	(c, b)	(d, d)

let $S = I$ (install) and $T = N$ (not install). Further along, this makes $a = \pi_{ii}$, $b = \pi_{in}$, $c = \pi_{ni}$ and $d = \pi_{nn}$.

The relations between payoffs of the two strategies are different for the 6 cases described above. We now determine in which cases the ESS conditions are satisfied with regard to strategy I (install); these are 111, 110, 011 and 010, respectively. Interestingly, with the exception of 111, these are exactly the controversial or uncertain cases going back to the prisoner's dilemma and coordination games. Put plainly, both players installing the app is an equilibrium in a stricter sense (ESS is stricter than NE) in the exact scenarios, where this is contrary to the individual and aggregate interest of players (inefficient equilibrium).

Multiple Players and Apps. When considering more than two players, an interesting phenomena arises. First, the positive externality is getting stronger with the increasing number of users installing the same app (network effect). On the other hand, the maximum impact of the negative externality is already present with a single other user deciding for installing the app: the privacy loss is already there. Also factoring in multiple different applications j, the payoff function of a user i will be composed of the aggregate valuation, network effects and negative externalities:

$$\pi_i = \sum_j s_{ij} \left(v_{ij} + \left(\sum_k s_{kj:k \neq i} \right) {e_j}^+ + I_{\{\sum_k s_{kj:k \neq i} > 0\}} {e_j}^- \right), \tag{6}$$

where $s_{ij} \in \{0, 1\}$ is the decision of user i whether to install app j, v_{ij} is the valuation of app j by user i, $I \in \{0, 1\}$ is a variable indicating whether at least one friend of user i installed app j , while ${e_j}^+ > 0$ (${e_j}^- < 0$) denotes the unit positive (negative) externality inflicted by app j. Note that apps differ in nature and permissions requested, so their inflicted externalities can be very different (e.g., game app vs. news feed app). Solving such an extended game is not straightforward. We plan to utilize the measurement results to give a numerical solution in future work.

5 Discussion

Table 7 summarizes the analysis of the different scenarios of our IPG model. We discuss several insights and implications in the following.

Sub-optimal Equilibrium. Notice that the Nash equilibrium is not always socially and/or Pareto optimal. For example, in case *011*, when negative externality e^- outweighs the sum of network effect e^+ and positive user evaluation

Table 7. Nash Equilibrium (NE) as well as the Social Optimality (SO), Pareto Optimality (PO) and Vendor Optimality (VO) of different app scenarios. v denotes if initial user valuation on app is positive (+) or negative (–), while $e^+ + v$ indicates if the network effect (e^+) offsets a negative initial valuation

Case	v	$e^+ + v$	NE	SO	PO	VO
111	+	+	(i,i)	Y	Y	Y
100	–	–	(n,n)	Y	Y	N
000	–	–	(n,n)	Y	Y	N
011	+	+	(i,i)	Y/N	Y/N	Y
110 *010*	–	+	(n,n), (i,i), mixed	Y/N	Y/N	Y/N

v, the game becomes the classic Prisoner's Dilemma. In this situation, while it is socially optimal not to install the app, both users do otherwise. Similarly, inefficiency can arise in the coordination game scenarios (i.e., case *110* and *010*). The question is who will be incentivized to remedy the situation. Inefficiency can cause users to suffer from installing potentially risky or useless apps, as well as to miss out on potentially good or useful apps.

Incentive Misalignment. It is not counter-intuitive to assume that a platform vendor such as Facebook has in its best interest to stimulate app installation and data sharing. We thus define vendor optimality (VO) based on the users' decision whether to install an app in equilibrium. Comparing the SO and VO columns in Table 7, one can quickly notice the mismatched interests between platform vendor and user. This has serious implications. For example, in the Prisoner's Dilemma version of case *011*, the vendor may not have direct incentives to warn against the potentially privacy-invasive apps. A similar situation can be observed on mobile application platforms. As platform owners compete in boosting the number of apps to increase platform attractiveness, problems with inappropriate apps, coupled with a lax app review process, have not been adequately addressed [9].

Absence of Risk Signaling. Attributable to the incentive mismatch, we see that the prominent cue for users to avoid bad apps today has remained with community app ratings. Unfortunately, most of the rating systems have neither factored in the risk aspects of an app nor the negative externality e^-. In their current form, ratings are thus not helpful for privacy control, particularly in the Prisoner's Dilemma situation of case *011*. Specifically to Facebook, the platform has stopped displaying the average community rating of an app in the permission request dialog with the launch of the Enhanced Auth Dialog (see Fig. 1). Instead, Facebook displays a list of friends, if any, who have installed the app. This is an interesting move. Knowing that friends have installed a particular app can indeed help users better estimate positive network effects and negative privacy externality; this helps in the coordination cases of *110* and *010*. However, the platform does not inform users when a friend uninstalls an app. Omitting the app ratings completely may also be unhelpful. While current user ratings do not

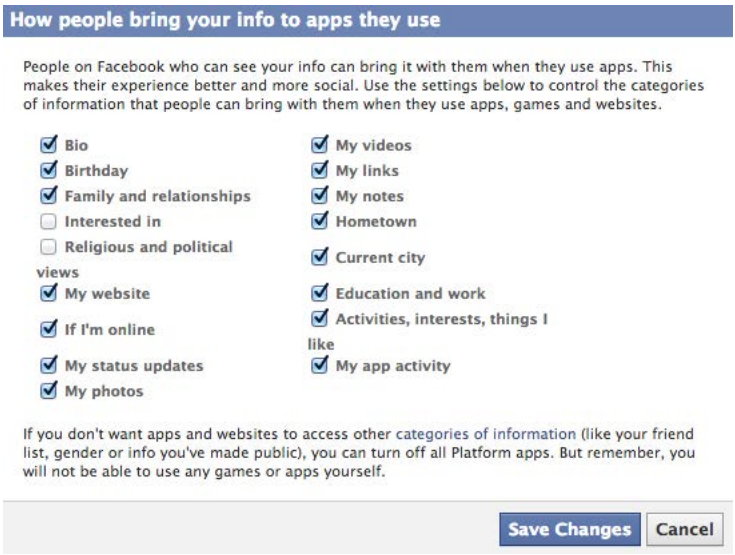


Fig. 2. Default interdependent privacy settings on Facebook

warn against privacy risks, they can be at least useful to filter off apps with low user valuation (i.e., a low v in our model).

User Awareness and Default Settings. Rather than blaming the users for not paying enough attention, we argue that a greater effort should be made to raise the user awareness on privacy implications. Privacy interdependence with regard to apps on Facebook is certainly an area urgently requiring more attention. We expect very few users to actually realize the interdependent privacy control with Facebook apps. Permissions requesting friends' personal information are not extended permissions; they are also not prominently shown on the (first screen of the) Enhanced Auth Dialog. Furthermore, the default settings in Facebook do not protect users against the negative privacy externality. As shown in Fig. 2, other than user interests, religious views and political activities, all other personal information may be "brought to others as friends use their apps". There is certainly more that Facebook can do to protect the users. By default, Facebook users also cannot review photo tagging by friends. However, when configured, users are prompted to approve or disprove individual photo tags. Such fine grain control is missing for apps; one cannot specify which apps that friends are using could gain access to their personal data currently.

6 Related Work

Since privacy is both an inherent human need and a complex technological challenge in OSNs, there has been a flurry of research in this area. Almost all authors agree on the fact that the privacy settings of OSNs are both complicated [10]

and non-uniform [11]. This hinders the users' ability to protect their online privacy [12], and gives rise to strange privacy patterns [13].

Chia et al. [5] studied the effectiveness of user-consent permission systems across three different platforms – Facebook, Chrome and Android. They found an absence of effective risk signals in the current app markets in addition to evidence of attempts to entice or trick users into compromising their privacy through free and mature apps [5]. Related to Facebook apps, King et al. [7] conducted a survey on the privacy knowledge, behavior and concerns of Facebook app users. They found that while almost all survey participants had heard of Facebook apps, only 77% of them were aware that apps are both created by Facebook and third parties. In addition, half of them were uncertain if Facebook reviews the apps [7]. A number of other works (e.g., [14–17]) have commented on the weaknesses of app permission systems in safeguarding user privacy and presented ways for improvement. In particular, Wang et al. [17] presented some insights regarding app permission dialogues, and gave an example where installing a given calendar app would violate the user's global privacy setting.

Observations that online privacy may be out of the control of the user himself have been made earlier. Researchers [18, 19] demonstrated the ability to infer private user information using only friendship links, group memberships and information shared by others publicly. Albeit similar, there is a distinctive difference between our work and theirs. Rather than focusing on unintended disclosure of private information inferred by combining pieces of public information, our work has looked into the case of explicit sharing of friends' data (through Facebook permissions).

Dealing with explicit collaborative information sharing, Hu et al. [20] proposed a method to detect and resolve privacy conflicts. Here, we focus on the interdependent nature of app privacy: we study how the Facebook permission system affects not only the user installing an app but also his friends.

Finally, [21] is closest to our work in terms of modeling interdependence. It shows the negative externality inflicted by websites using a weak password criterion to websites with strong authentication mechanisms. In order to incorporate both positive and negative externalities, our formulation of user welfare follows [22].

7 Summary and Future Work

In this paper we have taken a first step towards defining and understanding interdependent privacy. We have demonstrated the existence of privacy interdependence through a study on the Facebook application platform and its permission system. By constructing a simple Interdependent Privacy Game, we have analyzed the externalities caused by privacy interdependence and their effect on the users' and vendor's welfare in equilibrium. We have also discussed why these inefficient equilibria can emerge, and hinted on how to design a better application installation mechanism. We hope that our paper could also inspire further theoretical and experimental research on interdependent privacy.

Future Work. We have identified three potential directions of research.

Future modeling directions. Several game-theoretical extensions of IPG can be explored in the future. These include: taking into consideration the amount and sensitivity of personal data stored in the given OSN user account; incorporating unfriending by using a coordination game with a secure outside option [23]; repeated games leading to iterated PD; an evolutionary game of privacy-conscious and thrill-seeking users; and a game model based on the decisions of friends who cooperate. A very interesting improvement would be to play the multiple person, multiple app game on real social graphs.

Mechanism design based on economic theory and usability guidelines. The standard economic literature offers two ways of dealing with negative externalities: taxing of externality-producing activities and compensation of the victim by the entity inflicting the externality. While the theory is clear, it rarely makes its way into practice due to cost-minimizing behavior, hard-to-identify externality sources or the ineffectiveness of monetary compensation [24]. A possibly more effective way involves a slow cultural change which can be implemented by educating the actors of the ecosystem about externalities. Putting it into the context of OSNs, this means incentivizing the users to learn about privacy (interdependence). While this may be useful in the long run, current Facebook privacy settings may hinder its success [10].

Another alternative solution is redesigning the system mitigating negative externalities. Regarding OSNs and Facebook in particular, this means providing the user with an intuitive, yet economically inspired app install mechanism. Such a mechanism may include giving explicit control to the user (similarly to the already existing photo tagging) and/or introducing extended control over apps with negative externality potential. Explicit control for the user could be aided by an intuitive and informative interface, such as the one presented in [25]. Stronger control over apps with negative externality could be implemented, e.g., by requiring that such apps should be able to operate both without and with "friend permissions", defining two levels of operation. In addition, designing a method for acquiring meaningful user valuations for apps could help the platform vendor steer the system towards a favorable steady state. Note that all these ideas are very challenging to implement, since they have to satisfy strict usability requirements and be to the OSN providers' liking.

Other online platforms with privacy interdependence. Other online systems exhibiting privacy interdependence are abundant, e.g., various mobile application platforms (Android, iOS, Windows Phone), blogs, forums, webshops and even public cloud services. Collecting measurement data on such systems and modeling their interdependent privacy aspects are important future work.

References

1. The 3 Facebook permissions you should never agree too, `http://facecrooks.com/Internet-Safety-Privacy/the-3-facebook-app-permissions-you-should-never-agree-to.html` (last accessed: October 2012)

2. Clarke, R.: Introduction to dataveillance and information privacy, and definitions of terms (1997) (revised in 1999, 2005, 2006), http://www.rogerclarke.com/DV/Intro.html (last accessed: June 2012)
3. Mankiw, N.: Principles of Economics. Available Titles CourseMate Series, vol. 1. South-Western Cengage Learning (2008)
4. Facebook Help Center – App Basics, https://www.facebook.com/help/178140838985151/ (last accessed: October 2012)
5. Chia, P.H., Yamamoto, Y., Asokan, N.: Is this app safe? A large scale study on application permissions and risk signals. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012. ACM, New York (2012)
6. Facebook Permissions Reference, https://developers.facebook.com/docs/authentication/permissions (last accessed: June 2012)
7. King, J., Lampinen, A., Smolen, A.: Privacy: Is there an app for that? In: Proc. of the 7th Symposium on Usable Privacy and Security, SOUPS 2011, pp. 12:1–12:20. ACM (2011)
8. David, E., Jon, K.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press, New York (2010)
9. Chia, P.H., Heiner, A.P., Asokan, N.: Use of ratings from personalized communities for trustworthy application installation. In: Aura, T., Järvinen, K., Nyberg, K. (eds.) NordSec 2010. LNCS, vol. 7127, pp. 71–88. Springer, Heidelberg (2012)
10. Johnson, M., Egelman, S., Bellovin, S.M.: Facebook and privacy: it's complicated. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, SOUPS 2012, pp. 9:1–9:15. ACM, New York (2012)
11. Bonneau, J., Preibusch, S.: The privacy jungle: On the market for data protection in social networks. In: The Eighth Workshop on the Economics of Information Security, WEIS 2009 (2009)
12. Liu, Y., Gummadi, K.P., Krishnamurthy, B., Mislove, A.: Analyzing facebook privacy settings: user expectations vs. reality. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC 2011, pp. 61–70. ACM, New York (2011)
13. Dey, R., Jelveh, Z., Ross, K.: Facebook users have become much more private: A large-scale study. In: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 346–352 (March 2012)
14. Felt, A.P., Greenwood, K., Wagner, D.: The effectiveness of application permissions. In: Proc. of the 2nd USENIX Conf. on Web Application Development, WebApps 2011. USENIX Association (2011)
15. Barrera, D., van Oorschot, P.C., Somayaji, A.: A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android Categories and Subject Descriptors. In: Proc. of the 17th ACM Conf. on Computer and Communications Security, CCS 2010, pp. 73–84. ACM (2010)
16. Tam, J., Reeder, R.W., Schechter, S.: I'm Allowing What? Disclosing the authority applications demand of users as a condition of installation. Technical report, Microsoft Research, MSR-TR-2010-54 (2010)
17. Wang, N., Xu, H., Grossklags, J.: Third-party apps on facebook: privacy and the illusion of control. In: Proceedings of the 5th ACM Symposium on Computer Human Interaction for Management of Information Technology, CHIMIT 2011, pp. 4:1–4:10. ACM, New York (2011)

18. Zheleva, E., Getoor, L.: To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, pp. 531–540. ACM (2009)
19. Jernigan, C., Mistree, B.F.: Gaydar: Facebook friendships expose sexual orientation. First Monday 14(10) (October 2009)
20. Hu, H., Ahn, G.J., Jorgensen, J.: Detecting and resolving privacy conflicts for collaborative data sharing in online social networks. In: Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC 2011, pp. 103–112. ACM, New York (2011)
21. Preibusch, S., Bonneau, J.: The password game: negative externalities from weak password practices. In: Alpcan, T., Buttyán, L., Baras, J.S. (eds.) GameSec 2010. LNCS, vol. 6442, pp. 192–207. Springer, Heidelberg (2010)
22. Johari, R., Kumar, S.: Congestible services and network effects. In: Proceedings of the 11th ACM Conference on Electronic Commerce, EC 2010, pp. 93–94. ACM, New York (2010)
23. Goeree, J.K., Holt, C.A.: Ten little treasures of game theory and ten intuitive contradictions. Virginia Economics Online Papers 333, University of Virginia, Department of Economics (February 2000)
24. Center for the Advancement of Steady State Economy – Negative Externalities Are the Norm, `http://steadystate.org/negative-externalities/` (last accessed: October 2012)
25. Besmer, A., Lipford, H.R., Shehab, M., Cheek, G.: Social applications: exploring a more secure framework. In: Proceedings of the 5th Symposium on Usable Privacy and Security, SOUPS 2009, pp. 2:1–2:10. ACM, New York (2009)

A Secure Submission System for Online Whistleblowing Platforms

Volker Roth[1], Benjamin Güldenring[1], Eleanor Rieffel[2], Sven Dietrich[3], and Lars Ries[1]

[1] Freie Universität Berlin
[2] FX Palo Alto Laboratory
[3] Stevens Institute of Technology

Abstract. We motivate and introduce our design and development of a secure submission front-end for online whistleblowing platforms. Our system is designed to provide a level of anonymity in the face of adversaries who can perform end-to-end traffic analysis.

Keywords: Whistleblowing, traffic analysis, homomorphic encryption.

1 Introduction

Corporate or official corruption and malfeasance can be difficult to uncover without information provided by insiders, so-called *whistleblowers.* Even though many countries have enacted, or intend to enact, laws meant to make it safe for whistleblowers to disclose misconduct [2,12], whistleblowers fear discrimination and retaliatory action regardless, and sometimes justifiably so [4,10].

It is therefore unsurprising that whistleblowers often prefer to blow the whistle anonymously through other channels than those mandated by whistleblowing legislature. This gave rise to whistleblowing websites such as *Wikileaks.* However, the proliferation of surveillance technology and the retention of Internet protocol data records [3] has a chilling effect on potential whistleblowers. The mere act of connecting to a pertinent Website may suffice to raise suspicion [9], leading to cautionary advice for potential whistleblowers.

The current best practice for online submissions is to use an SSL connection over an anonymizing network such as Tor [7]. This hides the end points of the connection and it protects against malicious exit nodes and Internet Service Providers (ISPs) who may otherwise eavesdrop on or tamper with the connection. However, this does not protect against an adversary who can see most of the traffic in a network [5,8], such as national intelligence agencies with a global reach and view.

In this paper, we suggest a submission system for online whistleblowing platforms that we call *AdLeaks.* The objective of AdLeaks is to make whistleblower submissions unobservable even if the adversary sees the entire network traffic. A crucial aspect of the AdLeaks design is that it eliminates any signal of intent that could be interpreted as the desire to contact an online whistleblowing platform. AdLeaks is essentially an online advertising network, except that ads carry additional code that encrypts a zero probabilistically with the AdLeaks public key and

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 354–361, 2013.

sends the ciphertext back to AdLeaks. A whistleblower's browser substitutes the ciphertext with encrypted parts of a disclosure. The protocol ensures that an adversary who can eavesdrop on the network communication cannot distinguish between the transmissions of regular browsers and those of whistleblowers' browsers. Ads are digitally signed so that a whistleblower's browser can tell them apart from maliciously injected code. Since ads are ubiquitous and there is no opt-in, whistleblowers never have to navigate to a particular site to communicate with AdLeaks and they remain unobservable. Nodes in the AdLeaks network reduce the resulting traffic by means of an *aggregation* process. We designed the aggregation scheme so that a small number of trusted nodes with access to the decryption keys can recover whistleblowers' submissions with high probability from the aggregated traffic. Since neither transmissions nor the network structure of AdLeaks bear information on who a whistleblower is, the AdLeaks submission system is immune to passive adversaries who have a complete view of the network.

In what follows, we detail our threat model and our assumptions, we give an overview over the design of AdLeaks, we report on the current state of its implementation, we summarize the outcome of our scalability analysis, and we explain how AdLeaks uses cryptographic algorithms to achieve its security objectives. We give a more detailed report in [14].

2 Assumptions and Threats in Our Scope

The primary security objective of AdLeaks is to conceal the *presence* of whistleblowers, and to eliminate network traces that may make one suspect more likely than another in a search for a whistleblower. We assume that whistleblowers use AdLeaks only on private machines to which employers have no access. In fact, sending information from work computers even using work-related e-mail accounts is a mistake whistleblowers make frequently. We hope that the software distribution channels we discuss in Section 3.3 will help reminding whistleblowers to not make that mistake.

AdLeaks addresses the threat of an adversary who has a global view of the network and the capacity to store or obtain Internet protocol data records for most communications. The adversary may even require anonymity services to retain connection detail records for some time and to provide them on request. The adversary may additionally store selected Internet traffic and he may attempt to mark or modify communicated data. However, we assume that the adversary has no control over users' end hosts, and he does not block Internet traffic or seizes computer equipment without a court order. We assume that the court does not *per se* consider organizations that relay secrets between whistleblowers and journalists as criminal. The objective of the adversary is to uncover the identities of whistleblowers. The threat model we portrayed is an extension of [3] and it is likely already a reality in many modern states, or it is about to become a reality. For reasons we explain in [14] we do not consider additional threats that we would doubtless encounter, for example, in technologically advanced totalitarian countries.

3 System Architecture

AdLeaks consists of two major components. The first component is an online advertising network comparable to existing ones. The network has advertising partners (the publishers) who include links or scripts in their web pages which request ads from the AdLeaks network and display them. Publishers may receive compensation in accordance with common advertising models, for example, per mille impressions, per click or per lead generated. Advertisers run campaigns through the AdLeaks network. AdLeaks may additionally run campaigns through other ad networks to extend its reach, for example, funded by donations or profits from its own operations. The ecosystem of partners and supporters may include large newspapers, bloggers, human rights organizations and their affiliates. For example, Wikileaks has partnered with organizations such as Der Spiegel, El País and the New York Times, and OpenLeaks had hinted at support by Greenpeace and other organizations. The key ingredient of an AdLeaks ad is not its visual display but its active JavaScript content. Supporters who would forfeit significant revenue when allocating advertising space to AdLeaks ads have a choice to only embed the JavaScript portion. The JavaScript is digitally signed by AdLeaks and contains public encryption keys.

The second major component of AdLeaks is its submission infrastructure. This infrastructure consists of three tiers of servers. We refer to these tiers as *guards, aggregators* and *decryptors.* When a browser loads an AdLeaks ad, the embedded JavaScript encrypts a zero probabilistically with the embedded public key and submits the ciphertext to a guard. The guard strips unnecessary encoding and protocol meta-data from the request and forwards the ciphertext to an aggregator. An aggregator aggregates the ciphertexts it receives per second and transmits them to the decryptor. What makes this setting challenging is that we want to limit the bandwidth of the decryptor to a household Internet connection so that we can keep a close eye on the all-important machine with the decryption keys. The aggregation leverages the homomorphic properties of the Damgård-Jurik (DJ) encryption scheme [6], which means that the product of the ciphertexts is an encryption of the sum of the plaintexts. We chose the DJ scheme because it has a favorable plaintext to ciphertext ratio.

The decryptor decrypts the downloaded ciphertexts and, if it finds data in them, reassembles the data into files. The files come from whistleblowers. In order to submit a file, a whistleblower must first *obtain* an installer that is digitally signed and distributed by AdLeaks. This is already a sensitive process that signals intent. We defer the discussion of safe distribution channels for the installer to section 3.3. *Installing* the obtained software likewise signals the intent to disclose a secret, and therefore it is crucial that the whistleblower verifies the signature *before* running the installer, and assures himself that the signer is indeed AdLeaks. Otherwise, he is vulnerable to Trojan Horse software designed to implicate whistleblowers. When run, the installer produces an instrumented browser and an encryption tool. The whistleblower prepares a file for submission by running the encryption tool on it. The tool's output is a sequence of ℓ ciphertexts. Henceforth, whenever an instrumented browser runs an ad signed

by AdLeaks, it replaces the script's ciphertext with one of the ℓ ciphertexts it has not already used as a replacement.

In order to distinguish ciphertexts that are encryptions of zeros from ciphertexts that are encryptions of data we refer to the former as *white* and to the latter as *gray*. If the aggregator aggregates a set of white ciphertexts then the outcome is another white one. If exactly one gray ciphertext is aggregated with only white ones then the outcome is gray as well. If we decrypt the outcome then we either recover the data or we determine that there was no data to begin with. If two or more gray ciphertexts are aggregated then we cannot recover the original data from the decryption. We call this event a *collision* and we refer to such an outcome as a *black* one. Obviously, we must expect and cope with collisions in our system. In what follows, we elaborate on details of the design that are necessary to turn the general idea into a feasible and scalable system.

3.1 Disclosure Preparation

In order to handle collisions, the encryption tool breaks a file into blocks of a fixed equal size and encodes them with a loss tolerant *Fountain Code*. Fountain codes encode n packets into an infinite sequence of output packets of the same size such that the original packets can be recovered from any n' of them where n' is only slightly larger than n. For example, a random linear Fountain Code decodes the original packets with probability $1 - \delta$ from about $n + \log_2(1/\delta)$ output packets [11]. Let n'' be somewhat larger than n' and let $m_1, \ldots, m_{n''}$ be the Fountain encoding of the file. The tool then generates a random file identification number k and computes: $c_i = \mathrm{Enc}^{\mathrm{cca}}_{\kappa_1}(\mathrm{EncData}_{\kappa_2}(m_i, k||i||n))$ for $1 \leq i \leq n''$ where κ_1 is an aggregator key and κ_2 is the actual submission key. The purpose of the dual encryption will become clear in Section 4. We assume that the outer encryption is a fast hybrid IND-CCA secure cipher. We defer the specification of the inner encryption scheme to Section 6. It assures that, when the decryptor receives the ciphertexts, it can verify the integrity of individual chunks and of the message as a whole and he can associate the chunks that belong to the same submission with all but negligible probability (in $|k|$).

3.2 Decryption

It is substantially cheaper to multiply two DJ ciphertexts in the ciphertext group than it is to decrypt one. Furthermore, the product of ciphertexts decrypts to the sum of the plaintexts in the plaintext group. Recollect that we expect to receive a large number of white ciphertexts, that is, encryptions of zeroes. This leads to the following optimization: we form a full binary tree of fixed height, initialize its leaves with received ciphertexts $c_1, \ldots, c_n$ and initialize each inner node with the product of its children. Then, we begin to decrypt at the root. If the plaintext is zero then we are done with this tree, because all nodes in the tree are zeroes. Otherwise, the decryption yields $\gamma = \alpha + \beta \neq 0$ where α, β are the plaintexts of the left and right child, respectively. We decrypt the left child, which yields α, and calculate the plaintext of the right child as $\beta = \gamma - \alpha$ (without explicit

decryption). If α or β are zeroes then we ignore the corresponding subtree. Otherwise, we recurse into the subtrees that have non-zero roots. If a node is a leaf then we decrypt and verify it. If we find it invalid then we ignore the leaf. Otherwise we forward its plaintext to the file reassembly process. This algorithm saves us 61% decryptions or more, depending on the system load [14]. Our analysis and measurements suggest that a 12-core Mac Pro can serve 51480 concurrent whistleblowers at any time with a 18 Mb/s uplink for the decryptor, independent of the number of users whom AdLeaks serves ads.

3.3 Software Dissemination

We cannot simply offer the installer software for download because the adversary would be able to observe that. Instead, we pursue a multifaceted approach to software distribution. Our simplest and preferred approach involves the help of partners in the print media business. At the time of writing, popular print media often come with attached CDROMs or DVDs that are loaded with, for example, promotional material, games, films or video documentaries. Our installer software can be bundled with these media. Our second approach is to encode the installer into a number of segments using a Fountain Code. In this approach, AdLeaks ads randomly request a segment that the browser loads into the cache. A small bootstrapper program extracts the segments from the browser cache and decodes the installer from it when enough of them have been obtained. Since extraction happens outside the browser it cannot be observed from within the browser. The bootstrapper can be distributed in the same fashion. This reduces the distribution problem to extracting a specific small file from the cache, for example, by searching for a file with a specific signature or name in the cache directory. This task can probably be automated for most platforms with a few lines of script code. The code can be published periodically by trusted media partners in print or verbatim in webpages or it could even be printed on T-Shirts. Our third approach is to enlist partners who bundle the bootstrapper with distributions of popular software packages so that many users obtain it along with their regular software. With our multifaceted approach we hope to make our client software available to most potential whistleblowers in a completely innocuous and unobservable fashion.

4 Security Properties

Eavesdropping and Traffic Analysis AdLeaks funnels all incoming transmissions to the decryptor, and transmissions occur without any explicit user interaction. Hence, the posterior probability that anyone is a whistleblower, given his transmission is observed anywhere in the AdLeaks system, equals his prior probability. From that perspective, AdLeaks is immune against adversaries who have a complete view of the network. Furthermore, AdLeaks' deployment model is suitable to leapfrog the long-drawn-out deployment phase of anonymity systems that rely on explicit adoption. For example, if *Wikipedia* deployed an AdLeaks script then AdLeaks would reach 10% of the Internet user population overnight, based on traffic statistics by *Alexa* [1].

Outer Encryption and Dishonest Aggregators. Assume that AdLeaks did not use outer encryption. Then adversaries might employ the following *active* strategy to gain information on who is sending data to AdLeaks. The adversary samples ciphertexts of suspects from the network and aggregates the ciphertexts for each suspect. He prepares a genuine-looking disclosure that is enticing enough so that the AdLeaks editors will want to publish it with high priority. We call this disclosure the *bait.* The adversary then aggregates suspects' ciphertexts to his disclosure and submits it. If AdLeaks does *not* publish the bait within a reasonable time interval then the adversary concludes that the suspect is a whistleblower. The reasoning is as follows. If the suspect ciphertexts were zeroes then the bait is received and likely published. Since the bait was not published, the suspect ciphertexts carried data which invalidated the bait ciphertexts. This idea can be generalized to an adaptive and equally effective non-adaptive attack that identifies a single whistleblower in a group of W suspects at the expense of $\log_2 W$ baits. For this reason, AdLeaks employs an outer encryption which prevents this attack. However, if an adversary takes over an aggregator then he is again able to launch this attack. Therefore, aggregators should be checked regularly, remote attestation should be employed to make sure that aggregators boot the correct code, and keys should be rolled over regularly. Note that it may take months before a disclosure is published and that a convincing bait has a price — the adversary must leak a sufficiently attractive secret in order to make sure it is published. From this, the adversary only learns that a suspect has sent something but not what was sent.

5 Implementation

We developed fully-functional multi-threaded aggregation and decryption servers with tree decryption support as well as a Fountain Code encoder and decoder. Decryptors write recovered data to disk and the decoder recovers the original file. We also developed a fake guard server which is capable of generating and sending chunks according to a configurable ratio of white and gray ciphertexts. All servers connect to each other through SSH tunnels via port forwarding. The entire implementation consists of 101 C, header and CMake files with 7493 lines of code overall. This includes our optimized DJ implementation [6], which is based on a library by Andreas Steffen, a SHA-256 implementation by Olivier Gay, and several benchmarking tools. Our ads implement the DJ scheme based on the *JSBN.js* library and use *Web Workers* to isolate the code from the rest of the browser. The entire ad currently measures less than 81 KB. The size can be reduced further by eliminating unused library code and by compressing it. The ad submits ciphertexts via *XmlHttpRequests.* We instrumented the Firefox browser for our prototype and patched the source code in two locations. First, we hook the compilation of *Web Worker* scripts and tag every script as an AdLeaks script if it is labeled as one in lieu of carrying a valid signature. We placed a second hook where Firefox implements the *XmlHttpRequest.* Whenever the calling script is an AdLeaks script running within a *Web Worker,* we replace the zero chunk in its request with a data chunk. Since Web Workers run concurrently the cryptographic operations do not negatively affect the browsing experience.

6 Ciphertext Aggregation Scheme

Our ciphertext aggregation scheme is based on the Damgård-Jurik (DJ) scheme, which IND-CPA secure and is also an isomorphism of

$$\psi_s : \mathbb{Z}_{N^s} \times \mathbb{Z}_N^* \leftrightarrow \mathbb{Z}_{N^{s+1}}^* \qquad \psi_s(a; b) \mapsto (1+N)^a \cdot b^{N^s} \bmod N^{s+1}$$

where N is a suitable public key. The parameter s controls the ratio of plaintext size and ciphertext size. We use two two DJ encryptions c, t to which we jointly refer as a *ciphertext.* We refer to t separately as the *tag.* The motivation for this arrangement is improved performance. We wish to encrypt long plaintexts and the costs of cryptographic operations increase quickly for growing s. Therefore we split the ciphertext into two components. We use a shorter component with $s = 1$, which allows us to test quickly whether the ciphertext encrypts data or a zero. The actual data is encrypted with a longer component with $s > 1$. The two components are glued together using Pederson's commitment scheme [13], which is computationally binding and perfectly hiding. This requires two additions to the public key, which are a generator g of the quadratic residues of $\mathbb{Z}_N^*$ and some $h = g^x$ for a secret x. Instead of committing to a plaintext the sender commits to the hash of the plaintext and some randomness. We use a collision resistant hash function H for this purpose, which outputs bit strings of length $|N/16|$. Furthermore, let R be a source of random bits. The details of the data encryption and decryption algorithms are as follows:

EncData(m, r_0) =
  $r_1, r_2 \leftarrow R$
  chk $\leftarrow$ if $m, r_0 = 0$ then 0
    else $H(m, r_0)$
  $c \leftarrow \psi(m; h^{\text{chk}} \cdot g^{r_1}))$
  $t \leftarrow \psi(r_0||r_1; g^{r_2})$
  return c, t

DecVrfy(c, t) =
  $(m; k), (r_0||r_1; \cdot) \leftarrow \psi^{-1}(c), \psi^{-1}(t)$
  chk $\leftarrow$ if $m, r_0 = 0$ then 0
    else $H(m, r_0)$
  if $h^{\text{chk}} \cdot g^{r_1} = k$ then
    return m, r_0
  return $\perp$

We assume that $|r_0|, |r_1|, |r_2|$ are polynomial in the security parameter. Here, r_0 corresponds to $k||i||n$ as we introduced it in Section 3.1. We define EncZero = EncData$(0, 0)$. Aggregation is simply the multiplication of the respective ciphertext components. In order to avoid fields overflowing into adjacent ones we assume that r_0, r_1, r_0', r_1' are left-padded with zeroes. The amount of padding determines how many ciphertexts we can aggregate in this fashion before an additive field overflows into an adjacent one and corrupts the ciphertext. If we use B bits of padding then we can safely aggregate up to 2^B ciphertexts. A length of $B = 40$ is enough for our purposes.

7 Conclusions

AdLeaks leverages the ubiquity of online advertising to provide anonymity and unobservability to whistleblowers making a disclosure online. The system introduces a large amount of cover traffic in which to hide whistleblower submissions,

and aggregation protocols that enable the system to manage the huge amount of traffic involved, enabling a small number of trusted nodes with access to the decryption keys to recover whistleblowers' submissions with high probability. We analyzed the performance characteristics of our system extensively, please refer to [14] for details. Our research prototype demonstrates the feasibility of such a system. We expect many aspects of the system can be improved and optimized, providing ample opportunity for further research.

Acknowledgements. The first, second and last author are supported by an endowment of *Bundesdruckerei GmbH.*

References

1. Alexa (April 2012), `http://www.alexa.com`
2. Banisar, D.: Whistleblowing — International Standards and Developments. Transparency International (February 2009)
3. Berthold, S., Böhme, R., Köpsell, S.: Data retention and anonymity services. In: Matyáš, V., Fischer-Hübner, S., Cvrček, D., Švenda, P. (eds.) The Future of Identity. IFIP AICT, vol. 298, pp. 92–106. Springer, Heidelberg (2009)
4. E. R. Center: 2011 National Business Ethics Survey, 2345 Crystal Drive, Suite 201, Arlington, VA 22202, USA (2012), `http://www.ethics.org/nbes`
5. 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)
6. Damgård, I., Jurik, M., Nielsen, J.: A generalization of Paillier's public-key system with applications to electronic voting. International Journal of Information Security 9, 371–385 (2010)
7. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proc. USENIX Security Symposium, pp. 303–320 (2004)
8. Dyer, K.P., Coull, S.E., Ristenpart, T., Shrimpton, T.: Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In: IEEE Symposium on Security and Privacy, pp. 332–346 (2012)
9. Gustin, S.: Columbia university reverses anti-WikiLeaks guidance (December 2010), `http://www.wired.com/threatlevel/2010/12/columbia-wikileaks-policy/`
10. Lennane, K.J.: "Whisteblowing" a health issue. British Medical Journal 307, 667–670 (1993)
11. MacKay, D.: Fountain codes. IEE Proceedings 152(6) (December 2005)
12. Osterhaus, A., Fagan, C.: Alternative to Silence — Whistleblower Protection in 10 European Countries. Transparency International (2009)
13. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
14. Roth, V., Güldenring, B., Rieffel, E., Dietrich, S., Ries, L.: A secure submission system for online whistleblowing platforms (January 2013), `http://arxiv.org/abs/1301.6263`

Securing Anonymous Communication Channels under the Selective DoS Attack

Anupam Das and Nikita Borisov

University of Illinois at Urbana Champaign, USA
{das17,nikita}@illinois.edu

Abstract. Anonymous communication systems are subject to selective denial-of-service (DoS) attacks. Selective DoS attacks lower anonymity as they force paths to be rebuilt multiple times to ensure delivery, which increases the opportunity for more attack. We present a detection algorithm that filters out compromised communication channels for one of the most widely used anonymity networks, Tor. Our detection algorithm uses two levels of probing to filter out potentially compromised tunnels. We probabilistically analyze our detection algorithm and show its robustness against selective DoS attacks through simulation. We also analyze the overhead of our algorithm and show that we can achieve better security guarantee than the conventional Tor path selection algorithm, while adding only approximately 5% bandwidth overhead to the Tor network. Finally, we validate our design with experiments using the live Tor network.

Keywords: Anonymity, Tor network, denial of service (DoS) attack.

1 Introduction

Anonymous communication was first introduced in 1981 with Chaum's seminal paper on "Untraceable electronic mail, return addresses, and digital pseudonyms" [14]. Since then, many researchers have concentrated on building, analyzing and attacking anonymous communication systems such as Tor [18], I2P [4], Freenet [3]. In this paper we concentrate on Tor [18], one of the most widely used low-latency anonymity networks, which conceals users' identities and activities from surveillance and traffic analysis. Tor provides confidentiality and privacy to users of various types ranging from ordinary individuals to business personnel, journalists, government employees and even military personnel [8]. Currently, Tor has over 3000 relays all around the world and it is used by hundreds of thousands of people every day [7, 19, 21].

Users' identities, however, can become exposed when multiple relays are compromised. By default, Tor uses three relays and an attacker who can gain control of the entry and exit relays is capable of compromising user identity using timing analysis [20, 25]. Moreover, malicious nodes can perform a selective denial-of-service (DoS) attack [12, 13] where malicious relays drop circuits if they cannot compromise them. This increases the probability of such a path being built and as a result lowers anonymity. The selective DoS attack is particularly useful for an attacker with moderate resources; as one potential example, the Dutch ministry of Justice and Security proposed passing a law which would enable the law enforcement office to launch any form of attack

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 362–370, 2013.

(i.e., selective DoS could one form of attack) on any system in order to gather evidence [1]. So some form of mechanism is needed to ensure secure path construction in the presence of compromised/controlled relays.

Danner et al. [15] showed that it is possible to identify relays mounting selective DoS using exhaustive probing. The intent is to periodically carry out these probes and blacklist the misbehaving relays; however, the total number of probes required is prohibitive– 3 times the size of the network at a minimum, and many more (typically retrying each probe 10 times) to account for non-malicious failures. So, their approach seems practical for a centralized design, but we wanted to create a local mechanism to defend against selective DoS. By using probabilistic inference, we can make do with orders of magnitude fewer probes and thus our approach is practical to be executed at each individual client. We perform simulations and real world experiments to show the effectiveness of our detection mechanism.

2 Background

2.1 Tor Network

Tor [18] is an anonymous communication network that allows users to make TCP connections to Internet sites without revealing their identity to the destination or third-party observers. We briefly explain the main components of Tor relevant to this work. To initiate an anonymous TCP connection, a Tor user constructs a *circuit* (also known as a tunnel or path) comprised of several Tor *relays*. The relays form a forwarding chain that sends traffic from the user to the destination, and vice versa. Circuits typically involve three relays: the *entry*, *middle*, and *exit*. The traffic contents are protected by a layered encryption scheme [24], where each relay peels off a layer while forwarding. As a result, any individual router cannot reconstruct the whole circuit path and link the source to the destination. The relays in a circuit are chosen following specific constraints [17]. Each user selects a small, fixed number (currently 3) of entry relays that are used for all circuits. These relays are called *guard relays* [23, 28]; their use is designed to defend from the predecessor attack [29]. To choose the exit relay, the user picks from among those relays that have an exit policy compatible with the desired destination. After these constraints, the relays for each position are chosen randomly, weighted by their bandwidth[1].

Tor aims to provide low-latency traffic forwarding for its users. As a result, as traffic is forwarded along a circuit, timing patterns remain discernible, and an attacker who observes two different relays can use timing analysis to determine whether they are participating in the same circuit [20, 25, 27, 30]. So, to link a Tor user to a destination, it suffices to observe the entry and the exit relays of a circuit. Standard security analysis of Tor [18, 27] shows that if t is the fraction of relays that are observed, an adversary will be able to violate anonymity on t^2 of all of the circuits. Note that, due to bandwidth-weighted path selection in Tor, t is best thought of as the fraction of total Tor *bandwidth*

[1] Guard and/or exit relays are underweighted when chosen as middle node to improve the overall balancing of load, however, these details are not key to our discussion.

that belongs to relays under observation[2]. The security of Tor, therefore, relies on the assumption that a typical adversary will not be able to observe a significant fraction of Tor relays. The easiest way to observe relay traffic is to run your own relays as there is no barrier to entry other than Internet connectivity and sufficient bandwidth.

2.2 Selective Denial of Service in Tor

An adversary who controls a Tor relay can perform a number of active attacks to increase the odds of compromise [12, 13]. One approach, which is the focus of this work, is *selective denial of service* [13]. A compromised relay that participates in a circuit can easily check whether both the entry and exit relays are under observation. If this is not the case, the relay can "break" the circuit by refusing to forward traffic. This will cause the user to reformulate a new circuit for the connection, giving the adversary another chance to compromise the circuit. A simple analysis shows that this increases the overall fraction of compromised circuits to $\frac{t^2}{t^2+(1-t)^3} > t^2$, because only circuits with compromised entry and exit relays (t^2) or circuits with no compromised relays ($(1-t)^3$) will be functional, and out of those t^2 will be compromised. E.g., if $t = 0.2$, selective DoS increases the fraction of compromised circuits by 81.25%. The use of guard nodes changes the analysis somewhat; compromised guards can amplify the effect of selective DoS. Bauer et al. [11] showed that deploying a moderate number of inexpensive[3] middle-only relays can boost the effect of selective DoS attack.

2.3 Threat Model

In our threat model we assume that a small fraction (typically 20%) of the Tor relays are compromised and for each user g (where $g \in \{0, 1/3, 2/3, 1\}$) fraction of the guard nodes are compromised. Compromised relays carry out selective DoS attack, however they may choose to perform probabilistic dropping where a compromised relay terminates a certain fraction of all circuits that it cannot compromise. Finally, we assume that probes are indistinguishable from real user traffic[4].

3 Detection Algorithm

Our algorithm is built on the fundamental assumption of the Tor security model that a relatively small fraction of all relays are compromised. The algorithm works in two phases and runs periodically. Table 1 summarizes the different parameters used for our detection algorithm.

[2] To be more precise, the correct fraction would be $t_g \cdot t_e$, where t_g and t_e are the fractions of the guard and exit bandwidth under observation, respectively. For simplicity of presentation, we will assume $t_g = t_e = t_m = t$ in the rest of the paper.

[3] Middle-only nodes do not have to fulfill stronger commitments (e.g., minimum bandwidth, minimum uptime, legal issues related to exit policies) that guard and exit nodes have to fulfill.

[4] To mask probes from actual user traffic we propose downloading popular web pages listed by Alexa [5]. More discussion is available in our technical report [16].

Table 1. Parameters Used

Setting	Parameter	Description
Environmental	t	Fraction of relays compromised
	g	Fraction of compromised guards per user
	d	Random drop rate by compromised nodes
Tunable	N	# of working Tor circuits created in 1st phase
	K	# of probes used per circuit in 2nd phase
	θ	Threshold for classifying circuit

3.1 First Phase

Under active use, Tor will switch to a new circuit every 10 minutes, meaning that we need 6 non-compromised circuits every hour. So in the first phase of our detection algorithm we iteratively generate a random Tor circuit and test its functionality by retrieving a random web file through the circuit. If it fails we discard the circuit and try a new circuit. We stop when we have N (we can calculate the value of N using equation: $N = \left\lceil 6 \times \frac{gt+(1-g)(1-t)^2}{(1-g)(1-t)^2} \right\rceil$. Considering worst case scenario we can set the value of N to 10) working circuits. If an adversary is carrying out selective DoS attack then after the first phase we should have a set of circuits of form either CXC or HHH, where C denotes a compromised relay, H denotes an honest one, and X is a relay of any type.

3.2 Second Phase

In the second phase, we examine each of the circuits passing the first phase (we will call these circuits as *potential* circuits) as follows:

- We randomly pick $K(1 \leq K < N)$ other circuits (we will call them as *candidate* circuits) out of the list of potential circuits.
- For each of the K *candidate* circuits, we change the exit relay of the *potential* circuit being evaluated with the exit relay of the candidate circuit and choose a random middle relay from the candidate set. We then test the functionality of the new circuit by performing a web retrieval through it. If, out of these K probes, θ or more succeed, we consider the evaluated circuit to be honest; otherwise, we consider it to be compromised.

Note that under selective DoS, if we change the exit relay of a compromised circuit with that of an honest circuit, we will get a circuit where the entry is compromised while the exit is honest and hence the file retrieval should fail. On the other hand, if both the evaluated and candidate circuits are honest or compromised, the probe will succeed. We expect more success for an honest circuit, since most of the potential circuits are honest; we use θ as a threshold for distinguishing between the two circuit types. At the end of the second phase, we will have some number of potentially honest circuits. This collection of circuits is then used for making real anonymous connections.

4 Security Analysis

4.1 False Error Rates

We first evaluate the false-negative (FN) and false-positive (FP) errors of our algorithm under selective DoS strategy. False-negative (FN) error, i.e., the fraction of compromised circuits that pass our detection algorithm, depends upon the number of compromised ($n(\mathrm{CXC})$) and honest ($n(\mathrm{HHH})$) circuits that pass the first phase (we can compute such probabilities using Binomial distributions). Now, a false-negative error occurs when a compromised circuit is paired with at least θ other compromised candidate circuits in the second phase. So we can use hypergeometric distribution to calculate $\Pr(\mathrm{FN})$ (similarly we can compute $\Pr(\mathrm{FP})$). Detailed derivation of these false errors can be found in our technical report [16]. Transient network failure can directly influence the success rate of our probing, so it can affect both FN and FP errors. We discuss the impact of such failures in our technical report [16] as well.

4.2 Tuning Parameters

Security vs Overhead: Our detection algorithm has two tunable parameters (K, θ) (see Table 1 for description). For tuning purpose, we introduce two evaluation metrics: *security* (ψ) and *overhead* (η). We then tune K and θ in terms of these evaluation metrics. We define *security* as the probability of not choosing a compromised circuit for actual usage and *overhead* as the expected number of probes required for each usable circuit (by usable circuits we refer to circuits that are used for actual client traffic). We define ψ and η using the following functions (both of these metrics are approximations):

$$\psi = 1 - \frac{gt \times \Pr(\mathrm{FN})}{gt \times \Pr(\mathrm{FN}) + (1-g)(1-t)^2 \times (1 - \Pr(\mathrm{FP}))} \tag{1}$$

$$\eta = \frac{1 + \left[gt + (1-g)(1-t)^2\right] \times K}{gt \times \Pr(\mathrm{FN}) + (1-g)(1-t)^2 \times (1 - \Pr(\mathrm{FP}))} \tag{2}$$

Detailed derivations of these metrics can be found in our technical report [16]. We can then look at the distribution of ψ vs η for different values of (K, θ) and choose a (K, θ) pair that achieves satisfactory security guarantee at the cost of reasonable overhead.

5 Experimental Evaluation

5.1 Simulation Results

We implemented a simulator in C++ that emulates the basic functionality of Tor circuit construction and selective DoS attacks. We collected real Tor node information from the Tor network status page [10] and randomly tagged 20% ($t = 0.2$) of the bandwidth to be controlled by a compromised entity. To analyze the robustness and effectiveness of our detection algorithm we vary g ($0 \leq g \leq 1$) and d ($0 \leq d \leq 1$) in our simulations. Here, 100% drop rate refers to selective DoS and 0% drop means no dropping at all. Based on empirical results from our technical report [16], we set $K = 3$ and $\theta = 2$ in all the simulations. All simulation results are averaged over 100 runs with 95% confidence interval.

Robustness: First, we will look at the robustness of our detection algorithm in filtering out compromised circuits. For this purpose we evaluate the probability of selecting compromised circuits, $\Pr(\text{CXC})$[5], against different drop rates, d.

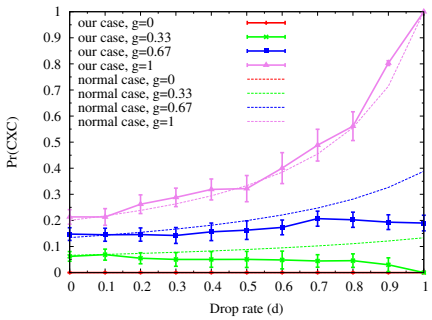


Fig. 1. Probability of selecting compromised circuits (CXC) for different drop rates d. In general $\Pr(\text{CXC})$ decreases as d rises compared to the conventional Tor network.

From Figure 1, we see that as drop rate d increases, the probability of selecting compromised circuits for our approach lowers compared to the conventional Tor circuit construction policy (indicated by the dashed lines). The main reason behind the decrease of $\Pr(\text{CXC})$ lies on the fact that as compromised nodes start to perform aggressive dropping, the pool of available circuits after the first phase quickly converges to the set $\{\text{CXC}, \text{HHH}\}$. This in turn lowers a compromised circuit's chance of selecting other compromised circuits as candidates in the second phase because honest circuits dominate over compromised circuits for $t = 0.2$.

5.2 Real-World Experiment

We carried out real-world experiments by introducing our own relays into the Tor network, all of which acted as compromised nodes. For this purpose we used 11 Emulab [2] machines, 10 of which were configured to act as Tor relays with a minimum bandwidth capacity of 20Kbps. Note that all our nodes belonged to the same /16 subnet, meaning that no user would (by default) choose two of our nodes in the same circuit. Moreover, to prevent other users from using our nodes as exit nodes, we configured our relays with a fixed exit policy (allowing connection to only specific destinations). All these measures were taken to respect the privacy of Tor users. To implement selective DoS, we take an approach similar to the one described by Bauer et al. [12]. We modified Tor source code version-*tor-0.2.2.35* and implemented our detection algorithm in the client side in Python (we used the Python version of Tor-Controller [6]).

Robustness: We first query the Tor directory server to retrieve a list of all available Tor relays and then consider only those routers which are flagged as *running*, *stable* and *valid*, because we want our routers to be alive and running during our experiments. We selected 40 Tor nodes (3 guards, 19 exits and 18 relays) at random with probability proportional to their bandwidth and added our own 10 nodes to this set to get a total of 50 nodes. Then we ran our experiments on this small set of Tor nodes where nodes were selected randomly. This choice results in about 20% of the nodes being compromised. To emulate user traffic, we retrieve random web files 100–300 KB in size. We set $K = 3, \theta = 2$ for our experiments. Table 2 summarizes our findings.

[5] $\Pr(\text{CXC}) = n(\text{CXC}) / [n(\text{HHH}) + n(\text{CXC}) + (1-d)n(\text{Others})]$.

Overhead: Let us now estimate what kind of bandwidth overhead our mechanism would inflict on the real Tor network. Now on average a single usable circuit requires approximately 4 probes (one in the 1st phase and 3 in the 2nd phase). And since we are proposing to use popular web sites as probing destinations we can approximate the average probe size to be 300KB [26]. So the total traffic used by a single user every one hour is $(6 \times 3 \times 300 \times 4)KB\approx$ 21MB. Now, Tor's bandwidth capacity was found to be 3.21GB/s [7] during the month of September 2012. If we allow 5% of the bandwidth to be used for our detection algorithm then we can support approximately 28,000 simultaneous users per hour (i.e.,$\approx 672,000$ user attempts daily which is comparable to $619,696$, the peak number of daily Tor users for October 2012 [7]).

Table 2. Results from the Tor Network

g	FN	FP	ψ	Security in Current Tor
0	0.0	0.0664	1.0	1.0
1/3	0.0	0.178	1.0	0.867
2/3	0.133	0.283	0.843	0.612
1	1.0	0.0	0.0	0.0

From Table 2 we see that as g increases the security assurance provided by both our approach and the conventional Tor network goes down. However, for $g = 1/3, 2/3$ our approach shows significant improvement in filtering out compromised circuits.

6 Related Work

Borisov et al. [13] first showed that carrying out selective DoS could benefit an adversary to increase its chance of compromising anonymity for both high and low-latency anonymous communication systems. In fact, it was pointed out that with 20% compromised nodes in Salsa [22], the selective DoS attack results in 19.14% compromised tunnels compared to the conventional security analysis of 6.82% compromised tunnels.

Later on Danner et al. [15] proposed a detection algorithm for selective DoS attack on Tor. Their algorithm basically probes each individual Tor node in the network and they prove that this requires $O(n)$ probes to detect all compromised nodes in a Tor network comprising of n participants. For circuits of length 3, their algorithm requires $3n$ probes; however to handle transient network failures they propose to repeat each probe 10 times. Clearly, their approach seems only practical for a centralized design, but ours is a local mechanism to defend against selective DoS.

Recently, Mike Perry proposed a client-side accounting mechanism that tracks the circuit failure rate of each guard node used by a client [9]. The goal is to avoid malicious guards that deliberately fail circuits extending to non-colluding exit nodes. We take a more proactive approach to finding malicious circuits through probing instead of tracking actual circuit usage.

7 Conclusion

Anonymous communication systems like Tor are vulnerable to selective DoS attacks that considerably lower anonymity. Such attacks however, can be detected through

probing. Our detection algorithm probes communication channels to filter out potentially compromised ones with high probability. We also show that adaptive adversaries who choose to deny service probabilistically do not benefit from adopting such a strategy. Our experimental results demonstrate that our detection algorithm can effectively defend users against selective DoS attack.

References

[1] Dutch government proposes cyberattacks against... everyone, https://www.eff.org/deeplinks/2012/10/dutch-government-proposes-cyberattacks-against-everyone
[2] Emulab, https://www.emulab.net
[3] Freenet, https://freenetproject.org/
[4] I2P, http://www.i2p2.de/
[5] Top sites on the web, http://www.alexa.com/topsites
[6] Tor controller, https://svn.torproject.org/svn/blossom/trunk/TorCtl.py
[7] Tor metrics portal, https://metrics.torproject.org/
[8] Tor project, https://www.torproject.org/
[9] Tor proposal-209, https://gitweb.torproject.org/user/mikeperry/torspec.git/blob/path-bias-tuning:/proposals/209-path-bias-tuning.txt
[10] Torstatus, http://torstatus.blutmagie.de/index.php
[11] Bauer, K., Juen, J., Borisov, N., Grunwald, D., Sicker, D., Mccoy, D.: On the optimal path length for Tor. In: 3rd Workshop on Hot Topics in Privacy Enhancing Technologies (2010)
[12] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against Tor. In: ACM Workshop on Privacy in Electronic Society, pp. 11–20 (2007)
[13] Borisov, N., Danezis, G., Mittal, P., Tabriz, P.: Denial of service or denial of security? In: 14th ACM Conference on Computer and Communications Security, pp. 92–102 (2007)
[14] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24, 84–90 (1981)
[15] Danner, N., Krizanc, D., Liberatore, M.: Detecting denial of service attacks in Tor. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 273–284. Springer, Heidelberg (2009)
[16] Das, A., Borisov, N.: Securing Tor tunnels under the selective-DoS attack, http://arxiv.org/abs/1107.3863
[17] Dingledine, R., Mathewson, N.: Tor path specification, https://gitweb.torproject.org/torspec.git/blob/HEAD:/path-spec.txt
[18] Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: 13th USENIX Security Symposium, pp. 303–320 (2004)
[19] Hahn, S., Loesing, K.: Privacy-preserving ways to estimate the number of Tor users (2010), https://metrics.torproject.org/papers/countingusers-2010-11-30.pdf
[20] Levine, B.N., Reiter, M.K., Wang, C.-X., Wright, M.: Timing attacks in low-latency mix systems. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 251–265. Springer, Heidelberg (2004)
[21] Loesing, K.: Measuring the Tor network: Evaluation of client requests to the directories. Tech. rep. (2009), https://metrics.torproject.org/papers/directory-requests-2009-06-25.pdf

[22] Nambiar, A., Wright, M.: Salsa: a structured approach to large-scale anonymity. In: 13th ACM Conference on Computer and Communications Security, pp. 17–26 (2006)
[23] Overlier, L., Syverson, P.: Locating hidden servers. In: IEEE Symposium on Security and Privacy, pp. 100–114 (2006)
[24] Reed, M., Syverson, P., Goldschlag, D.: Anonymous connections and onion routing. IEEE Journal on Selected Areas in Communications 16, 482–494 (1998)
[25] Shmatikov, V., Wang, M.-H.: Timing analysis in low-latency mix networks: Attacks and defenses. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 18–33. Springer, Heidelberg (2006)
[26] Sreeram Ramachandran, G.: Web metrics: Size and number of resources, `https://developers.google.com/speed/articles/web-metrics`
[27] Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an analysis of onion routing security. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 96–114. Springer, Heidelberg (2001)
[28] Wright, M., Adler, M., Levine, B.N., Shields, C.: Defending anonymous communications against passive logging attacks. In: IEEE Symposium on Security and Privacy, pp. 28–41 (2003)
[29] Wright, M.K., Adler, M., Levine, B.N., Shields, C.: An analysis of the degradation of anonymous protocols. In: Network and Distributed System Security Symposium (2002)
[30] Zhu, Y., Fu, X., Graham, B., Bettati, R., Zhao, W.: On flow correlation attacks and countermeasures in mix networks. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 207–225. Springer, Heidelberg (2005)

PIRMAP: Efficient Private Information Retrieval for MapReduce

Travis Mayberry, Erik-Oliver Blass, and Agnes Hui Chan

College of Computer and Information Science
Northeastern University, Boston MA-02115, USA
{travism,blass,ahchan}@ccs.neu.edu

Abstract. Private Information Retrieval (PIR) allows a user to retrieve bits from a database while hiding the user's access pattern. However, the practicality of PIR in a real-world cloud computing setting has recently been questioned. In such a setting, PIR's enormous computation and communication overhead is expected to outweigh the cost saving advantages of cloud computing. In this paper, we first examine existing PIR protocols, analyzing their efficiency and practicality in realistic cloud settings. We identify shortcomings and, subsequently, present an efficient protocol (PIRMAP) that is particularly suited to MapReduce, a widely used cloud computing paradigm. PIRMAP focuses especially on the retrieval of large files from the cloud, where it achieves good communication complexity with query times significantly faster than previous schemes. To achieve this, PIRMAP enhance related work to allow for optimal parallel computation during the "Map" phase of MapReduce, and homomorphic aggregation in the "Reduce" phase. To improve computational cost, we also employ a new, faster "somewhat homomorphic" encryption, making our scheme practical for databases of useful size while still keeping communication costs low. PIRMAP has been implemented and tested in Amazon's public cloud with database sizes of up to 1 TByte. Our evaluation shows that non-trivial PIR such as PIRMAP can be more than one order of magnitude cheaper and faster than trivial PIR in the real-world.

Keywords: Privacy, MapReduce, cloud computing, Private Information Retrieval.

1 Introduction

Cloud computing has recently been identified as an important business technology [10], as it offers greater flexibility and reduced costs to companies outsourcing data and computation. The cost advantage of cloud computing comes from the fact that cloud users do not need to maintain their own, expensive data center, but instead can pay a cloud provider for hosting. However, despite the hype, hesitations remain among potential cloud users. Lack of security and privacy guarantees has been identified as a major adoption obstacle for both large enterprise [19] and governmental organizations [14].

Privacy issues stem from the fact that, when a user hands over his information to the cloud, control of that data is relinquished to the cloud. Public clouds are threatened by hackers due to the much larger target they present. Insiders such as data center administrative staff can easily access private data that has been outsourced. Such attacks are

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 371–385, 2013.

realistic and have already been reported [11, 21]. This shows that users need to take precautions to ensure the privacy and integrity of their cloud data.

While encryption of data at rest helps to protect data confidentiality, it is often not sufficient: the subsequent data access patterns can leak information about patients. For example, if the outsourced data in question contains encrypted patient records, the cloud provider might learn that a patient has been diagnosed with cancer when it sees that the patient's records have been retrieved by an oncologist. In general, it is hard to assess which information may be leaked, what inferences an adversary may make, and what risks the leakage may incur.

Private Information Retrieval (PIR) offers a solution to the information leakage problem by hiding access patterns [6, 12], independent of an encryption mechanism. In PIR, a database stored at a server holds n strings each of size l bits, and a user can query for one l bit string without leaking *which* string to the database. A trivial way to accomplish this is to simply transmit the entire database to the user. However, this is *communication* inefficient, and research focuses on achieving lower communication bounds. In contrast, *computational* cost of every PIR scheme is necessarily $O(n \cdot l)$, as the server must "touch" every piece of the database if the server is to remain oblivious of the requested piece.

In existing work, the server's computation is comprised of expensive cryptographic operations over the entire database. Because of the significant overhead this imposes, it has recently been questioned whether PIR will ever become *practical* in a real-world cloud computing setting, cf. Sion and Carbunar [18]. Typically, cloud providers such as Amazon charge their customers for both data transfer *and* CPU hours [2]. Due to the necessary condition that PIR protocols compute over the entire database for each query, it has been argued that trivial PIR (retrieving the whole $(l \cdot n)$ bit database) is not only faster, but also *cheaper* for the cloud customer compared to a PIR query that involves lengthy computation [5, 16, 18].

Another open question is how to perform PIR in a real-world cloud computing environment. In contrast to a single machine server storing the data, one of the biggest challenges in cloud computing is a design that scales easily to the large distributed systems which are characteristic in cloud settings. In order to alleviate this difficulty, major cloud providers (e.g., Amazon, Google, IBM, Microsoft) offer an interface to the prominent MapReduce [8] API for distributed computing to their users. MapReduce comprises not only parallelization ("Map") of work, but an aggregation ("Reduce") of individual results to keep computational burden on the user side low.

This paper considers the single-server, computationally-private information retrieval setting. This is appropriate, because, although a cloud provider may allow access to many servers, they must all be considered a single "trust entity": they are under the control of the same organization. Trust-wise, the cloud as a single, large server with many distributed CPUs. In the single-server setting, it is known that unconditionally secure PIR cannot be more efficient than transferring the entire database [7], so we are concerned instead with a computationally secure protocol (cPIR). Further use of the term PIR will be in reference to computationally-secure PIR, unless otherwise noted.

We present PIRMAP, an efficient single-server cPIR protocol suited for MapReduce clouds. PIRMAP especially targets retrieval of relatively large files, a more specific

setting than considered by previous work. In a scenario with n files each of size l bits and $l \gg n$, PIRMAP achieves communication complexity linear in l with low constants. PIRMAP is designed for and leverages MapReduce parallelization and aggregation. We have implemented PIRMAP in Hadoop MapReduce, and its performance will be presented in Section 5.

Our contributions are:

1.) An analysis of existing work with respect to its practicality in a MapReduce cloud setting.

2.) PIRMAP, an efficient, practical PIR scheme for MapReduce with optimal communication complexity $O(l)$ when retrieving an l bit file for $l \gg n$. Additionally, PIRMAP has computational complexity as good or better than prominent related work due to use of a more efficient homomorphic encryption. PIRMAP runs on top of standard MapReduce, not requiring changes to the underlying cloud infrastructure.

3.) An implementation of PIRMAP that is usable in real-world MapReduce clouds today, e.g., Amazon. We evaluate PIRMAP, first, in our own (tiny) local cloud and, second, with Amazon's cloud. We verify its practicality by scaling up to 1 TByte of data in Amazon's cloud setting. Compared to the previously largest single database PIR experiment with up to 28 GByte of data [16], this demonstrates the efficiency and practicality of PIRMAP in the real-world. PIRMAP is more than one order of magnitude *cheaper* and *faster* than trivial PIR, and – in our particular scenario – we can significantly outperform related work. PIRMAP's source code is available for download [17].

2 Problem Statement

PIR: A PIR protocol is a series of interactions between a user and a server storing n files that results in the user retrieving one file while the server does not learn which file. More formally, the server cannot guess with probability greater than $1/n$ which file was queried. Note that this probability does not increase over multiple queries. This effectively hides the user's access pattern as each query is computationally indistinguishable from the others. The only information leaked is the number of files queried.

A $CPIR_l^n$ protocol $(\mathsf{Query}, \mathsf{Transfer}, \mathsf{Recover})$ is a computationally-secure private information retrieval protocol over a database of n elements, each of bit length l. The Query function generates the query and sends it to the server. Transfer is run on the server and involves transforming the received query into the results of the query and sending it back to the client. Finally, Recover is run by the client to transform the response from the server into the correct plaintext query result.

MapReduce: With the trend towards more computers and more cores rather than faster individual processors, it is important that any practical PIR implementation be deployable in a way that can take full advantage of parallel and cluster computing environments. Perhaps the most widely adopted architecture for scaling parallel computation in public clouds today is Google's MapReduce [8]. Its design allows for a set of computations, or "job", to be deployed across many nodes in a cloud cloud. Its biggest advantage is that it scales transparently to the programmer. That is, once an implementation is written using MapReduce, it can run on any number of nodes in the data center,

from one up to hundreds or thousands, without changes in the code. This is managed by splitting computation into two phases, each of which can be run in parallel on many computing nodes.

The first phase is called the "Map" phase. MapReduce will automatically split the input to the computation equally among available nodes in the cloud data center, and each node will then run a function called *map* on their respective pieces (called *InputSplits*). It is important to note that the splitting actually occurs when the data is uploaded into the cloud (in our case when the patient record/files are uploaded) and not when the job is run. This means that each "mapper" node will have local access to its InputSplit as soon as computation has started and you avoid a lengthy copying and distributing period. The map function runs a user defined computation on each InputSplit and outputs a number of *key-value* pairs that go into the next phase.

The second phase, "Reduce", takes as input all of the key-value pairs emitted by the mappers and sends them "reducer" nodes in the data center. Specifically, each reducer node receives a single key, along with the sequence of values output by the mappers which share that key. The reducers then take each set and combine it in some way, outputting a single value for each key.

Despite being widely used, MapReduce is a very specific computational model and not all algorithms can be easily adapted to it. A practical PIR such as PIRMAP has to take its specifics into account – as we will see below.

3 Related Work Analysis

Despite the large amount of theoretical research, there has not been much investigation into practical PIR. Recently, Trostle and Parrish [20] experimented with the state of the art and found that it took at least 8 minutes to retrieve a file of size only 3 MB, out of a data set of 3 GB on commodity hardware. Additionally, existing schemes have significant communication overhead for files of that size and larger. For example, Lipmaa [13], enjoying the most efficient asymptotic communication complexity today, requires 30 MB of communication in the above scenario.

In examining related work, we make two nonstandard observations which apply to our MapReduce cloud setting. First, since computation for PIR must necessarily be $O(n \cdot l)$, computation will most likely be the bottleneck in any protocol. The time to communicate queries and responses will be relatively small in our setting, compared to the time it takes to calculate over the entire database. Therefore, it may be useful to trade some extra communication for smaller computational burden. The second observation relates to MapReduce itself. Since each mapper may run on a different node in the cloud, communication and synchronization within the cloud are very expensive. To take full advantage of the cloud, there should be few, if any, inter-dependencies between stages of computation.

Overview. One of the first cPIR schemes to achieve sublinear communication is that of Kushilevitz and Ostrovsky [12], using an additively homomorphic cipher $\mathcal{E}$ as follows:

1. The server arranges an n elements database as a $\sqrt{n} \times \sqrt{n}$ matrix
2. $\mathsf{Query}(x)$ – The user generates a vector $\boldsymbol{v}$ of length $\sqrt{n}$ where $\boldsymbol{v}_x = \mathcal{E}(1)$, and $\boldsymbol{v}_i = \mathcal{E}(0), \forall i \neq x$

3. Transfer($\boldsymbol{v}$) – Vector $\boldsymbol{v}$ is sent to the server. The server multiplies $\boldsymbol{v}$ with the database matrix and returns the result as vector $\boldsymbol{v}'$
4. Recover($\boldsymbol{v}'$) – $\boldsymbol{v}'$ now consists of $\sqrt{n}$ encrypted elements, one of which is the element the user is interested in. The user chooses this element and decrypts it, discarding the rest.

The scheme is sound, because the user sends a vector $\boldsymbol{v}$ which "zeroes out" all the rows in the matrix besides the row requested. The communication costs is $O(l \cdot \sqrt{n})$ which is still quite high for large values of n. Kushilevitz and Ostrovsky [12] also show that this protocol can be repeated recursively to achieve communication less than $O(n^c)$ for any $c > 0$, but at the cost of worse constants and computational complexity. This "square-root solution" computes only once over the database, but has impractically large communication costs.

Lipmaa [13] reduces communication complexity to $O(l \cdot \log n + k \cdot \log^2 n)$, where k is a security parameter. This is accomplished by generalizing the Kushilevitz and Ostrovsky [12] scheme into a family of protocols, parameterized by a dimension α. The Kushilevitz and Ostrovsky [12] scheme can be seen as two-dimensional, because the database elements are arranged in a $\sqrt{n} \times \sqrt{n}$ matrix. If $\alpha = \log(n)$, the database is viewed as a $\log(n)$ dimensional $2 \times 2 \times \ldots \times 2$ matrix. In this manner, the user may send a sequence of $\log(n)$ vectors, each only length 2 which "zero out" half of the remaining database elements at each step. It is conceptually similar to specifying a path in a binary tree, and when the leaf node is reached the only element that will remain non-zero is the one corresponding to that node.

The down side of this scheme is that it requires computing over the entire database *twice* ($\sum_{i=1}^{\log n} \frac{1}{2^i} = 2$). While not a significant problem asymptotically, as mentioned before, cPIR schemes are usually bottlenecked by their computational cost. Increasing this cost by 100% dramatically effects the efficiency of the scheme, especially given the cloud setting where twice the computation corresponds to twice the monetary cost. Another problem is that the computational complexity of Lipmaa [13] is $O(n \cdot l \cdot k^2)$, because it requires modular exponentiations on ciphertexts of size k. This is a significant overhead that we can avoid by using a more efficient homomorphism.

Although Lipmaa [13] requires only one round of communication with the user, it is iterative in that it requires $\log(n)$ rounds of computation on the server, each round depending on the output of the previous round and operating on a smaller set of data. As we will see, a large part of the cost for MapReduce is in initializing the parallel computation, so restarting MapReduce $\log(n)$ times adds a significant overhead.

More recently, Aguilar-Melamine and Gaborit [1] have proposed a scheme using lattice-based homomorphic encryption (similar to NTRU) which has much better computational complexity, both asymptotically and in practice. Their scheme also takes advantage of the fact that modern GPUs can solve large parallel problems very quickly. Aguilar-Melamine and Gaborit [1] are able to achieve fast query response times, but at the cost of a large communication cost. Additionally, their protocol is tuned specifically to the requirements of GPUs and would not scale well in a real-world distributed cloud environment like MapReduce.

Using a different encryption scheme based on the Trapdoor Group Assumption, Trostle and Parrish [20] propose a more traditional PIR scheme based on Kushilevitz and

Table 1. Communication and computational complexities of related work

	Communication	Computation
Kushilevitz and Ostrovsky [12]	$O(\sqrt{n \cdot l})$	$O(l \cdot n \cdot k^2)$
Lipmaa [13]	$O(l \cdot \log(n)^2)$	$O(l \cdot n \cdot k^2)$
Aguilar-Melamine and Gaborit [1]	$O(l + n \cdot k)$	$O(l \cdot n \cdot \log(k) \cdot \log(\log(k)))$
PIRMAP	$O(l + n \cdot k)$	$O(l \cdot n \cdot \log(k) \cdot \log(\log(k)))$

Ostrovsky [12]. They also achieve much faster query response, but again, with large communication costs.

Comparison of existing work. To understand how existing work compares and is suited to the application targeted in this paper, we start by examining asymptotic computation and communication complexities in Table 1. We can see that Lipmaa [13] has very good communication complexity with relatively bad computational complexity, while Aguilar-Melamine and Gaborit [1] have the opposite. While our protocol PIRMAP has the same asymptotic complexities than Aguilar-Melamine and Gaborit [1], this does not adequately describe the costs involved with PIR: we are primarily addressing a concrete, practical setting where constants become very important.

Consequently, in Table 2, we present analytically calculated communication costs for specific database and file sizes. Aguilar-Melamine and Gaborit [1] state that their requests are of size $25kb \cdot n$ and the responses are $6 \cdot l$. For databases with $l > n \cdot k$, Lipmaa [13] requires transmitting $\log^2(n)$ elements of size k in the request and receiving a response of size $\log(n) \cdot l$. We chose to use $k = 2048$, as suggested by Lipmaa [13].

In Table 2, we can see that even though Lipmaa [13] has good asymptotic communication, there is room for improvement in a concrete setting, especially for databases composed of large files. Aguilar-Melamine and Gaborit [1], similarly to PIRMAP, is designed specifically for these types of databases but it also has relatively high communication costs due to large constants.

Our approach. In conclusion, we would like to achieve, in real world settings, the small communication cost of Lipmaa [13] with the fast query response times of Aguilar-Melamine and Gaborit [1] and Trostle and Parrish [20]. Lipmaa notes that, when $\alpha = 1$, the PIR scheme is particularly efficient for databases where $l > n \cdot k$. We choose this case as a base for our new protocol PIRMAP, because it requires only one iteration of the database during Transfer and has communication complexity of $O(l + n \cdot k)$. When $l > n \cdot k$, this is optimal, since the server response must always be of size l. In short, PIRMAP can be thought of as a combination of this protocol with the fast encryption scheme used by Trostle and Parrish [20]. This new "somewhat" homomorphic encryption scheme requires only modular multiplications rather than exponentiations, significantly improving the asymptotic and real world computational requirements.

Table 2. Real communication costs of related work in MB for different database sizes

	1 MB Files			5 MB Files			10 MB Files		
	1 GB	10 GB	100 GB	1 GB	10 GB	100 GB	1 GB	10 GB	100 GB
Lipmaa	10	13	17	38	55	71	66	100	133
Aguilar-Gaborit	29	240	2350	35	77	499	62	83	295
PIRMAP	3	6	38	11	12	18	22	22	26

4 PIRMAP

PIRMAP modifies the PIR protocol by Lipmaa [13], specifically addressing the shortcomings perceived in regards to retrieval of large files in a parallelization-aggregation computation framework such as MapReduce. We start by giving an overview of PIRMAP which can be used with *any* additively homomorphic encryption scheme.

Upload. In the following, we assume that the cloud user has already uploaded its files into the cloud using the interface provided by the cloud provider.

Query. Our data set holds n files, each of l bits length. Additional parameter k specifies the block size of a cipher. For ease of presentation, we consider the case where all files are the same length, but PIRMAP can easily be extended to accommodate variable length files using padding. PIRMAP is summarized as follows:

1. Query – If the user wishes to retrieve file $1 \leq x \leq n$ out of the n files, he creates a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$, where $v_x = \mathcal{E}(1)$ and $v_i = \mathcal{E}(0) \forall i \neq x$. Here, $\mathcal{E}$ is any IND-CPA additively homomorphic encryption scheme. The user sends $\boldsymbol{v}$ to the cloud.
2. Multiply – The cloud arranges files into a table $\mathcal{T}$ similar to Table 3. The cloud divides each file i into $\frac{l}{k}$ blocks $\{B_{i,1}, \ldots, B_{i,\frac{l}{k}}\}$ and multiplies each block by v_i, i.e., $B'_{i,j} = v_i \cdot B_{i,j}$. Here, "$\cdot$" denotes scalar multiplication.
3. Sum – The cloud adds column-wise to create one result vector $\boldsymbol{r} = (r_1, \ldots, r_{\frac{l}{k}})$. Each element r_i of that vector is of size k, so the total bit length of $\boldsymbol{r}$ is l bits. Vector $\boldsymbol{r}$ is an encryption of file x. Vector $\boldsymbol{r}$ is returned to the user who decrypts it.

Table 3. Cloud splits files into pieces

	1	2	$\cdots$	$\frac{l}{k}$
$file_1$	$B_{1,1}$	$B_{1,2}$	$\cdots$	$B_{1,\frac{l}{k}}$
$file_2$	$B_{2,1}$	$B_{2,2}$	$\cdots$	$B_{2,\frac{l}{k}}$
$\cdots$		$\cdots$		
$file_n$	$B_{n,1}$	$B_{n,2}$	$\cdots$	$B_{n,\frac{l}{k}}$

The cloud performs the matrix-vector multiplication $\boldsymbol{r} = \boldsymbol{v} \cdot T$. In practice, the cloud does not need to create a table, but just needs to perform the block wise scalar multiplications and additions.

The computation consists of multiplying each of the $n \cdot \frac{l}{k}$ blocks by a value in the request vector and then performing $(n-1) \cdot \frac{l}{k}$ additions to obtain the final sum. The computational complexity of this scheme is $O(n \cdot l \cdot M(k) + (n-1) \cdot l \cdot A(k)$, where $M(k)$ is the cost of performing one scalar multiplication in the additively homomorphic encryption system, and $A(k)$ is the cost of one addition. The communication complexity can be broken down into two parts: user to cloud and cloud to user. The user sends a vector of size n containing ciphertexts of size k, so the bandwidth complexity user-cloud is $O(n \cdot k)$. The cloud sends back a vector of $\frac{l}{k}$ entries, each of size $O(k)$, resulting in complexity $O(l)$. This results in the overall communication complexity of $O(n \cdot k + l)$.

PIRMAP achieves better overhead than related work [4, 13] in practice, due to the specific values of n and l in our setting. Existing work in PIR is efficient *only* for data sets with large values for n and small values for l. PIRMAP would do poorly in that setting, because the complexity would be dominated by n. However, PIRMAP's cloud-to-user communication is *optimal* at $O(l)$, because the cloud must send back a message at least the size of the file the user queries for. For values of l larger than $n \cdot k$, PIRMAP allows for complexity of $O(l)$ with small constants. We argue that in practice this is often true. For example, if a user has a 1 TB data set of 10 MB files, $\frac{l}{n} \approx 800$. In contrast, under the same conditions, arranging the files in a $\sqrt{n} \times \sqrt{n}$ matrix would result in a download cost of $1024 \cdot 10\,\text{MB} \approx 10\,\text{GB}$. Even if $n > l$, the actual communication costs are low for practical choices of the parameters, see Section 5. Many cloud providers such as Amazon do not charge the user for uploads, but only for downloads. This is an additional benefit of our scheme, because the query result (communication the user is charged for) is always very close to optimal, even when the overall communication is not.

Our assumption of $l > n$ is realistic and useful for many real-world applications. For example, medical records may contain one or more images (x-ray, MRI, etc) which would make them several megabytes at the very least.

Optimization: Although the cloud performs most of the computation in this scheme, the user is still required to generate a vector of ciphertexts of length n and then decrypt the resulting response. As encryption is relatively expensive for additive IND-CPA ciphers, this might require a non-trivial amount of computation that might hurt, e.g., users with low-powered devices such as smartphones. A way to alleviate this problem is to have a moderately powerful trusted server to pre-generate vectors of ciphertexts and upload them to the cloud for the users to use. This trusted server would generate m "disposable" vectors of size n such that $V_{i,j} = \mathcal{E}(1)$ where $j = \text{HMAC}(i)$ and $V_{i,j} = \mathcal{E}(0)$ otherwise. This allows the user to use one of these disposable vectors at query time and permute it so that the single $\mathcal{E}(1)$ is at the index of the file it wishes to retrieve. If the key used in the HMAC is shared between the user and trusted server, the user can efficiently locate $\mathcal{E}(1)$. The user then generates a description of a permutation which moves the $\mathcal{E}(1)$ value to the correct position and randomly shuffles all other locations. A description of this permutation is of size $n \cdot \log(n)$ which is smaller than the size of the vector for $k > 30$. This also effectively front loads the upload cost of the query and makes response time even faster.

Although the particular encryption scheme we use (see Section 4.2 below) can perform encryptions very quickly and does not require the use of this optimization, we point it out as a general improvement regardless of the cipher used.

4.1 PIRMAP Specification

In our protocol, the cloud performs two operations: multiplication of each block by the corresponding value in the "PIR vector" $\boldsymbol{v}$, and column-wise addition to construct the encrypted file chosen by the user. These two stages translate exactly to map and reduce implementations respectively. The files will be distributed evenly over all participating nodes where the map function will split each file into blocks and multiply the blocks by the correct encrypted value. The output of these mappers is a set of key-value pairs where the key is the index of the block, and the value is the product of the block and encrypted PIR value. These values are all passed on to the reducers, which take a set of values for each key (block position or column) and add them together to get the final value for each block.

Being interested in $file_x$, the user executes the above Query to compute $\boldsymbol{v}$ which is sent to the MapReduce cloud. There, each mapper node evaluates Multiply on its locally stored file and generates key-values pairs for the reducer. The reducer simply computes the Sum step by adding all values with the same key and sending them back to the user. The user receives $\frac{l}{k}$ values of size k from the reducers and decrypts to get $file_x$.

User

function GenQuery(n, x)
  $v := \{\}$
  for $i = 1$ **to** n **do**
    if $i = x$ **then**
      $v_i := \mathcal{E}(1)$
    else
      $v_i := \mathcal{E}(0)$
    end if
  end for
end function

Cloud

function Map$(file, v)$
  for $i = 1$ **to** n **do**
    $c := B_i \cdot v_i$
    $Emit(i, c)$
  end for
end function
function Reduce(key, v)
  $total := 0$
  for $i = 1$ **to** n **do**
    $total := total + v_i$
  end for
  $Emit(key, total)$
end function

4.2 Encryption Scheme

Since the map phase of our protocol involves multiplying every piece of the data set by an encrypted PIR value, it is important that we choose an efficient cryptosystem. Traditional additively homormorphic cryptosystems, such as Paillier's, use some form of multiplication as their homomorphism. That is, for elements a and b, $\mathcal{E}(a) \cdot \mathcal{E}(b) = \mathcal{E}(a + b)$. Since our map phase consists of multiplying ciphertexts by unencrypted scalars, we would have to do exponentiation of a ciphertext. PIRMAP, and all PIR schemes, must compute on the whole data set, so this step would be computationally intensive.

Similar to our recent findings [3], we mitigate this problem by using a *somewhat homomorphic* encryption scheme introduced by Trostle and Parrish [20] that relies on the

trapdoor group assumption. Fully homomorphic encryption schemes support an unlimited number of computations without increasing the size of the ciphertexts. In contrast, this scheme results in ciphertexts which grow in size by $O(\log_2 n)$ bits for n additions. In return for this size increase, we can have an encryption scheme where the additive homomorphism is integer addition. This scheme encrypts n bits with security parameter $k > n$ as follows:

KeyGen(1^k): Generate a prime m of k bits and a random $b < m$; b and m are secret.
Encrypt $\mathcal{E}(x) = b \cdot (r \cdot 2^n + x) \mod m$, for a random r
Decrypt $\mathcal{D}(c) = b^{-1} \cdot c \mod m \mod 2^n$

This encryption has the desired homomorphic property $\mathcal{E}(a) + \mathcal{E}(b) = \mathcal{E}(a + b)$. This scheme is *somewhat homomorphic*, because it cannot support an unlimited number of additions. When two ciphertexts c_1 and c_2 are added, you can express the sum as

$$b \cdot (r_1 \cdot 2^n + x_1) + b \cdot (r_2 \cdot 2^n + x_2) = b \cdot ([r_1 + r_2] \cdot 2^n + x_1 + x_2).$$

As m remains secret, note that the cloud performs addition as integer addition, not modulo m. If the inside term $(r_1 + r_2) \cdot 2^n + x_1 + x_2$ exceeds m and "wraps around", then it will not be decrypted correctly, because application of the modulus will cause a loss of information. The modulus m must be chosen large enough to support the number of additions expected to occur. To support t additions, m should be increased by $\log_2 t$ bits. Additionally, each scalar multiplication can be thought of as up to 2^n additions, meaning that the size of m must be doubled for each supported scalar product. For our PIR scheme, m must be chosen to be $O(2k + \log_2(n))$ to support the required homomorphic operations.

In return for the reasonable increase in ciphertext size caused by the larger modulus (about 300% in our evaluations in Section 5), we are able to do very efficient computations over the encrypted data. Additionally, encryption is equivalent to only two multiplications, an addition and a modular reduction, while decryption is one multiplication and a reduction. This compares very favorably with other homomorphic encryption schemes, such as Paillier, requiring a modular exponentiation.

With our encryption scheme defined, we can now express more precise computational complexities for the protocol. Our previous complexity was parameterized over $M(k)$ and $A(k)$, the complexity of scalar multiplication and addition, respectively. For our encryption, addition is simply regular integer addition. Since each cipher text is at most $2k + \log_2(n)$ bits long, addition is $O(2k + \log_2(n))$. We can do scalar multiplication as integer multiplication as well. Integer multiplication can be done for m bits in $O(m \log(m) \log(\log(m)))$ [9], so $M(k) = O(2k + \log_2(n) \log(2k + \log_2(n)) \log(\log(2k + \log_2(n))))$. The complexity is then dominated by the multiplication cost and results in:

$$O(n \cdot \frac{l}{k} \cdot (k + \log(n)) \log(k + \log(n)) \log(\log(k + \log(n)))) \quad (1)$$

$$= O((n \cdot l + n \cdot \log(n)) \log(k + \log(n)) \log(\log(k + \log(n)))) \quad (2)$$

If $l > n$, then $l > \log(n)$, and we can simplify to $O(n \cdot l \cdot \log(k + \log(n)) \cdot \log(\log(k + \log(n))))$. Additionally, k has to be much larger than $\log(n)$, otherwise the server has

the resources to find key (m, k) by brute force. This allows us to simplify to $O(n \cdot l \cdot \log(k) \cdot \log(\log(k)))$.

4.3 Privacy Analysis

PIRMAP inherits privacy properties of the work it is based on, i.e., Lipmaa [13] and the PIR variant by Trostle and Parrish [20]. In the following, we sketch our privacy rationale.

PIRMAP is privacy-preserving, **iff** an adversary (the cloud) cannot guess, after each query, with probability greater than $1/n$, which file was retrieved by the user after an invocation of the protocol. There are two pieces of information that the adversary has access to: the set of uploaded files and the vector $\boldsymbol{v}$ of PIR values. The uploaded files are independent of any encryption used in the PIR protocol and can be efficiently simulated by the adversary. Therefore, privacy depends only on $\boldsymbol{v}$.

Vector $\boldsymbol{v}$ contains many encryptions of "0" and one encryption of "1". The problem of determining which file was selected is then equivalent to *distinguishing* between encryptions of "0" and encryptions of "1" in the underlying encryption. However, the scheme we use is provably secure against distinguishing under the Trapdoor Group Assumption [20]. Consequently, PIRMAP preserves user privacy.

5 Evaluation

We have evaluated the performance of our scheme in three contexts: a local "cloud" (a single server with multiple CPUs), a commodity laptop, and Amazon's EC2 cloud using Elastic MapReduce [2]. We have implemented PIRMAP in Java for standard Hadoop MapReduce version 1.0.3 and the source code is available for download [17].

5.1 Setup

Local We used a local server to prototype and debug our application and to do detailed timing analysis requiring many runs of MapReduce. This server, running Arch Linux 2011.08.19, has a dual 2.4 GHz quad-core Xeon E-5620 processor and 48 GB of memory. Based on specs and benchmark results, this local server is closest to an "EC2 Quadruple Extra Large" instance, which has dual 2.9 GHz quad-core Xeon X-5570 processors and 24 GB of memory.

We have measured the time needed for PIR queries, i.e., the time to upload PIR vector $\boldsymbol{v}$ plus the time to process the query and download the result. Using Amazon's standard cost model, we have calculated the price of each PIR query as the amount of money required to run the query on one of the above EC2 instances (for the same amount of time it took to run locally) plus the bandwidth cost of downloading the results [2]. Since uploading data to Amazon is free, this does not add any additional cost. To put our measurements into perspective, we have also evaluated time and cost of two other, hypothetical, PIR protocols. We have implemented a *Baseline*, which does not perform *any* cryptographic operations and merely "touches" each piece of data through the MapReduce API. This measure shows the theoretical *lower bound* of computation

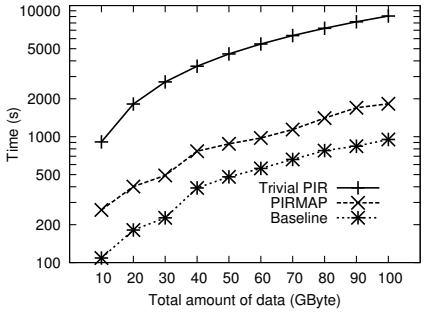


Fig. 1. Time per query, local server

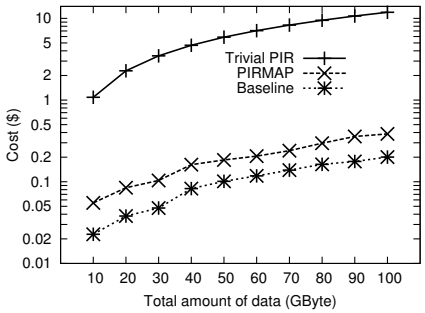


Fig. 2. Cost per query, local server

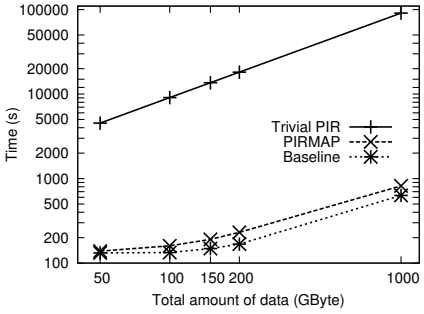


Fig. 3. Time per query, Amazon

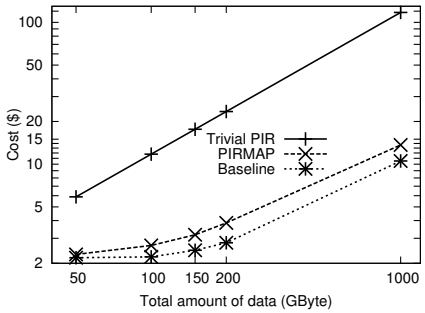


Fig. 4. Cost per query, Amazon

and time required for *any* PIR scheme that uses MapReduce, independent of the encryption and exact PIR method used.

To highlight the advantage of computational PIR, we have also included the time and cost required for the "trivial" PIR scheme. The trivial scheme is one where the user downloads the entire data set and simply discards the files he is not interested in. This is very bandwidth intensive, but computationally lightweight. Sion and Carbunar [18] conjecture this trivial PIR to be the most cost effective in the real-world. We have calculated the cost based on the amount that Amazon charges to download the corresponding amount of data and the time based on a 11.28 Mbps connection, an average as reported by Nasuni [15]. Note that we generally do not count the cost for long-term storage of data at Amazon. Although potentially significant for large amounts of data, the user has to pay for this regardless of whether he wants queries to be privacy-preserving or not. PIRMAP does not increase the amount of storage in Amazon.

Laptop. We also ran our implementation on a 2012 MacBook Pro with a 2.3 Ghz i7, 8 GB of RAM and an NVIDIA GeForce 650M. The purpose of this test was to compare query time with related work, and the results are shown in Table 4. The Aguilar scheme takes advantage of GPU resources, so we chose a machine with comparable graphics and GPU resources (unlike the above server, which had no discrete

Table 4. Query time in minutes, including generation, evaluation, and decryption, for databases of varying sizes, composed of 5 MB files

	5 GB	10 GB	15 GB	20 GB	25 GB	30 GB
Lipmaa	1852	3704	5508	7312	9116	10920
Aguilar-Gaborit	4	7	11	14	17	20
PIRMAP	1	2	3	3	4	4

graphics). This represents the performance you would get on a commodity machine and also shows that query generation is not very taxing. To compare with Lipmaa [13], we used `openssl speed rsa` to time determine how many RSA private key operations can be computed per second. Each scalar multiplication in that scheme is equivalent to one modular exponentiation, or one RSA private key operation as required by Lipmaa [13]. This estimation is quite generous for Lipmaa [13], because it does not include disk access or homomorphic additions (modular multiplication), but we believe it is close due to the time being largely dominated by exponentiations.

Amazon. Besides the local experiments, to demonstrate the scalability of PIRMAP, we have also evaluated it on Amazon's Elastic MapReduce cloud. Amazon imposes a maximum limit of 20 instances per MapReduce job by default. In keeping with this restriction, we used 20 "Cluster Compute Eight Extra Large" instances which each having dual eight-core Xeon E5-2670 processors and 64 GB of RAM.

5.2 Results

Time and Total Cost: Figures 1 to 4 show our evaluation results. Figures 1 and 2 show the local server evaluation, while figures 3 and 4 show evaluation with Amazon. In each figure, the x-axis shows the total amount of data stored at the cloud, i.e., number of files n times file size l. The y-axis shows either the time elapsed (for the whole query from the time the user submits the query until MapReduce returns the result back) or the cost implied with the query. In all four graphs, we scale the y-axis logarithmically, and in figures 3 and 4 we also scale the x-axis logarithmically. Each data point represents the average of at least 3 runs. Relative standard deviation was low at $\approx 5\%$.

To verify the impact of varying file sizes l, for our local evaluation, we show results with file sizes of 1 MB and 3 MB, attaining approximately equal runtime in both cases. This is to be expected because, in each data point, we have fixed the size of the database so varying retrieval sizes merely reshapes the matrix without changing the number of elements in it. The execution is dominated by the scalar multiplications that occur during the map phase, and the same number of those is required no matter the dimension of the matrix.

Our evaluation shows that PIRMAP outperforms trivial PIR in both time and cost by one order of magnitude. We also compare performance to a baseline PIR protocol implemented in MapReduce where the mappers simply read and ignore any rows that are no requested and return the single row that was. Any MapReduce-based PIR protocol will require at least this amount of computation. Compared to this theoretical, yet

unrealistic optimum, PIRMAP introduces only 20% of overhead in the case of Amazon. Locally, we experience slightly larger overhead of 100%. This is because executing on Amazon has a much higher "administrative" cost due to the higher number of nodes and more distributed setting. These results indicate not only PIRMAP's efficiency over Trivial PIR and Baseline, but also its real-world practicality: in a small database comprising 10,000 files of size 1 MB each (10 GByte), a user can retrieve a single record in ≈ 3 min for only $\approx \$0.03$. In a huge data set with $1,000,000$ files, a single file can be retrieved in ≈ 13 min for $\approx \$14$. In scenarios where it is necessary to retrieve data in a fully privacy-preserving manner, we conjecture that this to be acceptable.

Although a comparison with related research is not straightforward (as PIRMAP targets a very special scenario), we put our results into perspective with those of Aguilar and Lipmaa. We show that, while Lipmaa's scheme has very good communication complexity in all cases, it is completely impractical due to the enormous amount of computation needed to respond to queries. We also show that our scheme is comparable with that of Aguilar, in terms of computation, and beats it in communication cost by a significant margin.

Query Generation and Decryption: Due to the efficiency of the encryption in PIRMAP, PIR query generation is very fast. One ciphertext (element of $\boldsymbol{v}$) is generated for each file in the cloud, so the generation time is directly proportional to the number of files. We omit in-depth analysis, but in our trials on our commodity laptop running on a single core it takes about 2.5 seconds per 100,000 files in the cloud. Decryption is slightly more expensive than encryption, but we still managed, on the same machine, to decrypt approximately 3 MB per second. We conclude this overhead to be feasible for the real-world.

Bandwith: PIRMAP introduces bandwith overhead, 1.) to *upload* $\boldsymbol{v}$, and 2.) to *download* the encrypted version of the file. For security, we set $k = 2048$ bit, so each of the n elements of vector $\boldsymbol{v}$ has size 2048 bit. For a data set with 10,000 files (10 GByte), this requires the user to upload ≈ 2.5 MByte. As this can become significant with larger number of files, we suggest to then use the optimization in Section 4, especially for constrained devices.

6 Conclusion

Retrieval of previously outsourced data in a privacy-preserving manner is an important requirement in the face of an untrusted cloud provider. PIRMAP is the first practical PIR mechanism suited to real-world cloud computing. In the case where a cloud user wishes to privately retrieve large files from an untrusted cloud, PIRMAP is communication efficient. Designed for prominent MapReduce clouds, it leverages their parallelism and aggregation phases for maximum performance. Our analysis shows that PIRMAP is an order of magnitude more efficient than trivial PIR and introduces acceptable overhead over non-privacy-preserving data retrieval. Additionally, we have shown that our scheme can scale to cloud stores of up to 1 TB on Amazon's Elastic MapReduce.

Acknowledgments: This work was partially supported by NSF grant 1218197.

References

[1] Aguilar-Melamine, C., Gaborit, P.: A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol (2007), http://eprint.iacr.org/2007/446.pdf

[2] Amazon. Elastic MapReduce (2010), http://aws.amazon.com/elasticmapreduce/

[3] Blass, E.-O., Di Pietro, R., Molva, R., Önen, M.: PRISM – Privacy-Preserving Search in MapReduce. In: Fischer-Hübner, S., Wright, M. (eds.) PETS 2012. LNCS, vol. 7384, pp. 180–200. Springer, Heidelberg (2012)

[4] Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)

[5] Chen, Y., Sion, R.: On securing untrusted clouds with cryptography. In: Workshop on Privacy in the Electronic Society, Chicago, USA, pp. 109–114 (2010)

[6] Chor, B., Goldreich, O., Kushilevitz, E.: Private Information Retrieval. In: Proceedings of Symposium on Foundations of Computer Science (1995)

[7] Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: Proceedings of Symposium on Foundations of Computer Science, pp. 41–50 (1995)

[8] Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Proceedings of Symposium on Operating System Design and Implementation, San Francisco, USA, pp. 137–150 (2004)

[9] Fürer, M.: Faster integer multiplication. In: Proceedings of Symposium on Theory of Computing (1997)

[10] Gartner. Gartner Identifies the Top 10 Strategic Technologies for 2011 (2010), http://www.gartner.com/it/page.jsp?id=1454221

[11] Google. A new approach to China (2010), http://googleblog.blogspot.com/2010/01/new-approach-to-china.html

[12] Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: Proceedings of Symposium on Foundations of Computer Science, pp. 364–373 (1997)

[13] 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)

[14] McClure, D.: GSA's role in supporting development and deployment of cloud computing technology (2010), http://www.gsa.gov/portal/content/159101

[15] Nasuni: State of Cloud Storage Providers Industry Benchmark Report (2011), http://cache.nasuni.com/Resources/Nasuni_Cloud_Storage_Benchmark_Report.pdf

[16] Olumofin, F., Goldberg, I.: Revisiting the Computational Practicality of Private Information Retrieval. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 158–172. Springer, Heidelberg (2012)

[17] PIRMAP. Source Code (2012), http://pasmac.ccs.neu.edu

[18] Sion, R., Carbunar, B.: On the Computational Practicality of Private Information Retrieval. In: Proceedings of Network and Distributed Systems Security Symposium, San Diego, USA, pp. 1–10 (2007)

[19] Symantec. State of Cloud Survey (2011), http://www.symantec.com

[20] Trostle, J., Parrish, A.: Efficient computationally private information retrieval from anonymity or trapdoor groups. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 114–128. Springer, Heidelberg (2011)

[21] Whittaker, Z.: Microsoft admits Patriot Act can access EU-based cloud data (2011), http://www.zdnet.com/

Avoiding Theoretical Optimality to Efficiently and Privately Retrieve Security Updates

Justin Cappos
NYU Poly
jcappos@poly.edu

Abstract. This work demonstrates the feasibility of building a PIR system with performance similar to non-PIR systems in real situations. Prior Chor PIR systems have chosen block sizes that are theoretically optimized to minimize communication. This (ironically) reduces the throughput of the resulting system by roughly 50x. We constructed a Chor PIR system called upPIR that is efficient by choosing block sizes that are theoretically suboptimal (from a communications standpoint), but fast and efficient in practice. For example, an upPIR mirror running on a three-year-old desktop provides security updates from Ubuntu 10.04 (1.4 GB of data) fast enough to saturate a T3 link. Measurements run using mirrors distributed around the Internet demonstrate that a client can download software updates with upPIR about as quickly as with FTP.

Keywords: Private Information Retrieval, Performance, Practical Security.

1 Introduction

Each year, thousands of vulnerabilities in software are discovered and fixed. To fix a vulnerability, a computer will request and install a security update. However, the request to retrieve a security update is very much a public action. Most software updaters do not encrypt the request for a security update in any way and the request itself is often directed to an untrustworthy party like a mirror. For example, Cappos [1] set up an official mirror for popular Linux distributions using dubious credentials and rented hosting. The official mirrors received requests for security updates (and thus a notification that the requesting system is unpatched) from a large number of computers including banking, government, and military computers. Thus the act of fixing a security vulnerability ironically also notifies potential attackers that the client has a security vulnerability!

Fortunately, Private Information Retrieval (PIR) [2] addresses this issue. There are now myriad schemes proposing how clients can retrieve information from a database without disclosing which information is requested [2–5]. The academic literature has primarily optimized these systems by improving their theoretical properties [6–9], primarily to reduce communications overhead.

The biggest open problem related to PIR systems is how to make them practical. An academic panel titled "Achieving Practical Private Information Retrieval" lamented that the performance of existing PIR systems makes them

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 386–394, 2013.

unsuitable for practical use [10]. Recently, Sion suggested that many PIR techniques are so inefficient that it is faster to simply transmit all data stored on the server to the client [11]. More recently, Olumofin and Goldberg [12] have shown faster practicality results (especially with Chor PIR); however, these results are still much slower than non-PIR systems.

We demonstrate that it is possible to build a practical PIR system that provides performance similar to that of non-PIR production systems. Our system, upPIR uses the Chor multi-server PIR scheme [2], which uses XOR instructions that can be efficiently computed on modern hardware. By carefully choosing the block size to match the processor's cache size, upPIR's throughput is substantially faster than existing results. (This is opposed to prior work which has focused on reducing communication complexity.) upPIR allows clients to retain information-theoretic privacy while providing performance similar to popular HTTP and FTP servers.

2 Software Updaters

Software Updater Architecture. The architecture of software update systems (including upPIR) consists of three parties. The software vendor, such as Ubuntu or Microsoft, creates a set of updates and bundles them into a *release*. The vendor also creates some metadata that describes the release, called a *manifest*. The release is obtained and copied by a set of mirrors. For economic and configurability reasons, mirrors are an important and essential part of the software update landscape. Unfortunately, it is trivial for a malicious party to register as an official mirror and receive requests from clients, including requests for security updates [1].

Software Update Contents. The size and number of items stored by a mirror vary over software projects, ranging between .5GB and 4.3GB for recent versions of popular Linux distributions. The size of the security updates for a distribution is several orders of magnitude smaller than the full mirror data which contains normal updates. In this work we focus on distributing security updates and leave private distribution of complete software mirrors for future work. Further details about the suitability of PIR for software updates are provided in a tech report [13].

3 Threat Model

In our threat model, a client may contact many mirrors, including those that may be honest-but-curious. Our goal in this work is to prevent a mirror from knowing which software update is being retrieved by a vulnerable client. We assume that:

- The vendor is creating valid updates that the client wishes to retrieve.
- A non-malicious mirror may fail at any time.

- A malicious party may operate one or more mirrors. These mirrors may share or publicize any information they receive.
- An adversary may be able to observe all traffic sent over the network. This is consistent with a malicious access point or ISP.

4 Architectural Overview

Vendor. The vendor produces a set of updates that it wishes to package into a *release* and provide to clients. It also provides a list of mirrors to clients. The vendor generates a *manifest* that contains metadata about the updates provided in the release including secure hashes of the files. The release provided by a vendor conceptually breaks the updates into equally sized blocks. If this were not done, then performing an XOR of all updates together causes every XORed chunk of data to be the size of the largest update. This would effectively mask the size of the update being retrieved, but would be very inefficient if there is a wide distribution of update sizes. In our implementation, the vendor selects the block size when the manifest is created. (Section 5.3 discusses how to choose an efficient block size).

Mirror. An upPIR mirror obtains the files for the release from the vendor using `rsync` or another file transfer mechanism for distributing updates to mirrors. Following this, the mirror reads in all of the software updates in the release and stores them in one contiguous memory region. (The order of the software updates in memory is specified in the manifest file.) The mirror uses the manifest to validate each block. The mirror then notifies the vendor's server that it is ready to serve blocks to clients. The mirror provides the vendor with a public key to prevent a man-in-the-middle from viewing client requests. When a client sends a string of bits to the mirror, the mirror will XOR together all blocks with a 1 in their position of the client's request string. The mirror then sends the result back to the client (which is the size of one block). This response over an encrypted channel and is signed by the mirror's private key to provide non-repudiation. Note that the mirror can concurrently serve (non-private) FTP and HTTP requests.

Client. A client first contacts the vendor's server to obtain the latest manifest and mirror list. From the manifest, the client can determine which blocks of the release it needs to retrieve in order to receive its update. The client also has some value N that represents the number of mirrors that would have to collude to compromise the client's privacy. To retrieve a single block, the client generates $N - 1$ cryptographically suitable random strings. The client derives the Nth string by XORing the other $N-1$ random strings together and flipping the bit of the desired update. Each string is sent to a different mirror over an encrypted channel (to prevent eavesdropping on the strings). Each mirror returns a block consisting of the specified blocks XORed together. The mirror signs the request and response in its reply to allow the client to demonstrate to a third party when a mirror is corrupt or malicious. If multiple blocks are desired, the procedure is repeated.

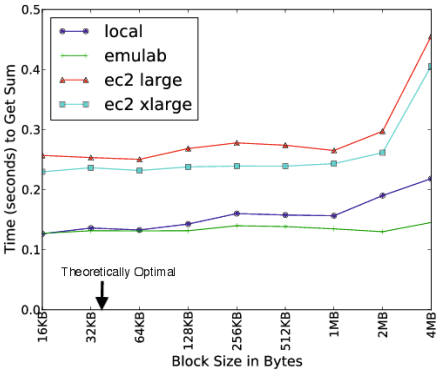


Fig. 1. Time to fetch one block against the Ubuntu release on multiple machines

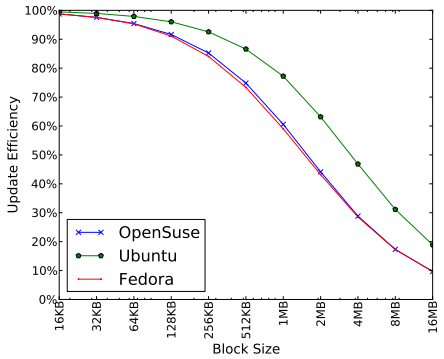


Fig. 2. Space efficiency of updates with various block sizes

5 Evaluation

This section compares the impact of different block size choices. Our focus is on examining whether the theoretically optimal block size from a communications perspective [12] (where the block size is the square root of the database size) results in good throughput. We also examine the performance of upPIR on real data sets and in a realistic deployment using the block size we recommend. Since mirrors are often set up using outdated server hardware or in a VM on shared resources. The different machines used are as follows:

- **ec2 large** and **ec2 xlarge** are Amazon EC2 instances [14].
- **emulab** is an three-year-old Emulab node with an Intel E5530 CPU with an 8MB L2 cache and 12GB of RAM on a virtual 100Mbps LAN [15, 16].
- **local** is an undergraduate student's three-year-old PC with an Intel E5506 CPU with a 4MB L2 cache and 6GB of RAM on a shared 100Mbps LAN.

5.1 Mirror XOR Microbenchmarks

Figure 1 demonstrates the time it takes to produce a block of data when a mirror serves the Ubuntu 10.04 data. We generated 10 random bit strings of the appropriate size and then measured the amount of time the mirror spent XORing the relevant update blocks together. Notice that the theoretically optimal block size from a communications standpoint [12] has essentially the same speed as block sizes up to 1MB. Both a 1MB and 2MB block size allow local and emulab to produce blocks quickly enough to saturate a T3 link. Once the block size increases to 2MB, the throughput no longer increases linearly with the block size due to data and code not fitting entirely in L2 cache. Producing a 1MB block in the same time as the theoretically optimal block size results in about a 50x increase in throughput.

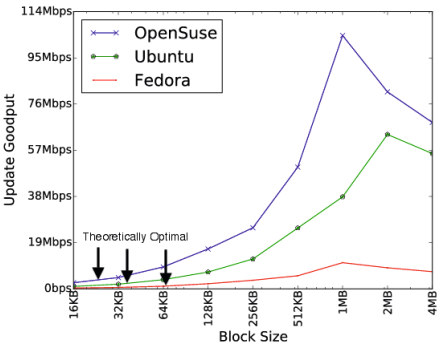


Fig. 3. Goodput for an average sized update in three releases

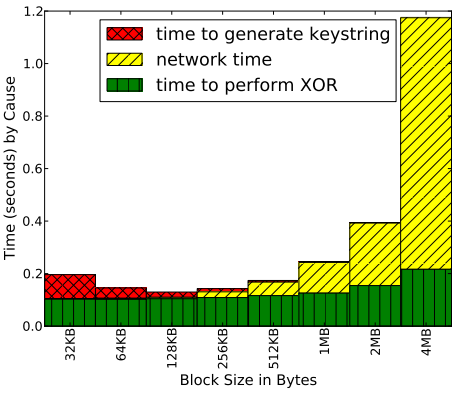


Fig. 4. Time to privately fetch one block

The release size is an important factor in speed because larger releases contain more blocks. To explore the impact of the release size, we fixed the block size to be 1MB and then varied the release size on 'emulab' (not shown) The performance scales at the same rate as the release size until the release no longer fits in memory. Once the release exceeds the size of RAM, the performance drops by over an order of magnitude as disk latency comes into play (not shown). Our results show that upPIR scales linearly as the release size grows provided the data served fits within memory.

5.2 The Impact of Block Size on Efficiency

The previous discussion showed how quickly a mirror could produce XOR blocks. However, there is a difference between useful data and data. If a mirror can produce a 1MB block in .1 second or a 2MB block in .15 second, from a throughput standpoint, the 2MB block size is superior. However, if the client wants a 1MB update but must retrieve 2MB of data to get it, then 1MB of the space is wasted. In essence, the data efficiency is the amount of retrieved data that is useful.

Figure 2 shows how changing the block size impacts efficiency for different data sets. The lines represent different update sizes (or data sets) and illustrate the performance difference when block size is varied. The values given were calculated by dividing the size of the release by the amount of data that a client would have to download to obtain every update in it one update at a time using our PIR scheme. This figure 2 demonstrates that the amount of useful data within a block decreases rapidly as the block size increases over 2MB. This is to be expected since larger blocks imply that there is more wasted space when retrieving an update. For example, between 70-85% of update data is unneeded when using 8MB blocks, but less than 5% is unneeded with 64KB blocks.

5.3 Choosing a Block Size to Optimize Goodput

One decision the vendor makes when creating the manifest for a release is to choose the block size. As we previously saw, this choice greatly impacts both the mirror XOR performance and the client's goodput. (Goodput is defined as the desired bytes per second, so ignores padding and packet headers.) In order to determine how to optimize the mirror's goodput, one can combine the mirror XOR time and the data efficiency to compute the goodput of the mirror.

Figure 3 shows how the goodput varies based on the throughput of the mirror and the space efficiency of the block size. This chart is generated by retrieving an average sized update from three distributions on the system 'local'. (Other systems show qualititatively similar results.) This graph shows that the goodput is optimal when block size is between 1MB to 2MB for each distribution. As a result, these block sizes seem to be the most efficient for this system. The theoretically optimal from a communications standpoint (the square root of the distribution size) has one to two orders of magnitude less throughput.

5.4 Controlled Macrobenchmarks

Figure 4 shows where time is spent retrieving a block from a mirror on emulab. First of all, the time to generate the cryptographically suitable random string is only paid on the client side of the connection. Similarly, the time that is spent XORing content is only performed by the mirror. The communication time is perceived by both systems. When the block size is small, the client's time to generate the string incurs a non-negligible cost. For larger block sizes, the network communication time is the dominant factor. For example, for a 4MB block size, the XOR takes about 200 ms, and the retrieval time is nearly 1 second. The theoretically optimal block size from a communications standpoint (about 38KB) has about 100 times slower throughput than a 2MB block.

5.5 Deployment

To understand the performance of uPPIR in realistic environments, we deployed our software on machines around the world. We used our machine 'local' as an Ubuntu 10.04 vendor with a 2MB block size, three EC2 instances in either the US East, US West, or EU West availability zones as mirrors, and ran the client at the University of Washington. For the worldwide setting, we ran one mirror in each availability zone. We compared the time to download the 1.5MB

Table 1. Ubuntu update times

Location	Protocol	Time
US-West (EC2)	HTTP	.64s
US-West (EC2)	FTP	1.5s
US-West (EC2)	uPPIR	1.2s
US-East (EC2)	HTTP	1.5s
US-East (EC2)	FTP	2.1s
US-East (EC2)	uPPIR	3.1s
EU-West (EC2)	HTTP	2.7s
EU-West (EC2)	FTP	4.5s
EU-West (EC2)	uPPIR	4.1s
US Mirrors	HTTP	1.6s
US Mirrors	FTP	2.1s
Worldwide (EC2)	uPPIR	3.5s

`libc6-prof_2.12.1-0ubuntu6_i386.deb` package using upPIR, HTTP (apache), and FTP (vsftpd) on the same EC2 instances. For comparison's sake, we also downloaded the same file using FTP and HTTP from every available official Ubuntu mirror inside the United States.

Table 1 shows the result of distributing updates via upPIR and other mechanisms. The first thing to observe is that HTTP is slightly faster than FTP. We believe that this is because FTP uses more back and forth communication than HTTP (or upPIR) and therefore suffers the most from latency. HTTP is faster than upPIR, which is expected because the client is downloading 2MB of data from three mirrors instead of 1.5 MB from one mirror. Despite the additional information downloaded, upPIR's time is comparable to FTP on the same hardware. However, unlike HTTP and FTP, upPIR retrieves the update privately. Since our upPIR client downloads from three mirrors, even if two mirrors collude, they do not learn which update the upPIR client is retrieving.

6 Related Work

Impracticality results from researchers including Sion [11], Yoshida [17] and Sassaman [18] reveal inefficiencies in existing PIR schemes. Perhaps most interesting is Sion's argument that many types of computational PIR are presently impractical and, given hardware trends, unlikely to improve from a performance perspective [11]. He argues that it is faster to transfer the entire database than to perform PIR with a large class of proposed schemes.

Olumofin and Goldberg [12] recently provided performance results for a PIR system that does not use the primitives mentioned as impractical in Sion's prior work. Olumofin's resulting system is shown to be one to three orders of magnitude more efficient than transferring the entire database. They use the theoretically optimal block size in their analysis, so their reported performance results are significantly slower. For example, for a 2GB database, they produce a 46KB block in just over 1 second (roughly 370Kbps). upPIR produces a 1MB block in .2 seconds from a 2GB data store, showing throughput of roughly 40Mbps: two orders of magnitude higher throughput. We contacted the authors and discovered that their Chor implementation is similar to ours in performance when given the non-theoretically optimal block size.

Similarly, Melchor [19] provides a fast PIR implementation that uses lattices instead of the XOR-based primitives in our work. The authors mention they aimed to maximize throughput by choosing experimental data that fit exactly within cache (instead of using realistic data sets). As a result, they retrieved 3MB results from a 36MB database to achieve their 230Mbps speed number. On our system with comparable hardware ('local') [20], our implementation can produce results for a 36MB database at over 1Gbps. This demonstrates that careful block size choice results in far greater throughput improvements.

Another common way to try to speed up PIR is to use specialized hardware. Proposals have suggested leveraging GPUs [19], secure co-processors [21] or oblivious RAM [22]. These results show promise, but our work demonstrates

that it is possible to achieve excellent performance simply with universally deployed hardware (commodity CPUs).

7 Conclusion

This work demonstrates that in PIR systems, the theoretically optimal block size (for minimizing communications cost) can be far less efficient than larger block sizes in practice. In fact, it is possible to construct a PIR system with performance similar to production non-PIR systems. We chose to motivate and test upPIR by privately distributing security updates on commodity hardware and show this has performance similar to FTP. Our source code is available at `https://uppir.poly.edu`.

References

1. Cappos, J., Samuel, J., Baker, S., Hartman, J.: A Look in the Mirror: Attacks on Package Managers. In: CCS 2008, pp. 565–574. ACM, New York (2008)
2. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. Journal of the ACM 45, 965–982 (1998)
3. Ding, X., Yang, Y., Deng, R., Wang, S.: A new hardware-assisted PIR with O(n) shuffle cost. International Journal of Information Security 9, 237–252 (2010), 10.1007/s10207-010-0105-2
4. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: STOC, pp. 294–303 (1997)
5. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: FOCS 1997, pp. 364–373 (October 1997)
6. Beimel, A., Ishai, Y., Kushilevitz, E., François Raymond, J.: Breaking the O(n 1/(2k-1)) Barrier for Information-Theoretic Private Information Retrieval. In: FOCS 2002, pp. 261–270 (2002)
7. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
8. Asonov, D., Freytag, J.-C.: Almost Optimal Private Information Retrieval. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 209–223. Springer, Heidelberg (2003)
9. Ambainis, A.: Upper bound on communication complexity of private information retrieval. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 401–407. Springer, Heidelberg (1997)
10. Achieving Practical Private Information Retrieval (Panel @ Securecomm 2006), `http://www.cs.sunysb.edu/~sion/research/PIR.Panel.Securecomm.2006/`
11. Sion, R.: On the Computational Practicality of Private Information Retrieval. In: NDSS 2007 (2007)
12. Olumofin, F., Goldberg, I.: Revisiting the computational practicality of private information retrieval. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 158–172. Springer, Heidelberg (2012)

13. Cappos, J.: Avoiding Theoretical Optimality to Efficiently and Privately Retrieve Security Updates (full version). Technical Report TR–CSE–2013–01, Department of Computer Science and Engineering, NYU Poly (February 2013)
14. AWS Instance Types, `http://aws.amazon.com/ec2/#instance`
15. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An Integrated Experimental Environment for Distributed Systems and Networks. In: Proc. 5th OSDI, Boston, MA, pp. 255–270 (December 2002)
16. Emulab d710 Node Type Information, `https://www.emulab.net/shownodetype.php3?node_type=d710`
17. Yoshida, R., Cui, Y., Shigetomi, R., Imai, H.: The practicality of the keyword search using PIR. In: ISITA 2008, pp. 1–6 (December 2008)
18. Sassaman, L., Preneel, B., Esat-cosic, K.U.L.: The Byzantine Postman Problem: A Trivial Attack Against PIR-based Nym Servers. Technical report, ESAT-COSIC 2007-001 (2007)
19. Melchor, C., Crespin, B., Gaborit, P., Jolivet, V., Rousseau, P.: High-Speed Private Information Retrieval Computation on GPU. In: SECURWARE 2008, pp. 263–272 (August 2008)
20. Compare of Intel E5506 to E5345, `http://ark.intel.com/Compare.aspx?ids=37096,28032`
21. Khoshgozaran, A., Shirani-Mehr, H., Shahabi, C.: SPIRAL: A Scalable Private Information Retrieval Approach to Location Privacy. In: MDMW 2008 (2008)
22. Williams, P., Sion, R., Carbunar, B.: Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: CCS 2008, pp. 139–148. ACM, New York (2008)

Three-Factor User Authentication Method Using Biometrics Challenge Response

Haruhiko Fujii and Yukio Tsuruoka

Nippon Telegraph and Telephone Corporation
{fujii.haruhiko,tsuruoka.yukio}@lab.ntt.co.jp

Abstract. We propose a three-factor authentication method by pointing out the weakness in the two-factor authentication method that uses telephony currently used in Internet banking by adding voice verification, creating a three-authentication method (password, possession of phone, and voice printing) .

Keywords: multi-factor authentication, phone as a token, voice verification.

The use of the two-factor authentication (two-path authentication) method using telephony has recently spread in terms of user authentication in electronic service applications, such as e-government, Internet banking, Amazon, Google, and Facebook. For example, Internet banking involves the following steps. Terminals such as PCs/smart phones send an ID and password a user enters into a server. The server checks the ID and password (knowledge factor), calls back the user's phone, which was registered beforehand, and finishes authentication when the user answers (possession factor). Since an attacker can remotely change the legitimate user's call-transfer setting to his/her telephone with only a stolen PIN, he/she can pretend to be the legitimate user; therefore, this system is not secure. In contrast, we have proposed a method with which users call the server and the server checks the caller's number (caller ID) [1]. Although this method solves the above-mentioned problem, the caller ID can still be changed in several countries, although not in Japan.

To solve this problem, we add voice verification to this method. The server asks the user a question, which is changed each time to prevent a replay attack in the audio line, receives a response, and uses voice recognition to check if the user has said the correct word, and conducts voiceprint recognition to check if the voice print matches the user. When all of these steps have been completed, user authentication (inherence factor) finishes successfully. This method enables a biometrics challenge response by using voice recognition and voice print recognition to prevent a replay attack. This feature is difficult to realize with other biometric methods such as face and fingerprint authentication. Even though another study argues that the crossover error rate of voiceprint authentication on public networks is 6.47% [2], our method is practical due to the use of multiple authentication factors (knowledge: password, possession: phone, and inherence: voice printing).

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 395–396, 2013.

References

1. Fujii, H., et al.: Telelogin: a two-factor two-path authentication Technique Using Caller ID. NTT Technical Review 6(8), 1–6 (2008)
2. Tsuchiya, N., et al.: Speech Identification in VoIP (Voice over IP) System. In: Symposium on Mobile Interactions and Navigation, March 17-18 (2004)

Synthetic Logs Generator for Fraud Detection in Mobile Transfer Services

Chrystel Gaber[1,2], B. Hemery[2], Mohammed Achemlal[1,2], M. Pasquet[2], and P. Urien[3]

[1] Orange Labs, France Telecom, 42 rue des Coutures, BP 6243, F-14066 Caen, France
[2] UNICAEN, ENSICAEN, CNRS, UMR 6072 GREYC, F-14032 Caen, France
[3] Telecom Paristech, UMR 5141, 37/39 rue Dareau 75014, Paris, France
{chrystel.gaber,mohammed.achemlal}@orange.com

Abstract. This article presents a simulator which generates synthetic data for fraud detection. It models fraudsters and legitimate users.

Keywords: synthetic data, simulation, fraud detection.

Mobile payments become more and more popular and, thus, are very attractive targets for fraudsters. As new ways to commit crimes and avoid detection appear, research in the field of fraud is always evolving. Yet, research in fraud detection is limited as publicly available transactional databases containing frauds and groundtruth are scarce [1]. The main cause is that stakeholders are very reluctant to disclose information about frauds and their clients. We address this issue by generating synthetic data of a mobile transaction system. Done in the scope of the European FP7 project MASSIF, our model is based on the mobile-based transaction system and its users described by the MASSIF scenario providers in [2]. We create a tool which can be used by the project contributors and the fraud detection community. Synthetic data are not commonly used in the field of fraud detection although there is a lack of test data. To our knowledge, only one method is used to generate synthetic data for the training and testing of fraud detection systems [3]. Compared to it, ours enables to highlight specific characteristics of fraud detection algorithms because it is not necessarily set up with parameters driven from real data. We model the mobile money transfer platform and the behavior of regular users and fraudsters. Regular users behavior is modeled as a set of habits whereas fraudsters behavior is based on attack patterns. Only superimposed frauds were modeled [1]. A first prototype based on multi-agent models was implemented. We evaluated the model and the data created with the implemented simulator. The period and amount parameters are overfitted. It results in frequent transactions of higher value but a correct value of the total amount of money spent during the simulation. The outcome is encouraging and sets a reference in this field.

References

1. Bolton, R.J., Hand, D.J.: Unsupervised profiling methods for fraud detection. In: Conference on Credit Scoring and Credit Control (2001)

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 397–398, 2013.

2. Achemlal, M., Gharout, S., Gaber, C., Llanes, M., Prieto, E., Diaz, R., Coppolino, L., Sergio, A., Cristaldi, R., Hutchison, A., Dennie, K.: Scenario requirements (March 2011), http://www.massif-project.eu/
3. Barse, E.L., Kvarnstrom, H., Jonsson, E.: Synthesizing test data for fraud detection systems. In: Proceedings of the 19th Annual Computer Security Applications Conference (2003)

Onions for Sale: Putting Privacy on the Market

Aaron Johnson, Rob Jansen, and Paul Syverson

U.S. Naval Research Laboratory, Washington, DC
{aaron.m.johnson,rob.g.jansen,paul.syverson}@nrl.navy.mil

Abstract. We propose that Tor supports the purchase of its services.

Keywords: Onion Routing, Tor, Privacy.

Problem Overview. Onion routing, and in particular the Tor network, is technically well-designed to provide communications privacy. However, the resource constraints of a volunteer network result in unacceptable performance for many users. As a consequence, many users turn to paid services, but even when available they aren't ideal solutions. For example, Virtual Private Networks (VPNs) are often used for anonymous communication and censorship avoidance. However, VPNs are not designed to work against an active or state-level adversary and present a fragile single source of trust, as well as suffering from more subtle flaws [1]. As another example, users hide peer-to-peer file sharing via "seedboxes" that run the P2P protocol at a paid host. However, accessing such services is recognizable, and these solutions again present a single source of trust.

Proposed Solution. We propose that Tor supports the optional purchase of its services to simultaneously provide communications privacy to a new population while improving privacy for the old. In this approach, the existing Tor network infrastructure will be used to provide both paid and unpaid service, but paid service will be prioritized to deliver acceptable performance. Users migrating their activity from existing services will improve their communication anonymity and privacy while at the same time providing additional cover traffic and additional resources for Tor's existing user base.

Technical Approach. There are several technical components needed to incorporate payments into Tor's main services. To provide incentive to pay for service, traffic from users who pay is prioritized over traffic from those who don't using a new circuit scheduling architecture [2]. Communication between hidden service providers who pay and their clients is similarly prioritized. To enhance censorship evasion, users paying for this service are provided with access to a special reserved pool of bridges. To allow anonymous payments, we can take advantage of Bitcoin, although there are other possibilities. Similarly, there are multiple options for investing those payments into Tor network improvement without centralizing control or liability, such as using trusted third-parties that take direct payment. We feel that the challenges in this approach are surmountable and that the benefits outweigh the associated risks.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, pp. 399–400, 2013.

References

1. Appelbaum, J., Ray, M., Koscher, K., Finder, I.: vpwns: Virtual pwned networks. In: The 2nd FOCI Workshop (August 2012)
2. Jansen, R., Johnson, A., Syverson, P.: LIRA: Lightweight Incentivized Routing for Anonymity. In: The 20th NDSS Symposium (February 2013)

Searchable Encryption Supporting General Boolean Expression Queries

Tarik Moataz[1] and Abdullatif Shikfa[2]

[1] Telecom Bretagne, Rennes, France
tarik.moataz@telecom-bretagne.eu
[2] Bell Labs Research, Alcatel-Lucent, Nozay France
abdullatif.shikfa@alcatel-lucent.com

Abstract. We present in this poster a symmetric searchable encryption scheme supporting general boolean search.

Keywords: Searchable encryption, boolean expressions, keyword search.

Searchable encryption is a mechanism that allows a user to store encrypted documents in an outsourced server, and later on query for some of these documents that match a given keyword. All these operations are performed with encrypted data, meaning both the documents and the queries are encrypted in such a way as to minimize leaked information: the server is considered to be semi-honest or honest-but-curious. Searchable encryption is an active research area and has witnessed several interesting schemes since the beginning of the 2000's, and in particular R.Curtmola et al. presented a construction which is asymptotically optimal with respect to search complexity. However, most prior works focused only on single keyword query and thus single keyword searches.

We target a more general model, which encompasses all basic boolean searches -the disjunction, the conjunction and the negation- over encrypted data at the same time. We propose a first construction of symmetric searchable encryption that supports generic boolean search over encrypted data which consists of four algorithms: Gen, Enc, $Query$ and $Test$. The construction is based on an original idea of considering keywords as vectors and using the Gram-Schmidt process to orthogonalize and then orthonormalize them. It further makes use of a very efficient operation, the inner product, to perform searches at the server side. The inner product indeed leverages the orthonormalized keywords to efficiently test if a boolean expression query matches the label corresponding to an encrypted document or not. The label construction consists on the orthonormalized keywords sum, while the queries sent for retrieving encrypted documents are further randomized to guarantee the security of our scheme. As the keywords need to be orthonormalized their size n is necessarily bigger than their number (otherwise we need to pad them). Hence, if the size of orthonormalized keywords is equal to n, then the size of any arbitrary query will be equal to n, the same follows for the size of labels. The details of the scheme are presented in [1].

Reference

1. Moataz, T., Shikfa, A.: Boolean symmetric searchable encryption. To Appear in AsiaCCS (2013)

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, p. 401, 2013.

A Privacy Preserving E-Payment Architecture

Aude Plateaux[1,2], Vincent Coquet[2], Sylvain Vernois[1], Patrick Lacharme[1], Kumar Murty[3], and Christophe Rosenberger[1]

[1] ENSICAEN, 17 rue Claude Bloch, 14000 Caen, France
[2] BULL SAS, Avenue Jean Jaurès, 78340 Les Clayes-sous-Bois, France
[3] Department of Mathematics, 40 St. George Street, Toronto, Canada

Abstract. This poster proposes a secure e-payment architecture for online shopping protecting users' privacy.

Keywords: e-payment, privacy, banking security.

Introduction. Online shopping is becoming more and more interesting for customers because of the ease of use and the choice of products. A vast amount of sensitive information is transferred during such online payment transactions involving privacy problems. Current e-payment schemes, such as 3D-Secure or the SET protocol, attempt to ensure the actors' security without adressing the privacy protection. For instance, when the customer purchases an online service, he/she must provide his/her personal bank information: Personal Authentication Number, Card Verification Value and expiration date. These data are then transferred and can be known by all actors while such knowledge is not necessary.

Proposition. In the proposed architecture, private information is only disclosed when necessary and hidden from both the service provider SP, and the payment providers. This solution is mainly based on the generation of two documents: an electronic bank cheque associated with certificates and a contract between the SP and the customer. In this architecture, we conserve two of the three 3D-Secure domains: the acquirer domain and the issuer domain. The interoperability domain is replaced by an interbank trusted third party. This latter enables communication between banks without disclosing information about the other actors and without adding any additional message. Moreover, this e-payment architecture is fully compliant with the data minimization, sovereignty and sensitivity principles. More particularly, the payment transaction never discloses any customer's bank information. Finally, the customer does not need to have particular cryptographic knowledges.

Conclusion. With an equivalent level of security, the proposed e-payment architecture is more respectful of the privacy than the ones currently used. This scheme also supports the following properties: the customer's basket, as well as the SP's name, are unknown to the customer's bank. Moreover, the customer does not know the SP's bank and is unknown to this latter. Finally, the customer's banking information and the customer's banks are unknown to the SP.

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, p. 402, 2013.

Communication Services Empowered with a Classical Chaos Based Cryptosystem

Gerard Vidal[1,2] and Mikel Hernaez[2]

[1] Institute of Physics, University of Navarra
[2] Enigmedia Corp.
{gerard,mikel}@enigmediacorp.com

Abstract. Enigmedia is an encryption system based on a pseudo-random bit generator (PRBG) which relies on hyperchaotic system, sampled under certain rules in order to avoid any attempt to reconstruct the original trajectory or statistical attacks [1]. The plaintext is XOR'ed with the keystream, obtained from the PRBG. This PRBG is highly efficient, being several orders of magnitude faster than other standards. Furthermore Enigmedia cryptosystem has passed NIST and Diehard statistical test, and it is involved in a validation process. First application featuring Enigmedia is a plug&play $70 USB-device transforming any TV into encrypted HD-videoconference for e-health purpose. This development is also available for mobile & tablet integration throught App. Further work will include sensor integration for monitoring patients.

Keywords: Lightweight Encryption, Real-Time Video, Chaos Systems.

Reference

[1] Vidal, G., Baptista, M.S., Mancini, H.: Fundamentals of a classical chaos-based cryptosystem with some quantum cryptography features. Int. J. Bif. Chaos 22, 1250243 (2012)

A.-R. Sadeghi (Ed.): FC 2013, LNCS 7859, p. 403, 2013.

Author Index